H2020 5G-TRANSFORMER Project

Grant No. 761536

Final design and implementation report on the Vertical Slicer (report)

Abstract

This deliverable provides the final specification of the Vertical Slicer architecture. In particular, it describes its functional components, its northbound and southbound interfaces and the workflows that model its interactions with the other entities of the 5G-T system for the major lifecycle actions on vertical services. Moreover, this deliverable also describes the implementations of the Vertical Slicer prototypes and explains how the Vertical Slicer functionalities are applied to the different 5G-T use cases.

Document properties

Document number	D3.3
Document title	Final design and implementation report on the Vertical Slicer (report)
Document responsible	Giada Landi (NXW)
Document editor	Giada Landi (NXW)
Editorial team	Giada Landi (NXW), Thomas Deiß (NOK-N)
Target dissemination level	Public
Status of the document	Final
Version	1.0

Production properties

Reviewers	Carla Fabiana Chiasserini (POLITO), Jorge Baranda (CTTC), Luis Miguel Contreras (TID), Thomas Deiß (NOK-N), Xi Li (NECLE), Carlos J. Bernardos (UC3M)

Disclaimer

Table of Contents

List of Contributors

Partner Short Name	Contributors
UC3M	Jorge Martin
TEI	Fabio Ubaldi
ATOS	Arturo Zurita
NOK-N	Thomas Deiß
IDCC	Charles Turyagyenda
TID	Luis Miguel Contreras Murillo
ORANGE	Thouraya Toukabri
CRF	Marina Giordanino, Aleksandra Stojanovic
BCOM	Céline Merlet
NXW	Giada Landi, Marco Capitani
CTTC	Ricardo Martínez, Josep Mangues, Jorge Baranda, Laia Nadal
POLITO	Giuseppe Avino, Claudio Casetti, Carla Fabiana Chiasserini
EURECOM	Adlen Ksentini
ITRI	Samer Talat

List of Figures

List of Tables

List of Acronyms

Acronym	Description
3GPP	Third Generation Partnership Project
5GC	5G Core
5G-T	5G TRANSFORMER
5GT-MTP	Mobile Transport and Computing Platform
5GT-SO	Service Orchestrator
5GT-VS	Vertical Slicer
AAA	Authentication, Authorization, Accounting
ADAS	Advanced Driver-Assistance Systems
AN	Access Network
API	Application Programming Interface
AppD	Application Descriptor
BSS	Business Support System
CaaS	Connectivity as a Service
CAM	Cooperative Awareness Message
CAT	Catalogue
CDN	Content Delivery Network
CIM	Cooperative Information Manager
CN	Core Network
CP	Connection Point
CRUD	Create-Read-Delete-Update
CSAR	Cloud Service Archive
CSMF	Communication Service Management Function
DB	Database
DENM	Decentralized Environmental Notification Message
DF	Deployment Flavour
DTX	Discontinuous Transmission
E2E	End to end
EM	Element Management
eMBB	Enhanced Mobile BroadBand
EPC	Evolved Packet Core
EPCaaS	EPC as a Service
ETSI	European Telecommunication Standardization Institute
EVS	Extended Virtual Sensing
FBP	Flow Based Programming
FCFS	First-Come-First-Serve
GRE	Generic Routing Encapsulation
GS	Group Specification
GUI	Graphical User Interface
HSS	Home Subscriber Server
IaaS	Infrastructure as a Service
IFA	Interfaces and Architecture
IL	Instantiation Level
JECRS	Joint Edge and Central Resource Slicer
KPI	Key Performance Indicator
LC	Lifecycle
LCid	Lifecycle Operation Occurance Id
LCM	Lifecycle Management
M&E	Media and Entertainment
mIoT	Massive Internet of Things

mMTC	Massive Machine Type Communications
MANO	Management and Orchestration
MEAO	Multi-access Edge Application Orchestrator
MEC	Multi-access Edge Computing
MEO	Multi-access Edge Orchestrator
MEP	Multi-access Edge Platform
MEPM-V	Virtual Multi-access Edge Platform Manager
MILP	Mixed Integer-Linear Programming
MIoT	Massive Internet of Things
MME	Mobility Management Entity
MNO	Mobile Network Operator
MON	Monitoring
MS	Multimedia Streaming
MVNO	Mobile Virtual Network Operator
NaaS	Network as a Service
NBI	Northbound Interface
NF	Network Function
NFP	Network Forwarding Path
NFV	Network Function Virtualization
NFVI	Network Functions Virtualisation Infrastructure
NFVIaaS	NFVI as a Service
NFV-NS	NFV Network Service
NFV-NSaaS	Network Service as a Service
NFV-NSO	Network Service Orchestrator
NFVO	NFV Orchestrator
NGMN	Next Generation Mobile Networks
NS	Network Slice
NSD	Network Service Descriptor
NS-DF	Network Service Deployment Flavour
NSI	Network Slice Instance
NSMF	Network Slice Management Function
NSP	Network Service Provider
NSSI	Network Slice Subnet Instance
NSSMF	Network Slice Subnet Management Function
NST	Network Slice Template
OAM	Operations, Administration, and Maintenance
ONF	Open Networking Foundation
OSM	Open Source MANO
OSS	Operating Support System
PM	Performance Monitoring
PNF	Physical Network Function
PNFD	PNF Descriptor
PoC	Proof of Concepts
PoP	Point of Presence
QoS	Quality of Service
R1	First Release
R2	Second Release
RAM	Resource Advertisement Management
RAN	Radio Access Network
REST	Representational State Transfer
RM	Resource Management
SAP	Service Access Point

SBI	Southbound Interface
SDK	Service Development Kit
SDM	Spatial Division Multiplexing
SDO	Standard Developing Organisation
SEBASTIAN	SErvice BAsed Slice Translation, Integration and AutomatioN
SLA	Service Level Agreement
SLO	Service Level Objective
SO	Service Orchestrator
SP	Service Platform
SPGW	Service/Packet Data Network Gateway
SPGW-C	Serving/Packet Data Network Gateway Control Plane
SPGW-U	Serving/Packet Data Network Gateway User Plane
SST	Slice/Service Type
TN	Transport Network
TOSCA	Topology and Orchestration Specification for Cloud Applications
TSC	5G-T Service Consumer
TSP	5G-T Service Provider
TTL	Time To Live
UC	Use Case
UE	User Equipment
UPF	User Plane Function
URLLC	Ultra-Reliable Low-Latency Communication
VA	Virtual Application
vEPC	virtual Evolved Packet Core
VIM	Virtual Infrastructure Manager
VL	Virtual Link
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	VNF Descriptor
VNFFG	VNF Forwarding Graph
VNFFGD	VNFFG Descriptor
VSaaS	Vertical Service as a Service
VSB	Vertical Service Blueprint
VSC	Vertical Slicer Customer
VSD	Vertical Service Descriptor
VSI	Vertical Service Instance
WDM	Wavelength Division Multiplexing
WIM	Wide area network Infrastructure Manager
YAML	YAML Ain't Markup Language

Executive Summary and Key Contributions

This deliverable provides the final specification of the 5G-T Vertical Slicer (5GT-VS) architecture, enhancing and extending the initial definition provided in D3.1 [1]. Such evolution is mostly based on an incremental approach, which introduces new functionalities and features without requiring major or disruptive changes to the initial architecture defined in D3.1. The 5GT-VS extensions have been driven by the feedbacks received from the initial 5GT-VS implementation [2] and its integration with the other Proof of Concept (PoC) components of the 5G-TRANSFORMER (5G-T) framework. Such activities have been contextualized in the demonstration of the target 5G-T use cases, in order to identify the 5GT-VS high priority functionalities from the perspective of vertical industries.

As already described in D3.1, the 5GT-VS provides the entry point of the 5G-T system, allowing vertical users to easily request services without the need to specify their infrastructure-oriented requirements but focusing only on their service and business aspects. In this sense, the 5GT-VS offers a catalogue of pre-defined service blueprints, where the vertical user can choose the desired service, customize and instantiate it according to the target business. The internal logic of the 5GT-VS will translate these business-oriented service concepts and requirements into a suitable and efficient set of network slices in terms of Network Function Virtualization (NFV) network services that will be automatically deployed and managed over a shared virtual environment. In this final version of the 5GT-VS architecture, the 5GT-VS logic and procedures have been extended to improve the efficiency of the 5GT-VS decisions from the 5G-T Service Provider perspective (e.g., to enable the sharing of service components among multiple instances of network slices), but also to enhance the offer of automated service functionalities for the 5G-T Service Customer. In this case, the 5GT-VS specifies novel or more advanced features for service composition, automated arbitration among multiple services, manual or automated scaling of vertical services, and support of policies differentiated on a per-tenant basis, applied to SLA management.

The key contributions of D3.3 to the overall 5GT-VS architecture are reported in the core of this report in section 2. In particular, they include:

- The evolution of the 5GT-VS functional architecture (Section 2.1), in terms of functional components, management and operational interfaces, internal procedures for modelling, provisioning and control of vertical services as well as efficient management of network slices.
- The identification of novel or extended features introduced in the 5GT-VS system (Section 2.1.1), with focus on (i) specification of composite services; (ii) advanced management and sharing of network slices composed of multiple slice subnets; (iii) procedures for manual and automated scaling of vertical services; (iv) management of policies on per-tenant and per-service basis, and (v) vertical-driven service configuration.
- The introduction of new functional components and the extension of existing ones in support of such features (Section 2.2). The major changes involve: (i) the 5GT-VS front-end and its Graphical User Interface (GUI) to provide access to the novel functions; (ii) the SLA Management component extended to support policies; (iii) the 5GT-VS Translator and Arbitrator with new logic and algorithms for vertical services arbitration and network slice sharing; (iv) the lifecycle managers of vertical services and network slices to handle service

scaling, composite network slices and aggregated lifecycle actions on correlated service and/or slice instances.

- The implementation of the 5GT-VS prototypes (Section 2.4), highlighting the new functionalities developed in its second release (R2) and how the open source reference implementation has been adopted in other European projects. Particularly relevant, the 5GT-VS prototype has been selected as baseline for the development of network slicing functionalities in other 5G-PPP phase 2 and phase 3 projects, guaranteeing the continuous evolution and enhancement of the 5GT-VS even after the termination of the 5G-T project.
- The analysis of how 5GT-VS functionalities are applied to the vertical services in the 5G-T use cases (Section 2.4).

It should be noted that Section 2 provides an overview of the new functionalities introduced in the 5GT-VS R2. However, in order to provide also a complete and self-contained document that does not require the previous knowledge of D3.1, in appendix (Section 5) we also provide the final specification of the 5GT-VS that integrates all the updates from D3.1 and refinements defined after the first release of the Vertical Slicer. Thus, this appendix can be considered as an independent document that defines the full architecture of the 5GT-VS.

1 Introduction

The 5G-TRANSFORMER (5G-T) Vertical Slicer (5GT-VS) is one of the main building blocks of the 5G-T framework and it provides the entry point for the vertical users to access the whole 5G-T system. The 5GT-VS operates at the top of the overall architecture and it manages vertical services and network slices, thus dealing with service- and business-related logical entities. On the other hand, it delegates all the resource-and infrastructure-related procedures to the underlying blocks, i.e., to the 5G-T Service Orchestrator (5GT-SO) and Mobile Transport and Computing Platform (5GT-MTP), respectively.

The 5GT-VS provides an interface for vertical users, where they can select service blueprints from a pre-defined catalogue and further specify, customize, configure, instantiate and operate their vertical services in an easy manner. In particular, all these procedures do not require verticals to have a detailed knowledge about how the service is structured or how it needs to be deployed and dimensioned in a virtual environment. This approach allows the vertical users to focus only on the business aspects of their services (e.g. how many customers they need to serve, the relative priorities among their own services, or the level of required security). Starting from these high-level specifications, the 5GT-VS translates the verticals' demands into a number of service-specific requirements and identifies the best combination of network slices to deliver the requested services. The 5GT-VS internal procedures arbitrate among multiple, concurrent services, take decisions about their mapping into network slices, potentially sharing their components, and interact with the underlying 5GT-SO to request the instantiation of the associated NFV Network Services (NFV-NS), which are used to deploy network slices.

The initial specification of the 5GT-VS has been reported in D3.1 [1], which introduced the 5GT-VS internal components, its Northbound and Southbound Interface (NBI and SBI) and its workflows. This D3.3 deliverable extends the previous work documented in D3.1, in order to enhance the 5GT-VS functionalities with a number of additional features designed to meet the requirements of vertical services in 5G-T use cases, but also to improve the infrastructure utilization from the 5G-T Service Provider point of view (e.g. through the sharing of network slice components). Regarding the design, management and operation of vertical services, the new 5GT-VS release (R2) includes new functionalities, supporting the definition of composite multiple vertical services, their manual or automated scaling, as well as their configuration. Moreover, it introduces novel arbitration algorithms and procedures for policy management applied to SLA enforcement. The evolution of the 5GT-VS architecture is reported in Section 2.1, which summarizes the new 5GT-VS features and their impact on the overall architecture design, and in Section 2.2 and 2.3, which focus on the changes in the 5GT-VS internal components and information models.

Beyond the evolution of the 5GT-VS architecture, D3.3 provides also an overview of the three 5GT-VS software prototypes developed in WP3 and released in D3.4: one reference implementation that includes all the major functionalities of the 5GT-VS and two more specific implementations, which are developed for particular use case demonstrations or a particular functionality. The focus on specific demonstrations allows to adopt a sub set of functionalities, tailored to the target use case, without the need of using the full 5GT-VS features. It should be noted that the design of the 5GT-VS has been strongly driven by the 5G-T use cases and the requirements of their vertical services. This is particularly reflected on the functionalities implemented in the 5GT-VS prototypes, which are used in the demonstration of the use cases in the context of WP5 activities. The software implementation and the 5GT-VS applicability to 5G-T use cases are addressed in Section 2.4.

In order to keep the main body of the document as short as possible, this is focused only on a summary of the changes or additions introduced in the second release of the 5GT-VS architecture and software implementation. However, in the annex (Section 5) we include an updated version of D3.1 that integrates and extensively discusses these modifications within the overall architecture specification. This annex can be considered as a self-contained document that provides the full specification of 5GT-VS architecture and software prototypes.

2 Update of Vertical Slicer architecture

The architecture of the Vertical Slicer (5GT-VS) evolves the initial architecture proposed in D3.1 [1], with updates mostly based on an incremental approach. Thus, no major or disruptive changes have been applied. Instead, new functionalities and features have been introduced, based on the consolidated requirements derived from the target use cases or feedbacks from the early implementation activities. The feedback from the implementation helped to identify a few gaps in the requirements. Corresponding new requirements have been identified in D1.3 [38] for the 5G-T system and the 5GT-VS requirements have been extended accordingly. The 5GT-VS updates involve additional functional components, new or extended workflows and procedures, updates in the interfaces towards the verticals and towards the Service Orchestrator (5GT-SO), as well as new or enhanced mechanisms adopted in 5GT-VS modules already identified in the previous version of the architecture. This section highlights the major updates of the 5GT-VS architecture, functional components and information models, presenting also an overview of the 5GT-VS software prototype as well as its applicability to the specific Proof of Concepts (PoC) designed for the 5G-T use cases.

2.1 Overview of Vertical Slicer functional architecture

The refined version of the 5GT-VS architecture, reported in Figure 1 and documented in Section 5.3.2, follows the same principles and the same structure initially defined in D3.1 [1]. Its evolution, on an incremental basis, consists of the introduction of additional components and interfaces to provide new or enhanced features. Moreover, beyond the core of the 5GT-VS, a web-based Graphical User Interface (GUI) has been introduced to simplify the access to the 5GT-VS management (for administrators) and operational services (for verticals). The new or highly extended components are represented in the picture by blue boxes; however, minor changes have been also applied to other components of the 5GT-VS, as reported in section 2.2.

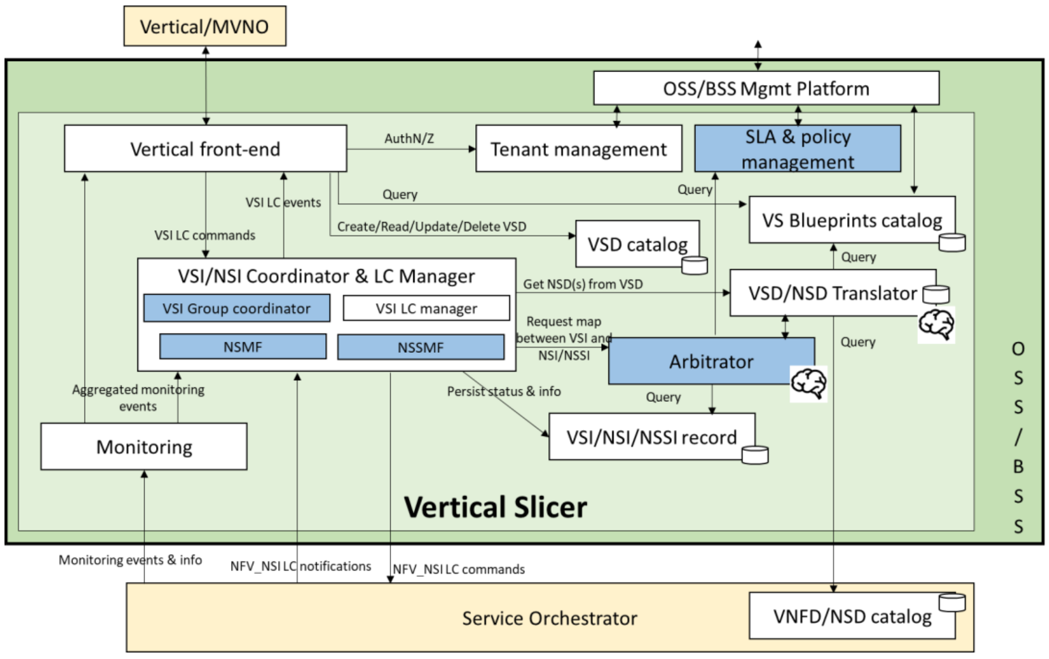


FIGURE 1: THE VERTICAL SLICER (5GT-VS) ARCHITECTURE

As shown in the architectural figure, the 5GT-VS still provides two major entry points: the management and the operational access. The **management** access is reserved for the administrators of the system and allows the configuration of the 5GT-VS with the required information about tenants, Service Level Agreements (SLA), and vertical service blueprints (VSBs). In the refined version, the management functionalities include also the possibility to define policies, which are then applied on a per-service basis according to the tenants' profile. This feature is supported through the introduction of *Policy Management* functionalities integrated within the existing *SLA Management* module, which has been extended to offer a REST API on the 5GT-VS NorthBound Interface (NBI) to create and manage policies. The **operational** access is reserved for the vertical users and allows the request for instantiation, termination, monitoring, and management of vertical service instances. On this side, the main advancements from the previous version have been focused on improving the level of information provided to the vertical users, offering the access to monitoring data and deployment details from the *5GT-VS GUI*, and supporting mechanisms for service scaling and configuration.

Further advancements have also been introduced within the logic of the service requests processing and network slice management strategies in the internal functionalities of the 5GT-VS. In this area, the major novelty is given by the support of composite slices in the *Network Slice Management Function (NSMF)* and the implementation of arbitration algorithms exploiting the concept of "*slice subnets sharing*" within the *Arbitrator* component. Composite slices are structured entities that include one or more slice subnets, where each of them may support a sub-component of the end-to-end service. This concept is aligned with the modelling of network slices, as standardized in the 5G Network Resource Model defined in the 3GPP TS 28.541 [43].

Moreover, mechanisms for automated vertical service and network slice scaling as consequence of arbitration decisions have been introduced through the *VSI Group coordinator* component. This element coordinates ordered lists of actions on groups of vertical service instances that need to be scaled up/down (or in/out), instantiated or terminated as part of atomic procedures related to single verticals' requests.

In the following, we present the new features supported by the second release (R2) of the 5GT-VS (Section 2.1.1), while the new or enhanced 5GT-VS functional components are detailed in Section 2.2.

2.1.1 Major updates in Vertical Slicer R2 features

The second release R2 of the 5GT-VS introduces new features and functionalities with respect to the initial architecture specification reported in D3.1 [1] and the first release (R1) of the prototype delivered in D3.2 [2]. In particular, the following features have been added:

- **Composite services**: the service provider can define a service as a composition of other services, each of them called nested services. A functionality common to several services needs to be described only once and can be referred to upon encoding the services that include such functionality. The components of the 5GT-VS have been extended to handle composite services as well, as described in the next point.
- Advanced management of **network slices composed of multiple slice subnets** (in alignment with the network slice model defined by 3GPP in [43]), with mechanisms for the sharing of slice subnets among several slice instances. This

feature allows, on the one hand, combining different slice subnets, where each of them may provide services at the communication level only or atomic functionalities at the application level, into more complex end-to-end network slices able to deliver composite vertical services with given communication characteristics. On the other hand, the concept of end-to-end slice decomposition allows the re-use of some of the slice resources and components (i.e. a slice subnet) in multiple upper-layer slices, thus extending the concept and the advantages of resource sharing to slice sharing. This allows optimizing the usage of the slices across multiple services whenever a given function or sub-service is sharable among groups of slices. This sharing is still in compliance with the service requirements in terms of per-tenant isolation and defined vertical-driven policies. However, sharing a slice subnet across multiple end-to-end slices may impact its performance and, consequently, the performance of the vertical services running on top of it. To solve this issue, the size of a slice subnet instance must be dynamically adjusted based on the composite slices that are sharing it. In the 5GT-VS, it is a task of the Arbitrator to decide which slice subnets must be re-used and shared to instantiate new end-to-end network slices and how these existing subnets must be modified to support the new traffic and processing load. The procedures to scale slice subnets and instantiate composite network slices are handled within the functional entities responsible of the network slice management, but also require an upper layer coordination to regulate the bunch of correlated and ordered actions needed to scale and instantiate the corresponding NFV Network Service (NFV-NS) instances at the 5GT-SO level.

- Management of **service scaling**. In 5G-T three different types of service scaling have been defined: (i) vertical-driven service scaling, where the scaling of a vertical service is explicitly requested by the vertical sending a request to the 5GT-VS NBI; (ii) automated vertical service and network slice scaling, where a vertical service and/or a network slice is scaled up or down based on a 5GT-VS decision; and (iii) automated NFV-NS scaling, where decisions about scaling are taken at the 5GT-SO level, typically based on autoscaling-rules encoded in the NFV Network Service Descriptor (NSD). In the first two cases, the 5GT-VS has an active role in coordinating the whole scaling procedure (even if just in the second case it acts as decision point for that) and the corresponding lifecycle managers for both vertical service and network slice instances have been updated accordingly in order to manage this kind of procedures. In the third case, the 5GT-VS has only a passive role in the scaling process, but it needs to receive notifications from the 5GT-SO about any scaling actions in order to keep up-to-date information about the amount of resources consumed by each vertical.
- **Management of policies**, on a per-tenant or a per-service basis, as a manner to enforce SLA-related decisions. In particular, the 5GT-VS is configured through administrative actions with policies at the tenant level (i.e. for each vertical). According to these policies, as well as particular service configuration requirements specified by the vertical user through the Vertical Service Descriptor (VSD) or at the instantiation request, the 5GT-VS sends the 5GT-SO directives related to the instantiation of specific NFV-NS through per-service policies. This approach allows, for example, to specify a given optimization criterion during the resource allocation phase, or the value of parameters like the required coverage area of a service instance. Policy management in 5G-T is

aligned with the latest specification of the NFVO northbound interface defined in [11].

- Support of **vertical service monitoring** integrated in the 5GT-VS GUI. This feature allows the vertical to use the web interface of the 5GT-VS to visualize the monitoring data collected, at the 5GT-SO Monitoring Platform level, for its own Vertical Service Instances (VSI) through a dashboard with customized graphs according to service monitoring requirements.
- **Service configuration**, allowing the vertical user to specify per-instance configuration parameters during the service instantiation request. The type of parameters that can be configured are defined in the Vertical Service Blueprint (VSB) and their values, as provided by the vertical user, are encoded in the NFV-NS request sent to the 5GT-SO. This function allows the user to define service-level configurations that can differentiate multiple vertical service instances (e.g. according to the geographical deployment environment, the profile of the verticals' customers accessing the specific service instance, etc.).

A more detailed description of the features mentioned above is provided in section 5.3.1.1.

2.2 Functional components of the Vertical Slicer

The detailed list of 5GT-VS functional components (see the 5GT-VS architecture in Figure 1) is reported in Section 5.3.3, while this section summarizes only the new or enhanced modules introduced in the 5GT-VS R2. The most relevant modifications involve the following functional entities:

- *Vertical front-end:* the operational NBI of the 5GT-VS has been improved to support user service configuration, scaling requests and more detailed reports on active service instances, including monitoring data.
- *SLA and Policy Manager*, extended to manage policies, exposing a REST API for the administrator to create, remove, activate/deactivate, and modify policies expressed on a per-tenant level. Then, such policies are transferred to the 5GT-SO and activated on-demand for specific services.
- *Arbitrator and Translator*, extended to handle composite services and decisions about policy enforcement. The management of composite services has an impact on the translation between VSD and NSD, as well as on the arbitration decisions for the sharing of network slice subnets and their automated scaling.
- *VSI LifeCycle (LC) Manager*, extended to support different types of service scaling procedures.
- *Network Slice Management Function (NSMF) and Network Slice Subnet Management Function (NSSMF):* these components have been extended (NSMF) or introduced from scratch (NSSMF) to handle the composition and sharing of multiple slice subnets within an end-to-end, composite network slice. In particular, composite slices are mapped over composite NFV-NSs including nested network services (i.e. "child" services that are embedded in the composite one and that implement the functionalities of a slice subnet).
- *VSI Group Coordinator:* this new component is introduced to handle lists of correlated actions to be performed on multiple services and slices as a consequence of arbitration decisions.

2.3 Updates in Vertical Slicer information models

A few extensions of ETSI NFV NSDs are needed to extend the information model initially defined for R1, see Section 5.4.3.4. The proposed extension to express latency constraints along with forwarding paths (as defined in the NSD defined by IFA 014 [28]) has been revised and better aligned with previously existing definitions such as the QoS Information Element.

2.4 Vertical Slicer implementation and applicability to 5G-T use cases

As mentioned previously, the implementation of the 5GT-VS prototype has been enhanced and improved in the R2, following the demonstration requirements of the different 5G-T use cases, planned and designed in WP5 [44]. In particular, the 5GT-VS R2 includes additional functionalities (service scaling, management of slice subnets, arbitration among multiple services, service configuration, SLA management) and includes general enhancements in the usability of the system with a more powerful GUI (visualization of monitoring data and details about vertical service instances, more flexible configurability of vertical services, etc.). Table 1 reports the main 5GT-VS functionalities which are applied to each 5G-T use case, as further detailed in Section 5.7.

The 5GT-VS prototype has already been showcased in several public events (EuCNC 2018, ECOC 2018, and ICT 2018, with plans for future demonstrations at EuCNC 2019 and Mobihoc 2019) and it is gathering interest in the 5G research community. In particular, it has been adopted as baseline for a network slicing solution in two other 5G-PPP phase 2 projects (SliceNet[1] and blueSPACE[2]), while its VSB/VSD catalogues and its VSD-NSD Translator will be extended and integrated into the platform under development in the 5G-PPP phase 3 5G-EVE[3] project.

More details about the 5GT-VS implementation and how it is used in other 5G-PPP projects are reported in Section 5.8. The software code is delivered as part of D3.4 [3].

TABLE 1: MAPPING BETWEEN 5G-T USE CASES AND 5GT-VS FUNCTIONALITIES

5G-T Use Case	5GT-VS functionalities
MVNO	Provisioning and termination of vertical service instances.
e-Industry	Provisioning and termination of vertical service instances.
Automotive	Provisioning and termination of vertical service instances. Monitoring of vertical service instances. Arbitration among multiple services with different priorities. Service scaling (automated NFV-NSI scaling option). Service configuration. Support of MEC Applications.

[1] https://slicenet.eu/
[2] https://www.bluespace-5gppp.eu/
[3] https://www.5g-eve.eu/

Entertainment (vCDN)	Provisioning and termination of vertical service instances. Monitoring of vertical service instances. Service scaling (automated NFV-NSI scaling option). Support of MEC Applications.
e-Health	Provisioning and termination of vertical service instances. Composition of vertical services. Management of network slice subnets shared among multiple end-to-end network slices. Programmable request of vertical services from vertical's applications. Service configuration.

3 Conclusions

In this deliverable we have described the evolution of the 5GT-VS architecture final release, where novel features and functionalities have been introduced. Their impact on the internal components of the 5GT-VS have been analysed, highlighting the new functional entities and the extensions to the existing ones in terms of interfaces, information models and internal mechanisms.

This deliverable has also documented the implementation of the 5GT-VS software prototype and its applicability to the 5G-T use cases. This prototype constitutes the main input from WP3 to WP5 activities, where it will be integrated and experimentally validated with the rest of the 5G-T components. In particular, the 5GT-VS reference implementation is adopted in the public demonstrations of the 5G-T use cases. Particularly relevant is the fact that the 5GT-VS prototype has been selected as initial baseline for the development of network slicing components in other 5G-PPP phase 2 and phase 3 projects, as reported in Section 2.4. This is extremely promising to guarantee the continuous evolution and enhancement of the 5GT-VS component itself and, as a consequence, of 5G-T concepts about vertical services and network slices as a whole.

4 References

[1] 5G-TRANSFORMER, D3.1, Definition of vertical service descriptors and SO NBI, March 2018.

[2] 5G-TRANSFORMER, D3.2, Initial Vertical Slicer Reference Implementation, November 2018.

[3] 5G-TRANSFORMER, D3.4, Final design and implementation report on the Vertical Slicer (reference implementation), May 2019

[4] 5G-TRANSFORMER, D1.1, Report on vertical requirements and use cases, 2017.

[5] 5G-TRANSFORMER, D1.2, 5G-T initial system design, May 2018.

[6] 5G-TRANSFORMER, D4.1, Definition of service orchestration and federation algorithms, service monitoring algorithms, 2018.

[7] ETSI GS NFV-MAN 001, V1.1.1, Management and Orchestration, 2014.

[8] 3GPP TR28.801, V15.0.0, Telecommunication management; Study on management and orchestration of network slicing for next generation network, 2017.

[9] 3GPP TS23.501, V15.0.0, System Architecture for the 5G System; Stage 2, 2017.

[10] 3GPP TS28.530, V15.1.0, Telecommunication management; Management of 5G networks and network slicing; Concepts, use cases and requirements, 2018.

[11] ETSI GS NFV-IFA-013, V3.1.1, Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification, 2018.

[12] ETSI GS NFV-IFA-005, Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification", v2.4.1, 2018.

[13] 3GPP TS 38.300, "NR; NR and NG-RAN Overall Description; Stage 2; (Release 15)", v15.0.0, December 2017.

[14] 3GPP TR 28.531, "Management and orchestration of networks and network slicing; Provisioning; Stage 1 (Release 15)", v15.2.0, 2019.

[15] ETSI GS NFV-IFA 028, Management and Orchestration; Report on architecture options to support multiple administrative domains", v3.1.1, 2018.

[16] ETSI GS NFV-IFA 010, "Management and Orchestration; Functional requirements specification", v2.4.1, 2018.

[17] ETSI GR NFV 001, "Network Functions Virtualisation (NFV); Use Cases", v1.2.1, 2017.

[18] ETSI GR NFV-IFA 022, Management and Orchestration; Report on Management and Connectivity for Multi-Site Services", v0.8.2, 2018.

[19] ETSI GR MEC 017, "Mobile Edge Computing (MEC); Deployment of Mobile Edge Computing in an NFV Environment", v1.1.1, 2018.

[20] ETSI GS NFV-INF 003, Infrastructure; Compute Domain", v1.1.1, 2014.

[21] ETSI GS NFV-INF 004, Infrastructure; Hypervisor Domain", v1.1.1, 2015.

[22] ETSI GS NFV-INF 005, Infrastructure: Network Domain, V1.1.1, 2014.

[23] ETSI GS MEC 010-2, V1.1.1, Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management, 2017.

[24] Topology and Orchestration Specification for Cloud Applications Version 1.0. 25 November 2013. OASIS Standard. [Online] http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html

[25] TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0, Edited by Shitao Li and John Crandall. 11 May 2017. OASIS Committee Specification Draft 04, [Online] http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.pdf

[26] The TOSCA Cloud Service Archive (CSAR), [Online] https://www.oasis-open.org/committees/download.php/46057/CSAR%20V0-1.docx
[27] YAML, [Online], http://yaml.org/
[28] ETSI GS NFV-IFA-014, V2.4.1, Management and Orchestration; Network Service Templates Specification, 2018.
[29] ETSI GS NFV-IFA 011, V2.3.1, Management and Orchestration; VNF Packaging Specification, 2017.
[30] ETSI GR NFV-EVE 012 V3.1.1, Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework, 2017.
[31] Luis M. Contreras and Diego R. López, A Network Service Provider Perspective on Network Slicing, IEEE Softwarization, Jan 2018. [Online] https://sdn.ieee.org/newsletter/january-2018/a-network-service-provider-perspective-on-network-slicing
[32] TM Forum, Impact of SDN/NFV on Charging and Billing, [Online] https://www.tmforum.org/resources/how-to-guide/gb989-impact-of-sdnnfv-on-charging-and-billing-r15-5-1/
[33] ETSI GS NFV-IFA 006, V2.3.1, Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification, 2017.
[34] ETSI GS NFV-IFA 007, V2.1.1, Or-Vnfm reference point - Interface and Information Model Specification, 2016.
[35] SONATA D2.2. Architecture Design, [Online] http://www.sonata-nfv.eu/content/d22-architecture-design-0
[36] 5GCity, D4.1: Orchestrator design, service programming and machine learning models, 2018, available at https://zenodo.org/record/2558306
[37] 5G-TRANSFORMER, D4.3, Final design and implementation report on service orchestration, federation and monitoring platform, May 2019
[38] 5G-TRANSFORMER, D1.3, 5G-TRANSFORMER Refined Architecture, May 2019
[39] J. P. Morrisson, Flow-Based Programming, 2nd Edition: A New Approach to Application Development. CreateSpace, Paramount, 2010.
[40] Matilda, D1.2, Chainable Application Component & 5G-ready Application Graph Metamodel, 2018, available at https://private.matilda-5g.eu/documents/PublicDownload/198
[41] SliceNet D7.1 "Cross-Plane Slice and Service Orchestrator", October 2019, work in progress
[42] blueSPACE D5.1, “Extensions to 5G architecture, planning and virtualized network functions for blueSPACE”, June 2018
[43] 3GPP TS28.541, V15.1.0, 5G Network Resource Model (NRM); Stage 2 and stage 3, April 2019
[44] 5G-TRANSFORMER, D5.2, Integration and proofs of concept plan, December 2018.
[45] C. Casetti et al., "Arbitration Among Vertical Services," 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Bologna, 2018, pp. 153-157
[46] C. Casetti et al., "The Vertical Slicer: Verticals’ Entry Point to 5G Networks," 2018 EuCNC, Ljubljiana, Slovenia, June 2018.
[47] ETSI GR MEC 024, “Mobile Edge Computing (MEC); Support for network slicing”, v2.0.12, 2019.

5 Appendix 1: Updated 5G-T Vertical Slicer architecture

This appendix provides the final version of the 5GT-VS specification, based on D3.1 [1], integrating all the updates and refinements defined after the first release of the Vertical Slicer (5GT-VS). Thus, this appendix can be considered as a self-contained document.

5.1 Requirements of the 5GT-VS

Technical requirements on the 5G-T system and the vertical services have been defined already in [4]. These requirements have focused on properties of the vertical services and the corresponding network slices. More general requirements on the 5G-T system are described in [5]. In this section, we will define specific requirements on the 5GT-VS. In Section 5.1.1 we present requirements from the business perspective point of view, while in Section 5.1.2 we present requirements corresponding to the different phases of the lifecycle of vertical services. The notation used for requirements follows the one used in [4]; and generic requirements on vertical services, derived from ETSI NFVI IFA 013 [11] on network services, have been defined in [1].

5.1.1 Business Requirements

The 5GT-VS forms part of the business front-end of the 5G-T system. It is a component that directly interacts with the vertical customer through the service request, which is internally transformed into a network slice instantiation.

To provide context for the business requirements we describe first the kind of services we consider. We base this on a categorization of services defined by 3GPP, see [8].

- The first distinguishing feature is the type of business service. The service provided to 5G-T service customers can be End-to-End E2E connectivity as a service (CaaS), network slice as a service (NSaaS), and virtualized infrastructure as a service (NFVIaaS). The type of business service is further detailed below.
- The second important property is the service type, which is characterized in the form of functional and non-functional (performance, scalability, reliability, etc.) descriptions of the network slice. 3GPP in SA2, see [9] and Table 2, has defined some values for slice/service types (SST) such as enhanced Mobile Broadband (eMBB), Ultra-Reliable Low Latency Communication (URLLC), and massive IoT (mIoT). However, the consumer and the provider may have a common understanding of service behavior and a mutually agreement on a service type definition. For instance, the service provider may dedicate a service type other than V2X to an Extended Virtual Sensing service.

TABLE 2: 3GPP STANDARDIZED VALUES OF NETWORK SLICE TYPES

Slice/Service type	SST value	Characteristics
eMBB	1	Slice suitable for the handling of 5G enhanced Mobile Broadband
URLLC	2	Slice suitable for the handling of ultra- reliable low latency communications
mIoT	3	Slice suitable for the handling of massive IoT

Existing and future networks will be intrinsically multi-service. Then this view from 3GPP can be augmented to cover other potential business needs than 5G. Some potential new SSTs to consider are described in Table 3:

TABLE 3: ADDITIONAL SLICE SERVICE TYPES

Slice/Service type	SST value (proposed)	Characteristics
Enterprise	4	Slice suitable for enterprise connectivity services. This kind of service typically connects a number of sites from a given organization, including voice, data, and video (e.g., connection to private data centers for database access, tele-presence communications, etc.)
(NFV)IaaS	5	Slice suitable for the handling and operation of an (NFV) infrastructure, focused on the availability of raw compute and storage resources

To improve the slice designation accuracy, additional requirements specific to the service type may also be considered. In 5G-T, the vertical service extends the 3GPP communication service to all kind of services a vertical can provide to its end-users. In particular, it includes the functions to provide the E2E connectivity from mobile users to applications and vertical applications as well. In addition, E2E connectivity can be based on 3GPP and non-3GPP managed components. Table 4 summarizes the different levels of categorization.

TABLE 4: HIERARCHY LEVELS OF CATEGORIZATION OF NETWORK SLICE, [8]

Level 1	Level 2	Optional	Attributes and their ranges
Categorization based on Business Service Type (Vertical Service aaS - VSaaS, NSaaS, NFVIaaS)	Categorization based on service characterization eMBB, URLLC, mIoT, other non-standardized slice types	Additional levels (3, 4, 5), e.g. exposure levels. Service types with higher variation in service requirements may be further classified into sub-categories to allow an operator to offer a finer granularity to its customers. For example, eMBB type may be classified into QoS ranges, or QoS based categorization may be classified into user to server and user to user service.	e.g.: Capacity, throughput, delay, number of users, geographical identifications, authentication level etc.

The following functional (F) and non-functional (NF) business-related requirements are identified:

TABLE 5: BUSINESS REQUIREMENTS

ID	Requirement	F/NF
ReqVS.B.01	The 5GT-VS should include a portal to interface with the vertical customer.	F
ReqVS.B.02	The 5GT-VS shall support different kinds of vertical customers.	NF
ReqVS.B.03	The 5GT-VS shall be open and extensible to support any new kind of vertical customer.	NF
ReqVS.B.04	The 5GT-VS should be able to accept the service specification from the vertical customer including both functional and non-functional requirements expected for the requested service.	F
ReqVS.B.05	The 5GT-VS should be agnostic with regard to any collaboration of federated 5G-T administrative domains for the deployment of the service.	F
ReqVS.B.06	The 5GT-VS shall expose appropriate interfaces to external customers to allow them to consume services offered by the 5G-T system.	F
ReqVS.B.07	The 5GT-VS shall be able to perform translation and mapping of the vertical customer service request to network slice(s) implementing such service.	F
ReqVS.B.08	The 5GT-VS shall adhere to industry multi-tenancy requirements including isolation, scalability, elasticity and security, where security is meant to provide protection to prevent attacks, denial of service or information leaking.	NF
ReqVS.B.09	The 5GT-VS shall allow the negotiation and monitoring of service SLAs, with appropriate granularity according to the final service characteristics.	F
ReqVS.B.10	The 5GT-VS shall provide vertical customers with service catalogue information about available service offers and capabilities, in order to facilitate the automated provisioning of services.	F
ReqVS.B.11	The 5GT-VS shall provide a mechanism to perform vertical service accounting and charging. This information should be available internally and externally (for the vertical customer).	F
ReqVS.B.12	The 5GT-VS should be able to support long-live and short-lived services, i.e. the 5GT-VS should be able to express / contain duration characteristics.	F
ReqVS.B.13	The 5GT-VS should be reliable.	NF
ReqVS.B.14	The 5GT-VS should be available (as carrier class component providing 99.999% availability.).	NF

ReqVS.B.15	The 5GT-VS should keep responsiveness for vertical customer requests. The 5GT-VS should provide response times as an interactive system.	NF
ReqVS.B.16	The 5GT-VS shall allow blueprints composed of other blueprints.	F
ReqVS.B.17	The 5GT-VS shall allow VSDs composed of other VSDs.	F
ReqVS.B.18	The 5GT-VS shall allow a vertical to specify which vertical service instance to use in a composed vertical service.	F
ReqVS.B.19	The 5GT-VS shall support the specification of preferred, non-preferred, and prohibited virtual infrastructure providers.	F
ReqVS.B.20	The 5GT-VS system shall allow a vertical to define whether a constituent service of a composed service instance has the same lifecycle as the composed service instance or whether it has a lifecycle of its own.	F
ReqVS.B.21	The 5GT-VS system shall allow a vertical to define whether a vertical service instance can be a constituent service of several composed services, i.e. whether it can be shared among other vertical service instances.	F
ReqVS.B.22	The 5GT-VS shall support to specify indirectly the deployment area based on KPIs[4] of another service.	F
ReqVS.B.23	The 5G-VS should allow vertical-driven service composition.	F

5.1.2 Functional Requirements

The 5GT-VS is involved in the service lifecycle management at different phases, each one having different requirements. The phases are discovery, fulfillment, assurance, and decommissioning. All requirements in this section are functional ones, therefore the corresponding marking in the table was removed.

5.1.2.1 Discovery

The discovery phase facilitates the 5GT-VS to understand what are the capabilities and services (in terms of descriptors) supported by the 5GT-SO. That information will be exposed to the vertical customers for 5G-T service offering.

The following requirements are identified:

TABLE 6: REQUIREMENTS ON THE DISCOVERY PHASE

ID	Requirement
ReqVS.Di.01	The 5GT-VS must provide vertical customers with the means to send detailed requests including information regarding the location of resources, the location of service points, QoS, and charging options.

[4] As an example, extended virtual sensing should cover critical intersections, where 'critical' is defined in terms of occurrence of abrupt braking manoeuvres in the past.

ReqVS.Di.02	Service catalogue entries may contain a service manifest and a price tag (or an indicative price range from which the exact price can be extracted at run-time).
ReqVS.Di.03	The 5GT-VS shall provide the customer with the ability to request a service from the catalogue along with the expected SLA.
ReqVS.Di.04	Service catalogue entries and satisfied service requests should result in an SLA commitment for the corresponding service.
ReqVS.Di.05	The 5GT-VS should be able to keep up to date the network service catalogue entries as informed by the 5GT-SO.
ReqVS.Di.06	The 5GT-VS shall be able to translate the service request to one or more vertical service descriptors reflecting the agreed service levels.
ReqVS.Di.07	The 5GT-VS should keep association among the network slices generated as result of a single service request.
ReqVS.Di.08	The service request (and associated SLA) must contain a parameter describing the service aging or Time To Live (TTL).
ReqVS.Di.09	The 5GT-VS must support both private, i.e. towards specific vertical customer(s), and public dissemination of service offers.
ReqVS.Di.10	The 5GT-VS should provide a mechanism to set-up, scale, and terminate services.
ReqVS.Di.11	The 5GT-VS shall allow a vertical to create several instances of the same vertical service.
ReqVS.Di.12	The 5GT-VS shall allow a vertical to store its service descriptions persistently, and to create, retrieve, update, and delete vertical service descriptions[5].
ReqVS.Di.13	The 5GT-VS shall allow the 5G-T Service Provider (TSP) to onboard vertical service blueprints.

5.1.2.2 Fulfilment

During the service fulfilment phase, the 5GT-VS produces the necessary mapping from the customer request to the network slice templates, i.e. network service descriptors, to properly instruct the 5GT-SO.

[5] A vertical may provide the location of virtual machine images of its virtual applications as part of its service descriptions. These images may have to be certified by the 5G-T system provider before actually on-boarding them to the 5G-T system.

The following requirements are identified:

TABLE 7: REQUIREMENTS ON THE FULFILMENT PHASE

ID	Requirement
ReqVS.Fu.01	Depending on the modality[6] of the contracted service, the 5GT-VS could be required to offer proper configuration and management interfaces to the slice capabilities honoring the service request, in order to manage them similarly as if they were owned and dedicated resources (tenant-managed slices).
ReqVS.Fu.02	Depending on the modality of the contracted service, the 5GT-VS could accommodate a vertical service instance in some existing network slice (provider-managed slices).
ReqVS.Fu.03	The 5GT-VS shall support the vertical to express policies[7].
ReqVS.Fu.04	The 5GT-VS could allow the vertical customer to specify policies[8] associated to the service to describe, e.g. elasticity rules to be enforced when the service requires re-deployment/re-configuration.
ReqVS.Fu.05	The 5GT-VS should allow slice aggregation[9] (and optimization) as part of the slice lifecycle management.
ReqVS.Fu.06	The 5GT-VS shall support a vertical to manage the lifecycle of each of its vertical service instances separately.
ReqVS.Fu.07	The 5G-VS shall support the dynamic on-boarding of NSDs to the 5GT-SO.

5.1.2.3 Assurance

The 5GT-VS informs the vertical customer about events and performance of its vertical service instances. Consequently, the 5GT-VS should gather monitoring information and expose it to the vertical, so that it can take proper action. In addition, the monitoring information can be consumed internally by the 5GT-VS to trigger events like arbitration.

The following requirements are identified:

TABLE 8: REQUIREMENTS ON THE ASSURANCE PHASE

ID	Requirement
ReqVS.As.01	The 5GT-VS must provide the vertical customer with tools to monitor the QoS attained for the requested service.

[6] Modality refers to the possibility of requesting tenant-managed network slices (i.e., request of control and management capabilities of the allocated resources functions) or provider-managed network slices (i.e. the control and management is retained by the provider and the tenant simply uses the network slice).
[7] QoS policies, charging policies, security policies, and regulatory policies where needed.
[8] As an example of such policies, it could be possible to specify the concentration of resources by night for an efficient management of energy consumption during low workloads.
[9] Aggregation means to scale up/down an existing slice, i.e. adding or removing resources according to the demanded services.

ReqVS.As.02	The 5GT-VS should provide a mechanism for the vertical for reward/penalty in service provisioning in case of SLA conformance/failure.
ReqVS.As.03	The 5GT-VS should be able to map the associated SLA into KPIs[10] to be monitored during the slice execution lifecycle.
ReqVS.As.04	The 5GT-VS should be able to support lawful interception mechanisms.
ReqVS.As.05	The 5GT-VS should prevent DoS attacks from a malicious behavior from vertical customers.
ReqVS.As.06	The 5GT-VS should provide isolation among vertical customers' workflows and requests status.
ReqVS.As.07	The 5G-VS shall allow to dynamically set-up a traffic monitoring service in any given 5G network slice.
ReqVS.As.08	The 5GT-VS shall arbitrate resources among vertical service instances of one vertical based on priorities and SLA requirements.
ReqVS.As.09	The 5G-VS system shall support the definition of scaling rules to support service scalability for assuring proper service delivery.

5.1.2.4 Decommissioning

The 5GT-VS also participates in the proper release of services to the vertical once a service instance is no longer required.

The following requirements are identified:

TABLE 9: REQUIREMENTS ON THE DECOMMISSIONING PHASE

ID	Requirement
ReqVS.De01	The 5GT-VS should be able to identify the slice(s) to be decommissioned as a result of a service instance termination.
ReqVS.De.02	The 5GT-VS should be able to identify the monitoring mechanisms to be de-activated as a result of a service instance termination.
ReqVS.De.03	The 5GT-VS should have means for receiving acknowledgement of releasing resources.
ReqVS.De.04	The 5GT-VS should be able to notify the vertical customer about service instance termination.
ReqVS.De.05	The 5GT-VS should be able to keep track of charging and accounting information even after service termination for periods of time according to the local laws.

[10] The SLA will be expressed in terms of service parameters, while the KPIs represent technical parameters. For example, service latency (as part of the SLA) could require monitoring transport network latency but also VM execution latency (as two different KPIs to monitor).

5.2 State of the Art Solutions for the Descriptors

This section presents an analysis and comparison of the current methods/languages to describe the information a descriptor for network services and/or network functions should include.

5.2.1 ETSI Network Service Descriptor (NSD)

ETSI has defined descriptors for NFV technologies in ETSI NFV MAN 001 [7] and IFA 014 [28]. Both documents provide a very detailed description of every information element present in the description of an NFV Network Service (NFV-NS), and the reader can identify the dependencies between them. The specifications give an exhaustive description of the different requirements that every NFV-NS can have as well as the properties that the Virtual Links (VL) and network functions have to meet. To satisfy variations in deployments of several services, the documents introduce the concept of deployment flavours; these are entities that can specify the number of instances, parameters to meet QoS, and monitoring parameters that can be related to the services deployed.

There is still a gap in how to deal with scenarios such as failure recovery, healing and scaling, according to events that might be triggered during the Life Cycle Management (LCM).

5.2.2 TOSCA Network Function Virtualization (NFV)

Topology and Orchestration Specification for Cloud Applications (TOSCA) [24] is a data model standard used to orchestrate NFV services and applications. It helps to define topologies, dependencies, and relationships between virtual applications (VA).

With the growing ecosystem for NFV and the need to model the relationships among VNFs, several organizations proposed languages to model the relationships. Thus. ETSI NFV defined standard fields and features that the languages have to satisfy. Correspondingly, the TOSCA community created the TOSCA NFV simple profile [25] to adapt these features and follow the standard version of ETSI. The work on this profile is still ongoing, extended with the new necessities the contributing companies are finding. Furthermore, several contributing companies are trying to extend the existing standards according to their own needs and TOSCA is trying to introduce the extensions in the standard. The TOSCA NFV profile specifies an NFV specific data model using TOSCA language and following the standardized fields required by the information model of ETSI NFV.

The deployment and operational behavior requirements of each NFV-NS in NFV will be captured in a deployment template, stored during the NFV-NS on-boarding process in a catalogue, for future selection for instantiation. This profile using TOSCA as the deployment template in NFV, defines the NFV specific types to fulfill the NFV requirements.

TOSCA deals properly with the gap mentioned in ETSI NSD in the previous section and tries to specify what are the type of events that can trigger the life cycle management operations.

Two other relevant characteristics of TOSCA that are worth mentioning, due to the implication of the implementation of the VS, are the packaging and the notation used. For the packaging, the TOSCA standard defines the Cloud Service Archive (CSAR) [26], which is a compressed file with a specific structure of files and folders. This file contains all the information required to describe the network service, and the scripts required by the lifecycle of the service. This standard is important as it is becoming one of the de-facto standards for service specification.

The notation used in most of the files within a CSAR package follows the YAML [27] syntax. This notation has been widely adopted due to its simplicity and its capability to model complex relationships. In the 5G-T project we aim to adopt both, i.e., the CSAR packaging and the YAML notations, to aid the on-boarding of new services.

5.2.3 Descriptors in H2020 SONATA project

The SONATA project[11] provides a tightly integrated Service Development Kit (SDK) to design and debug network services and a Service Platform (SP), which manages the execution of network services. The information exchanged between the SDK and the SP relies on descriptors that convey all the information required for deploying and instantiating network services and their VNFs and setting up the networks. One of the key outcomes of the project is, therefore, the information model of these descriptors. In this sense the Sonata project proposes a new information model based on the functional model proposed in ETSI NFV IFA 007 [34].

[35] describes the details of the information model and, in particular, all the proposed extensions in the following items:

- NFV specific entities: VNFD, NSD, etc.
- NFV hosting relationships.
- NFV composition relationships.

In the 5G-T project we have analyzed how to adapt or extend this information model (e.g. for the Vertical Service Descriptors) according to the requirements we have identified, see Section 5.1.

5.2.4 5G-City Service Development Kit

5G-City[12] develops functionality to turn a city into a distributed and multi-tenant edge infrastructure to integrate 5G services administered by a neutral host. The project enables the definition of the services to be executed on the neutral hosts via a Service Development Kit (SDK) [36]. The SDK, implemented as a graphical tool, allows verticals, service providers, or even third-party developers to define services as chains of network functions. Details of the underlying infrastructure are hidden by an abstraction layer. The resulting descriptions consist of four types of abstract models.

- **Network service**: sets of network functions, lists of monitoring parameters, and a topology connecting the network functions.
- **Network functions**: a single deployment flavour defining the needed resources for CPU, storage, and memory.
- **Monitoring parameters**: a list of parameters and thresholds. Examples of typical monitoring parameters are CPU_UTILIZATION, PACKETS_SENT, DISK_READ_OPS.
- **Topology**: A set of links, where each link is defined by the set of endpoints, i.e. the network functions and SLA parameters of the link.

[11] http://www.sonata-nfv.eu/
[12] https://www.5gcity.eu/

In the 5G-T project we use the SDK of 5G-City as an option for vertical-driven service composition, see Section 5.4.4.

5.3 5GT-VS Architecture

This section describes the architecture of the 5GT-VS, as well as its main functional blocks and the algorithms therein. The section is organized as follows: we provide an overview of the 5GT-VS in Section 5.3.1, and we detail its architecture in Section 5.3.2 and its main components in Section 5.3.3. Eventually, we describe the northbound and southbound interfaces of the 5GT-VS in Sections 5.3.4 and 5.3.5. Note that the southbound interface of the 5GT-VS is the northbound interface of the 5GT-SO.

5.3.1 Vertical Slicer Overview

The 5GT-VS is the common entry point for all verticals into the 5G-T system, being part of the OSS/BSS of the administrative domain of a 5G-T 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) [28], which may define forwarding graphs composed of a set of Virtual Network Functions (VNFs) or Virtual Applications (VAs) connected with Virtual Links (VLs), 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.

Beyond these basic functionalities, the second release (R2) of the 5GT-VS includes additional features which are reported in the next subsection.

5.3.1.1 Advanced Vertical Slicer features added in R2

Several advanced features have been implemented in the 5GT-VS R2. The first feature extends the 1:1 mapping among Vertical Service Instances (VSIs) and Network Slice Instances (NSIs) to a N:M mapping; this feature includes the handling of composite services. The second feature provides support for scaling of VSIs. The third feature introduces policy management for ensuring some of the SLA requirements of vertical services. A fourth feature allows verticals to compose vertical services from basic building blocks.

5.3.1.1.1 Management of Network Slices for composite services

In some cases, an E2E vertical service can be decomposed in multiple "elementary" services, each of them implementing a specific set of functions, which can be interconnected and combined together to deliver a "composite" service. An example of this is an elementary mobile communication service including a vEPC instance, which is composed together with further application-level functions to deliver a vertical service for a given application. Moreover, multiple application level functions can be also combined together to provide new services. This is the case of the eHealth service (see Section 5.7.3), where the eHealth monitoring service constitutes a baseline component for more advanced eHealth emergency services.

In 5G-T, the concept of "elementary" and "composite" vertical service is managed through a composition of network slices, where an elementary vertical service is mapped to a *network slice subnet* and a composite vertical service is mapped to a *network slice that includes multiple slice subnets*. This approach opens the door to adopt strategies where a specific slice subnet instance can be re-used in multiple end-to-end slice instances, thus enabling the sharing of network slices and all its benefits in terms of infrastructure utilization efficiency. In fact, this allows optimizing the usage of the slices across multiple services. For example, a vEPC slice subnet can be shared among multiple slices delivered to serve different vertical services with similar mobile connection requirements. Similarly, a slice instance running the eHealth monitoring service can provide a common eHealth backend function to serve multiple instances of the eHealth emergency service.

The logic behind network slice sharing should be service- and customer-driven. In other terms, any 5GT-VS decision about re-using a slice subnet instance should be in compliance with the requirements of the services running on top of it, for what concerns per-tenant isolation. This should be combined with business-oriented considerations at the service provider level, which may be captured through per-vertical policies defining if a given function or sub-service is sharable among groups of slices.

It should be noted that sharing a slice subnet across multiple end-to-end slices may impact its performance and, consequently, the performance of the vertical services running on top of it. Therefore, in order to guarantee the required performance, the 5GT-VS may need to adjust the size of a slice subnet instance according to the number and characteristics of the end-to-end slice instances that are sharing it (this is further described in Section 5.3.1.1.2).

The 5GT-VS Arbitrator is the centralized entity responsible for taking any decision about network slice sharing and potential scaling. In particular, the 5GT-VS Arbitrator decides the following:

- if an upcoming VS request must be served through an existing or a new NSI and, in the former case, if this NSI must be scaled to fit the additional VSI. This approach leads to an N:1 relationship between VSI and NSI, i.e. an NSI can host several VSIs;
- if a new composite NSI must be created from scratch, thus instantiating all its elements ex-novo, or if it can re-use one or more already existing NSIs (the latter ones thus becoming NSSIs). If existing NSSIs can be re-used, the arbitrator should also decide if a scaling action for any of the shared NSSIs is required. This

approach leads to a 1:M relationship between VSI and NSI, i.e. a VSI can be provided through the composition of multiple NSIs;

- if a combination of the two strategies mentioned above is needed, leading to a N:M relationship between VSI and NSI.

Further details about the 5GT-VS arbitrator logic for network slice sharing are provided in Section 5.3.3.2.1.

At the NFV-NS level, network slices and network slice subnets are mapped into composite and nested network services, respectively. Since in 5G-T the description of network slices is modelled through the NSD, the decomposition of a network slice into one or more network slice subnets is also statically encoded in the NSD, which thus becomes a composite NSD. In this sense, the 5GT-VS does not take any decision about how to decompose the slice, but it just selects the target subnets to be shared, based on the available slice instances, the NSD corresponding to the requested service, and the indications from the vertical user about its sharing preferences.

At the 5GT-VS, we can identify two different control modes for the decomposition of end-to-end NFV-NSIs into nested NFV-NSIs, as follows:

- In the **5GT-VS-driven service decomposition**, the 5GT-VS maintains the full control of the nested NFV-NSIs. In other terms, the 5GT-VS requests the 5GT-SO to instantiate the nested NFV-NSIs and, when requesting the instantiation of the end-to-end NFV-NSI, it specifies the IDs of the nested ones. This option allows the 5GT-VS to explicitly control the lifecycle of the nested NFV-NSIs, which can thus be used for sharable network slice subnets and manipulated by the 5GT-VS according to the Arbitrator's logic.
- In the **5GT-SO-driven service decomposition**, the 5GT-VS just requests the 5GT-SO to instantiate a composite NFV-NSI, according to the service structure defined in the NSD, but without specifying the IDs of its nested NFV-NSIs. In this case, the 5GT-SO takes care of creating the nested services in an implicit manner. This option is typically selected whenever the 5GT-VS is not interested in using a given NFV-NSI for a slice subnet that can be shared among multiple end-to-end slices. For example, in the eHealth service, the nested service component related to the emergency management will not be shared among different services, as it happens instead for the eHealth monitoring service (see Section 5.7.3 for further details). For this reason, the instantiation request of the end-to-end eHealth emergency service includes only the ID of the eHealth monitoring component, but not the one of the eHealth emergency component. It is also worth to note that the 5GT-SO, even if managing the service decomposition, does not take any decision about NFV-NS sharing. In other terms, the 5GT-SO always creates new NFV-NSIs for nested services that are not explicitly indicated by the 5GT-VS in the composite NFV-NSI instantiation request. Even if some suitable NFV-NSIs are already available at the 5GT-SO level, they should not be re-used as nested services of the new composite one. This approach allows to keep the service sharing logic only at the 5GT-VS level, while the 5GT-SO is responsible only of the resource sharing logic, without any risk of conflicts between the two decision entities.

5.3.1.1.2 Scaling of vertical services

In 5G-T, three different types of service scaling have been defined, based on the kind of event that triggers the scaling procedure, as summarized in Table 10.

TABLE 10: TYPES OF SERVICE SCALING IN 5G-T

Service scaling type	Decision entity	Scaling executor	Scaling trigger
Vertical-driven VSI scaling	Vertical	5GT-VS and 5GT-SO	"Modify VSI" request received on the 5GT-VS NBI.
Automated VSI scaling	5GT-VS	5GT-VS and 5GT-SO	Arbitration decision at the 5GT-VS.
Automated NFV-NSI scaling	5GT-SO	5GT-SO	Detection of a resource shortage condition, based on an autoscaling rule, at the 5GT-SO.

Each of the service scaling types has different implications on the 5GT-VS functionalities, as discussed in the following.

In the **vertical-driven VSI scaling**, the vertical explicitly requests the 5GT-VS, through its North-Bound Interface (NBI), to modify a VSI providing the ID of a new VSD (see Section 5.10.1.9 for the details of the primitive and see Section 5.4.2 for details on the VSD). This request can be motivated by business-oriented decisions or application level logic that are not handled by the 5G-T system. When receiving the request, the 5GT-VS verifies if the current VSI meets the parameters of the new VSD and, if not, it starts a procedure to scale up/down the VSI itself and, consequently, the NSI and NFV-NSI that are associated to it. An NFV-NSI scaling request is thus sent to the 5GT-SO, which starts its internal procedures to scale the service. The detailed workflow for this procedure is reported in Section 5.5.4.

In the **automated VSI scaling**, the 5GT-VS decides to scale up/down (or in/out) a vertical service or the network slice where it is running, as consequence of an arbitration decision. In particular, such a decision may be derived from the need to instantiate an additional, high-priority service for a vertical that had not negotiated enough resources in its SLAs to accommodate all the range of services it has requested. In order to handle this situation, the 5GT-VS decides to scale down (or in) low-priority services in order to make room for the high-priority ones. An alternative situation that also brings an autonomous scaling decision at the 5GT-VS level is due to the need of sharing a network slice (subnet) instance with additional VSIs or end-to-end NSIs. If the original NS(S)I has not enough resources to support the additional load, the 5GT-VS decides to scale it up (or out). As in the former case, an NFV-NSI scaling request is sent to the 5GT-SO, which in turn scales the service. The internal 5GT-VS workflow is very similar to the previous one, with the only difference that the entity triggering the scaling procedure is the 5GT-VS arbitrator instead of an external request; the scaling execution process remains exactly the same.

In the **automated NFV-NSI scaling**, the 5GT-VS actually, does not take any decision, since the entire scaling process is managed at the 5GT-SO in terms of both decision and execution. In particular, the 5GT-SO uses the monitoring data collected by the 5G-T Monitoring Platform to detect a resource shortage condition that may impact the performance of the service, triggering a scaling up (or out) procedure to prevent any SLA breach. This action is performed according to auto-scaling rules encoded in the NSD, which identify the "triggering condition" (e.g. a monitoring parameter that is higher than a threshold value for a given time interval) and define the kind of scaling action(s) to be performed. The detailed 5GT-SO procedure for the automated NFV-NSI scaling is described in D4.3 [37]. Once the procedure is successfully terminated, the 5GT-VS is notified by the 5GT-SO and it updates its internal records about the affected NS(S)Is and VSIs.

As captured in Table 10, in the first two cases, the 5GT-VS has an active role in coordinating the whole scaling procedure (even if just in the second case it acts also as a decision point for that) and it cooperates with the 5GT-SO to handle the scaling procedure. Instead, in the third case, the 5GT-VS has only a passive role since all the scaling process is managed at the 5GT-SO. However, it still needs to receive notifications from the 5GT-SO about the result of the scaling actions, in order to keep the synchronization between the current status of the NFV-NSI (managed at the 5GT-SO) and the status of the associated NS(S)I and VSI, which are directly managed by the 5GT-VS. Moreover, these notifications allow the 5GT-VS to keep trace of the amount of resources consumed by each vertical.

5.3.1.1.3 Policy Management

The SLA requirements of vertical services, as defined in [4] cannot be treated in a uniform way by the 5G-T system. Some of the SLA requirements such as latency, user data rate, or mobility can be translated into SLA requirements or deployment flavours within NFV NDSs. E.g. latency can be translated into latency of a Network Forwarding Path (NFP), see Section 5.4.3.4.2, while mobility can be translated into corresponding capacity of the control plane for handling handovers. Other SLA requirements such as preferred or prohibited infrastructure providers, energy reduction, or service creation time are better handled by policies for the 5GT-SO and 5GT-MTP entities. Here, 'policy' is understood as an additional rule or optimization target. E.g. the 5GT-SO should orchestrate a vertical service such that energy consumption is minimized.

The first class of SLA requirements described above simply has to be satisfied by the 5GT-SO. Still, the 5GT-SO may have several choices about how to satisfy these SLA requirements. Policies are understood as directives to the 5GT-SO to make decisions in favour of a specific target, where the targets correspond to the second class of SLA requirements. As an example, the amount of gNbs used to provide coverage for a vertical service could be minimized. Additionally, features supporting discontinuous transmission (DTX) or putting cells of unused carriers to sleep mode could be activated to reduce energy consumption of the RAN.

The 5GT-VS has been extended to activate policies and associate them with NSIs based on the policy management interface defined in [11]. 5G-T has contributed to this interface by defining the associate and disassociate operations via change requests[13]. In addition, the Translator component has been extended to map SLA requirements to policies. Note that the policies associated to a VSI are not included in the corresponding NSD, but they trigger calls of the associate operation at the Vs-So interface (see Section 5.3.5 and

[13] NFVIFA(18)0001001, IFA013ed321 CR add policy associate disassociate operations, NFVIFA(18)0001002, IFA007ed321 CR add policy associate disassociate operations, NFVIFA(18)0001003, IFA008ed321 CR add policy associate disassociate operations, NFVIFA(18)0001012, IFA010ed321 CR add policy associate disassociate operations.

Section 5.5.3 for the details of the interface and the associated workflows). Although the policy management interface has been standardized in [11], ETSI NFV specifies neither a specific notation for policies nor a set of standardized policies. Therefore, the service provider has to ensure the consistency of the meaning of specific policies between the 5GT-VS and 5GT-SO.

5.3.1.1.4 Service Composition

Providing a catalogue of Vertical Service Blueprints (VSBs) to the verticals makes it very easy for them to define their services. Essentially a vertical has to select a VSB from the catalogue and provide the missing information. Providing the VSBs, corresponding VNFs, and Virtual Deployment Units (VDU) is left to the service provider. When the number of verticals using 5G-T increases, we expect the verticals to ask for variations of the vertical services in the catalogue. To some extent, this can be handled by additional parameters, but, at some point, additional VSBs for different variations have to be added to the catalogue. This makes the maintenance of the catalogue and the corresponding descriptors cumbersome. It would be easier to offer the verticals a set of smaller building blocks, e.g. different variants of similar functionality, and allow the verticals to compose them into a vertical service. This would provide flexibility to the vertical and at the same time ease the maintenance burden for the service provider.

Therefore, we propose an approach for composing services by verticals, which we name *vertical-driven service composition* with the aim to increase the flexibility for the verticals, without increasing the complexity. The approach has to fit into the existing 5G-T system, i.e. the 5GT-VS should handle the service composition and map this into vertical service and network service descriptors, and onboard them as usual to the 5GT-SO. For the 5GT-SO and 5GT-MTP, however, no changes should be needed. The approach is described in more detail in Section 5.4.4.

5.3.2 Vertical Slicer Architecture

The final architecture of the 5GT-VS is shown in Figure 2. As mentioned, the 5GT-VS is part of the provider's OSS/BSS and interacts with the vertical (or the M(V)NO) through its NBI and with the service orchestrator through its southbound interface (SBI). The 5GT-VS is expected to interact as well with other components of the OSS/BSS Management Platform of the TSP. This in turn interacts with the TSP, but this interaction among management platform and TSP is out of scope.

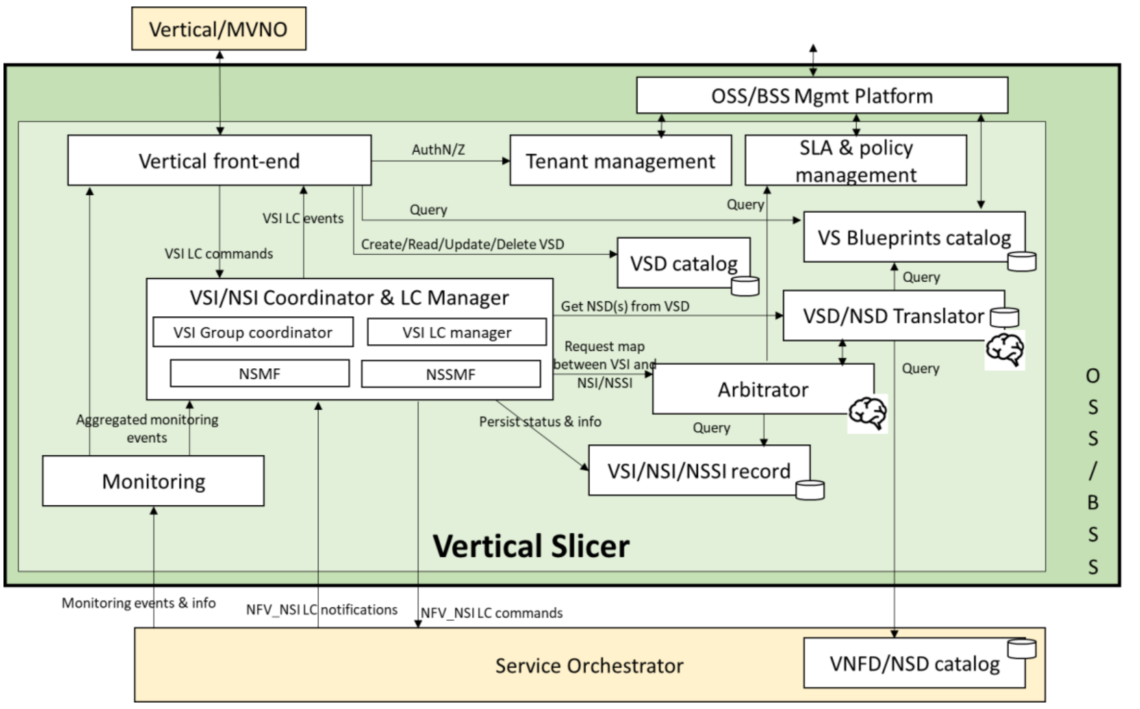


FIGURE 2: THE VERTICAL SLICER ARCHITECTURE

Looking at Figure 2, at the top we find the Vertical front-end, which provides an entry point to receive requests from the verticals/MVNO about the service management and monitoring. Importantly, the Vertical Front-end presents the vertical user with Vertical Service Blueprints (VSB), see Section 5.4.1, stored and handled through the related catalogue, and supports the vertical user in providing the parameters therein to obtain the corresponding Vertical Service Descriptor (VSD), see Section 5.4.2.

The lifecycle (LC) of Vertical Service Instances (VSIs) and the corresponding Network Slice Instances (NSIs) are handled through the component called “VSI/NSI Coordinator & LC Manager”. This component, described in Section 5.3.3.4, implements the central engine of the 5GT-VS: it manages the association between VSIs and NSIs, regulating the sharing of network slices among different vertical services and their decomposition in network slice subnets. Moreover, it also handles the finite state machines for VSIs and NSIs/NSSIs lifecycle, coordinating commands and events associated to them. The network slice management is handled through requests for instantiation, modification or scaling and termination of the corresponding NFV-NSs, interacting with the 5GT-SO. The status and the current characteristics of VSIs and NSIs/NSSIs are stored persistently in the VSI/NSI records.

While the “VSI/NSI Coordinator & LC Manager” manages the lifecycle of slices and services, the complete 5GT-VS decision logic is implemented in the VSD/NSD Translator and in the Arbitrator. The VSD/NSD Translator, described in Section 5.3.3.1, selects the descriptors of the NFV network services that are able to support the requested vertical services. In particular, the VSD/NSD Translator identifies the network service deployment flavour (DF) and instantiation level (IL) (e.g., number of VNF instances, amount of vCPU or RAM for VNFs, bandwidth of VLs) most suitable to guarantee the performance and the characteristics defined in the VSD. The Arbitrator, described in Section 5.3.3.2, makes decisions about the sharing of network slices among different vertical services and the scaling of vertical services based on service priority and resource budget in verticals' SLAs. The Arbitrator takes also decisions about policies that should be adopted for specific VSIs (e.g. based on the vertical profile or specific parameters provided in the VSI request) and, consequently, how to apply these policies at the network slice and NFV-NS level.

Finally, the 5GT-VS Monitoring Service, described in Section 5.3.3.3, interacts with the 5GT-SO to collect monitoring data about the established NFV network services and correlates or aggregates these data in order to produce metrics and KPIs for network slices and vertical services. These metrics may be used as input for SLA verification or to make decisions about the lifecycle of a network slice, for example triggering a scale up action in case of decreasing performance.

The 5GT-VS NBI, detailed in Section 5.3.4, 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 (see Figure 3). In particular, the Ve-Vs reference point provides mechanisms for VS blueprint (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, policies and VSBs.

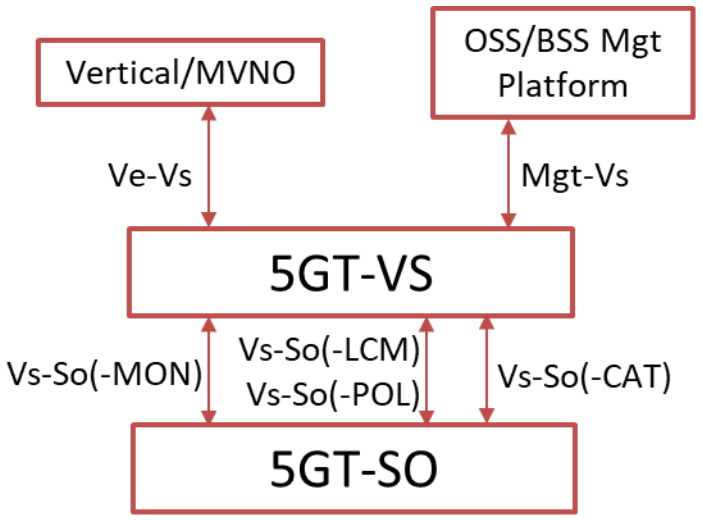


FIGURE 3: VERTICAL SLICER REFERENCE POINTS

The 5GT-VS SBI, and thereby the 5GT-SO NBI, described in Section 5.3.5, allows the interaction with the Service Orchestrator (5GT-SO). Its definition is based on the ETSI NFV IFA 13 [11], which defines the NBI of an NFVO. The 5GT-VS SBI implements the following reference points:

- Vs-So(-LCM), for the operation of network services. It offers primitives to instantiate, terminate, query, update and re-configure network services or receive notifications about their lifecycle;
- Vs-So(-MON), for the monitoring of network services and VNFs through queries or subscriptions and notifications about performance metrics and failures;
- Vs-So(-CAT), for the management of NFV NSDs and VNF package descriptors and storing them in catalogues, including mechanisms for their on-boarding, removal, updates and queries;
- Vs-So (-POL), for the management of policies to be associated to NFV-NSIs.

5.3.3 Vertical Slicer functional components

5.3.3.1 The VSD/NSD Translator Module

The VSD/NSD Translator module is an entity within the 5GT-VS that maps the vertical's requirements into technical specifications needed by the 5GT-SO to perform an NFV-NS deployment.

To define a vertical service or prepare it for deployment, the vertical first selects a VSB, which contains parts that have to be filled in with the requirements it demands. Once the vertical provides actual values for the missing parts in the VSB, it becomes a Vertical Service Descriptor (VSD): a high-level specification of a network service, based on a service logic perspective rather than a resource-based perspective.

The VSD is then mapped into a Network Service Descriptor (NSD) through the following steps: (1) the Translator queries the NSD catalogue for the NSDs referenced in the VSD; (2) the Translator queries the Virtual Network Function Descriptor (VNFD) catalogue for the VNFs referenced in the NSD; (3) adopting service-specific translation rules, the Translator selects the deployment flavour and instantiation level for each VNFD and each Virtual Link Descriptor (VLD), using the QoS restrictions of the VSD (such fields will be finalized by the Arbitrator later on).

A relevant example of VS is a video streaming service. The VSB contains fields for service type, number of users, and availability, which have been completed by the vertical in this example as follows:

- Service type: video streaming.
- No_users: 10.000.
- Time_availability: 24 h/day - 7days /week.

Upon receiving the VSD, the Translator looks for the VNFDs of firewall, load balancer, and streaming servers. Such VNFDs are referenced in the NSD and connected with the VLs composing the VNFFG shown in Figure 4. Some of the VLs and VNFs are backup components to fulfil the availability requirement, while the number of streaming VNF instances is set to a suitable value to satisfy both availability and number of users to serve. With a less stringent availability requirement, the Translator would choose a VNFFG without redundant components and paths. The NSD also specifies the initial values of VNF instances and VLs deployment flavours and instantiation levels, as well as monitoring and autoscaling or auto-healing rules[14] indicating how the service should be modified in the case some VNFs go down. Then this NSD is given as input to the Arbitrator, in order to finalize the CPU and bandwidth requirements specified in the aforementioned deployment flavours.

[14] The specific format of autoscaling rules is not specified in the standard IFA 014; in 5G-T they have been modelled as a set of "conditions" based on monitoring values and a set of "reactions" (e.g. scaling to a target instantiation level).

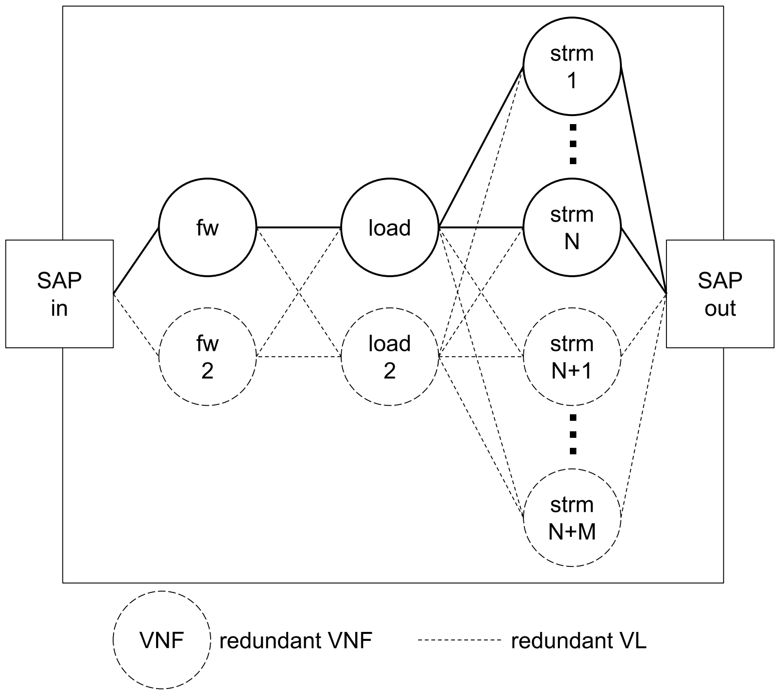


FIGURE 4: VIDEO STREAMING VNFFG

5.3.3.2 The Arbitrator Module

SLA management is a key aspect in the provisioning of services to a vertical on top of 5G networks. Any degradation of the SLA negotiated between the provider's OSS/BSS and the vertical can impact not only the technical behaviour of the vertical service but also the reputation or business leadership of the vertical itself. Finally, legal consequences could occur depending on the nature of the service being provided: SLAs are more than just sales agreements, they are also part of a legally binding contract.

The deployment of a vertical service is a multi-dimensional problem, where various objectives, resource types (e.g., CPU, memory, storage) and autonomous infrastructure providers can be involved. Thus, interdependencies exist among the different resources to be provisioned and configured.

The vertical, as 5G-T service consumer, will specify some Service Level Objectives (SLOs) adapted to its service needs. Examples of SLOs can be, e.g., a maximum end-to-end latency of 20 ms for the Extended Virtual Sensing (EVS) use case of an automotive vertical. Vice versa, the provider's OSS/BSS can state different service level classes, representing distinct guarantees on resource availability. The matching between the SLOs desired by the vertical and the service level classes offered by a provider will define an SLA. The contents and terms of an SLA will be inevitably used as specific directives for configuration and orchestration of services and resources in provisioned network slices. A slice life-cycle operation is effective if an acceptable trade-off is found between satisfying the terms in its SLA and satisfying the rest of the service requests in the system. It is here where arbitration mechanisms are required for managing such trade-offs.

The Arbitrator within the 5GT-VS ([45], [46]) is the entity where these mechanisms are implemented and that provides the 5GT-SO with support for service deployment. It operates based on the SLA, as well as the information on each service provided by the vertical, namely: (1) the service priority level, (2) the VNFFG representing the service, (3) the relative CPU requirements of the VNFs in the VNFFG, as well as their memory/storage requirements, and (4) the vertical's quality of service (QoS) requirements (e.g., end-to-end latency, service availability, or reliability level). Note that such information is described in the NSD created by the VSD/NSD Translator for the received VSD.

The Arbitrator's tasks are twofold:

1. To decide how to map new vertical services in NSIs, allowing also multiple vertical services to share one or more NSIs or NSSIs.
2. To determine the deployment flavours associated with each service and network slice, such that the vertical's QoS requirements are met, while accounting for the services priority level. Note that, upon the request of a new service instance by a vertical, the Arbitrator may need also to update the deployment flavours and instantiation levels associated with previously allocated service instances.

These two tasks are detailed in the following subsections for the case of services requested by the same verticals.

5.3.3.2.1 Sharing of network slices among vertical services

During the instantiation of a vertical service, the Arbitrator is responsible for making decisions about the sharing of NSIs and NSSIs among different vertical service instances. A typical example could be the sharing of the same vEPC among different vertical service instances with similar requirements in terms of mobile access, or the sharing of a service component for the collection of vehicle messages among multiple automotive services that make use of the same messages.

The decision about slice sharing is made by the Arbitrator analysing the triple <NSD; Deployment Flavour (DF); Instantiation Level (IL)> initially computed by the VSD/NSD Translator, which describes the structure, the characteristics and the cardinalities of a potential NSI able to support the requested VSI. Starting from the NSD, the Arbitrator verifies if one or more existing NSIs can be re-used to accommodate the new VSI or at least part of it, e.g. one or more nested NFV-NS that may be included in the NSD. Obviously, the decision is also affected by the constraints specified by the vertical in the VSD, in particular about the possibility to share (part of) the service components with existing ones already instantiated for the same or different verticals.

If no suitable NSIs are already available, the Arbitrator simply decides to create a new NSI, based on the NSD originally computed by the VSD/NSD Translator. It will then proceed with the update of the deployment flavour and instantiation level initially set by the VSD/NSD Translator, as necessary (see Section 5.3.3.1). The "VSI/NSI Coordinator & LC Manager" is then responsible for managing the actual instantiation of the new network slice and its slice subnets, triggering the 5GT-SO to deploy the required NFV-NSs.

On the other hand, if the Arbitrator finds existing NSIs that can be shared, it must determine if and how they should be modified to accommodate the additional service. For this, it interacts with the VSD/NSD Translator, providing as input the VSDs of all the VSIs that will share resources from the given set of network slices. The VSD/NSD Translator should then combine the requirements from the various VSDs and return a suitable set of triples <NSD; DF; IL> defining the characteristics and the size of the target NSIs. Assuming that the elaborated solution is compliant with the vertical's SLAs, the

Arbitrator identifies the NSIs to be modified and the ones to be created entirely, according to the result provided by the VSD/NSD Translator. As in the previous case, it is up to the "VSI/NSI Coordinator & LC Manager" to manage the whole set of procedures to execute the required actions.

5.3.3.2.2 Computation of deployment flavours and instantiation level

As far as the deployment flavours and instantiation levels are concerned, let us assume for now that, to support the requested vertical services, new NSIs should be deployed. We use the following algorithm for the Arbitrator. Let us denote by *C, B, M* and *S* the total amount of, respectively, CPU, bandwidth, memory, and storage that the vertical's services can use per SLA. First, the Arbitrator orders the service instances that the vertical wants to deploy, or it has deployed, from the highest priority level to the lowest. It then considers the highest-priority service instance, say, *s*, and allocates memory and storage based on the needs exhibited by the VNFs in the VNFFG representing *s*. A more complicated procedure is instead required for CPU and bandwidth allocation, since this depends on the VNF placement decisions made by the 5GT-SO. For clarity of presentation, let us focus on the service latency as the main performance metric, and denote by D_s the maximum latency that service *s* is allowed to experience. Two components contribute to the service latency: (i) the processing time, due to the execution of the VNFs in the VNFFG, and (ii) the network delivery time, which is due to the time needed to transfer data from one VNF to the next in the VNFFG. While the former depends on the CPU allocated to the VNFs execution, the latter depends on the deployment decisions made by the 5GT-SO, and on the bandwidth associated with the VLs connecting the servers when the 5GT-SO places VNFs on different servers.

Ideally, all VNFs in the VNFFG, whose set is denoted by *V*, are deployed within the same NFVI Point of Presence (NFVI-PoP). In this case, the network travel time is minimal and the complete latency budget can be used as processing time, reducing the amount of required processing resources. I.e., the bandwidth required for data transfers over VLs for service *s*, β^b, can be set to zero, while the allocated CPU, μ^b, can be computed so that:

$$\sum_{v \epsilon V} \frac{1}{f_v u^b - \lambda_v} \leq D_s \qquad \text{and} \quad \mu^b \leq C \qquad (1)$$

where, for simplicity, we neglected the delay due to memory/storage access. In the above equation, f_v is the relative computational requirement of VNF *v*, such that; $\sum_{v \in V} f_v = 1$. Moreover,

- the first inequality imposes that the latency experienced by the service does not exceed the maximum value. The total latency is given by the sum of the latency contributions due to all VNFs in the VNFFG. Each contribution is computed by modeling the generic VNF v as a FIFO M/M/1 queue with $f_v \mu^b$ as the output rate and λ_v as the service request rate input to v;
- the second inequality imposes that the CPU allocation does not exceed the vertical's amount of CPU, *C*, that is available to the vertical.

However, having all VNFs in one PoP might be overly restrictive. Therefore, the Arbitrator determines a second, more flexible, deployment flavour. As a smaller portion of the latency budget would be available for processing, the required amount of processing resources is higher. Specifically, in the worst case, each VNF in *V* is deployed in a different server, thus implying the need for bandwidth allocation. Specifically, the allocated CPU, μ^w, and bandwidth, β^w, now should satisfy the following constraints:

$$\sum_{v \epsilon V} \frac{1}{f_v \mu^w - \lambda_v} + \sum\nolimits_{(u,v) \epsilon E} \frac{d_{v,w}}{f_{u,v} \beta^w} \le D_s \tag{2}$$
$$\mu^w \le C \; ;$$
$$\beta^w \le B$$

where $f_{(u,v)}$ is the relative bandwidth requirement for the VL connecting VNFs $_u$ and $_v$. Also,

- the first inequality accounts for the latency due to both the VNF execution and the travel time over the VLs connecting any two adjacent VNFs (*E* denotes the set of edges in the VNFFG);
- the second and third imposes that both the total CPU and bandwidth allocations do not exceed the corresponding budget available to the vertical.

Once the Arbitrator has determined the values (μ^b,0) and (μ^w, β^w) for the tagged service, it proceeds with the second service in the list, following the same steps as above but using the remaining amount of resources available to the vertical in terms of CPU (*C*) and bandwidth (*B*).

Note that, using the above values for the CPU and bandwidth that should be allocated for a service, the Arbitrator can determine the corresponding per-VNF and per-VL values based on the relative "weight" that, respectively, VNFs and VLs have in the NSD initial setting. The Arbitrator will thus update the deployment flavour in the VLDs and the VNFDs of the NSD created by the VSD/NSD Translator. The updated NSD is then returned to the VSI/NSI Coordinator & LC Manager, which will request its instantiation from the 5GT-SO. The 5GT-VS will pass a list of initial deployment flavours to the 5GT-SO, which will try to instantiate them in order until one deployment flavour could be instantiated successfully. This requires an extension of the life cycle management interface of [11] to pass such a list of initial deployment flavours instead of just a single one. After the 5GT-SO has made its deployment decisions, it will notify the 5GT-VS about the current resource allocation (e.g., in terms of characteristics of the instantiated VNFs) so that the Arbitrator can compute the new amount of resources that are available to the vertical.

Let us now consider the case where the vertical service to be processed can be supported on an existing NSI, or, using an existing NSSI. In this case, the Arbitrator should add the traffic load due to the newly requested vertical service to the load of the existing VNFs/VLs, and re-compute the necessary CPU and bandwidth, as described above. Again, the Arbitrator will use the CPU and bandwidth values obtained for the different cases, to update the deployment flavour of the involved VLDs and VNFDs.

It is clear that, in case of resource shortage, some, lower-priority services, may not be accommodated, or may be terminated due to the need to re-allocate resources to higher priority services.

To summarize, the Arbitrator module allows the 5GT-SO to make deployment decisions meeting the vertical's indications even if *(i)* it is unaware of higher-layer information like the SLA between the vertical and OSS/BSS, and *(ii)* the total amount of available resources is not sufficient to adequately deploy all requested services.

5.3.3.3 The Monitoring Service

In the 5G-T framework, each architectural component (i.e. 5GT-VS, 5GT-SO, 5GT-MTP) includes a monitoring service able to provide performance metrics and failure reports targeting the logical entities managed by each component (see D1.3 [38] for further details). The monitoring service at each architectural layer operates on specific resources; in particular, the 5GT-VS Monitoring Service is responsible for producing monitoring data about network slices and vertical services and for elaborating more elementary monitoring data about VNFs and NFV network services (NFV-NS), as retrieved from the underlying 5GT-SO Monitoring Service. Figure 5 shows the different interfaces of the 5GT-VS Monitoring Service. A description of service monitoring for the complete 5G-T system as well as monitoring requirements per use case is presented in [6].

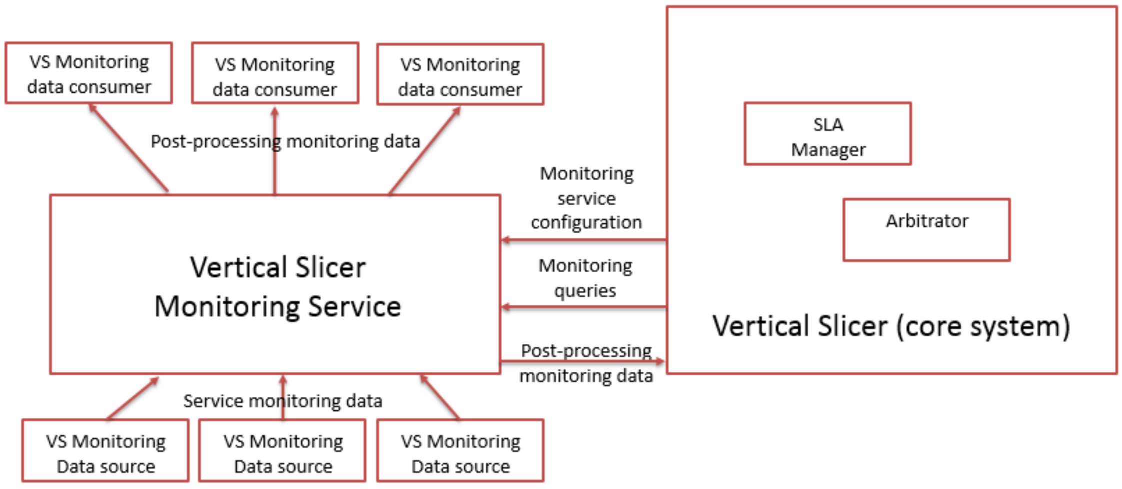

VS Monitoring data consumer
VS Monitoring data consumer
VS Monitoring data consumer
Post-processing monitoring data
Monitoring service configuration
Monitoring queries
Post-processing monitoring data
Vertical Slicer Monitoring Service
SLA Manager
Arbitrator
Vertical Slicer (core system)
Service monitoring data
VS Monitoring Data source
VS Monitoring Data source
VS Monitoring Data source

FIGURE 5: 5GT-VS MONITORING INTERFACES

The 5GT-VS Monitoring Service provides monitoring data of VSIs as input for internal decisions for SLA management at the Arbitrator. The 5GT-VS provides also monitoring data about the deployed VSIs to the verticals, to feed the internal processing of vertical applications, where needed. According to ETSI GS NFV-IFA013 [11], an NFVO exposes interfaces towards the OSS/BSS. The NFV-NS Performance Management interface and NFV-NS Fault Management interface allow the 5GT-VS to get monitoring data from the 5GT-SO, see Section 5.3.5 for further details on the 5GT-SO NBI. In Figure 5 the 5GT-SO is depicted abstractly as VS Monitoring Data source.

The 5GT-VS monitoring service should be able to expose the monitoring information it receives from the 5GT-SO Monitoring Service to the VS Front-end. Actually, this exposure can be part of the SLA. Thus, a field about monitoring, in addition to the one on SLA, has to exist in the VSB and VSD. This field allows a vertical to specify which monitoring data to receive. Monitoring data for VSIs can also be visualized by the vertical through the graphical interface of the 5GT-VS, making use of the dashboard tool offered by the monitoring service.

The monitoring data about VSIs provided at the 5GT-VS NBI should be aggregated and/or abstracted. 3GPP [8] defines 2 types - performance measurement and alarm - of data aggregation for network slices. Within each type there are two categories of data: network related and vertical service related. The 5GT-VS Monitoring Service receives and shares monitoring data from the 5GT-SO via the 5GT-SO-MON interface or generated internally. The VSI LC Manager and/or NS(S)MF send a request to the 5GT-VS Monitoring Service indicating the monitoring metrics to subscribe to. The 5GT-VS Monitoring Service relays the request to the 5GT-SO Monitoring Service. During service runtime, the 5GT-SO indicates the availability of monitoring data to the 5GT-VS Monitoring Service. The 5GT-VS Monitoring Service retrieves the data and informs the VSI LC Manager, NS(S)MF, and/or Arbitrator. It can also send this information directly to the 5GT-VS Front-end to provide it to the vertical, depending on the SLA and what was defined in the VSD.

The 5GT-VS Monitoring Service can supervise alarms related to an NSI and/or VSI only if it has subscribed to them. Thus, the NS(S)MF sends a corresponding request for subscription to NSI alarm notifications towards the 5GT-VS Monitoring Service, in order to receive the alarm notifications related to the NSI. As a result of this operation, whenever an alarm is raised, a notification will be sent to the 5GT-VS Monitoring Service, which will transmit it to the NS(S)MF.

If the 5GT-VS Monitoring Service has subscribed to performance information (if asked by the VSI LC Manager and/or NS(S)MF), then it will receive a notification when new collected performance data are available, After the notification, according to the monitoring data, it will relay it to the VSI LC Manager and/or NS(S)MF. To this end, the 5GT-VS Monitoring Service has to keep, e.g., the history of all the requests.

The NS(S)MF can request the 5GT-VS Monitoring Service to create (and delete) a threshold on a specified performance metric (for NFV-NS(s)) for which notifications will be generated when crossed. The VSI LC Manager can do the same for vertical service(s).

5.3.3.4 VSI/NSI Coordinator & LC Manager

This component coordinates the activities of the other 5GT-VS sub-components, handling the coordination between the lifecycle management of different entities which are associated to each other (e.g. a network slice instance and the vertical service instances running on top of that). To do so, it includes several subcomponents and interacts with the other modules and databases of the 5GT-VS. The subcomponents are the VSI Group Coordinator to handle the VSIs of one vertical, the VSI LC Manager to handle the lifecycle of single VSIs, and the NSMF and NSSMF to handle the lifecycle of NSIs and NSSIs.

5.3.3.4.1 VSI Group Coordinator

A vertical may define one or several resource budgets for its VSIs. These resource budgets define the maximum amount of e.g. storage, vCPU, memory, and bandwidth. The resource consumption of a specific VSI is checked against one such resource budget. This component maintains the relation between resource budgets and VSIs for each individual vertical. The actual checking and arbitration among different VSIs is done by the Arbitrator (see Section 5.3.3.2).

Moreover, the VSI Group Coordinator coordinates procedures that involves correlated actions on multiple VSIs or NSIs/NSSIs. This is the typical case of an arbitrator's decision that requires to scale-down an existing VSI to make room for an additional VSI, belonging to the same tenant and with higher priority. In this case, the VSI Group Coordinator needs to handle a workflow that impacts the lifecycle of two different VSIs, triggering initially the scaling-down of the existing one and, once this has been completed, triggering the instantiation of the new one.

5.3.3.4.2 VSI LC Manager

The VSI LC Manager proposes vertical services to customers via the catalogue of VSBs, which may include also information on the services' cost. Internally, the VSI LC Manager keeps a record of the requested VSIs and maintains their references to the associated supporting NSIs. It also performs accounting of vertical service resource consumption per VSI, interacting with the SLA & Policy Manager.

The major task of the VSI LC Manager is the coordination of the life cycle management operations, performance monitoring and fault reporting of VSIs. These operations need to be mirrored on the lifecycle management of the network slice(s) used to run the VSI. If a customer changes the requirements of a VSI, the VSI LC Manager triggers the Arbitrator to update the network slice requirements via recalculation of the NSD deployment flavour. Eventually, the VSI LC Manager asks the NSMF for a modification of the NSI capacities.

The 5GT-VS SBI is used to link the NSMF in the 5GT-VS with the 5GT-SO, e.g. for the management of the NFV-NSIs associated to specific NSIs: allocation and LCM of NFV_NSIs, performance monitoring, fault management and accounting. Thanks to the monitoring and the fault management of NFV_NSIs, which provides information to derive the status of the NSIs, the VSI LC Manager is aware of the running state of each NSI and able to verify whether it matches the expected SLAs of the corresponding VSIs. If the outcome indicates that a vertical service SLA has not been fulfilled, remediation is required.

As mentioned, the VSB catalogue includes information on SLA costs. To enable the charging of the verticals for the VSIs, the VSI LC Manager has to trace their resource consumption according to the information (e.g., monitoring data, events) reported by the 5GT-SO to the 5GT-VS Monitoring Service. Thus, the VSI LC Manager subscribes to alarm notifications, using the 5GT-VS Monitoring Service. Some service-related alarm data may be provided, if required, to the verticals. Also, the VSI LC Manager subscribes to performance measurement notifications. Again, some service related performance measurement data may be provided to the verticals according to their SLAs.

5.3.3.4.3 NSMF and NSSMF

The NSMF manages NSIs; correspondingly the NSSMF manages NSSIs. As mentioned, the VSI LC Manager relies on the NSMF and NSSMF to assess the feasibility of providing NSIs and NSSIs, respectively. As the functionalities of NSMF and NSSMF are rather similar, we describe in the following the NSMF functionality only. The NSMF and NSSMF are based on [8].

Specifically, the NSMF checks which PNFs and/or VNFs are referenced in the NSDs (the NSDs are exposed by the 5GT-SO to the 5GT-VS via the NSD catalogue), then the NSMF provides the VSI LC Manager with the NSIs, to which a VSI could be deployed. The NSMF also keeps records of all the network slice requirements (e.g. number of CPU, storage) per NSI. This information can be used by the Arbitrator to recalculate the deployment flavours of the NFV-NSs. However, the NSMF is not aware of the VSIs but only of the requirements received from the Arbitrator. Due to the monitoring and the fault management of NFV-NS instances, the NSMF is aware of the running state of the latter and able to verify the expected functionalities and whether its SLAs are met or not. In case of SLA violations the NSMF can trigger corresponding alarms.

In the 5GT-VS R2 the NSMF is enhanced to handle the relationship between an NSI and its internal NSSIs, also coordinating their LCM operations and keeping trace of their status.

5.3.3.5 SLA and Policy Management

The SLA and Policy Management allows the 5GT-VS administrator to configure SLAs and policies on a per-tenant basis, i.e. for each vertical. The SLAs managed at the 5GT-VS level includes information about the maximum amount of resources that a vertical can use for all its services, with the possibility to distinguish between cloud and edge resources. The policies are used to define administrative preferences based on the customer's profile. For example, they may indicate which resource allocation strategy should be adopted for a given tenant, or even for a given type of service, or they may put restrictions on federated domains that are acceptable to instantiate services.

During the VSI instantiation procedure, the SLA and Policy Manager is used to verify if the requested service is acceptable under the SLA currently in place and to identify the policies that must be associated to a given VSI. Such policies are then transferred to the 5GT-SO, associating them to the NFV-NSI(s) corresponding to the NSI running the VSI.

5.3.4 Vertical Slicer NBI

Two reference points are defined at the northbound of the 5GT-VS (see Figure 6):

- **Ve-Vs**, between a vertical and the 5GT-VS. This reference point provides the mechanisms to allow the vertical to retrieve VSBs, to manage VSDs, to request operational actions on VSIs, like instantiation, termination, modification, and to monitor performance and failures of instantiated vertical services.
- **Mgt-Vs**, between the OSS/BSS Management Platform and the 5GT-VS. This reference point provides primitives to manage tenants, SLAs, policies and VSBs. It is used mainly for management and administrative issues and it is handled internally within the 5G-T service provider.

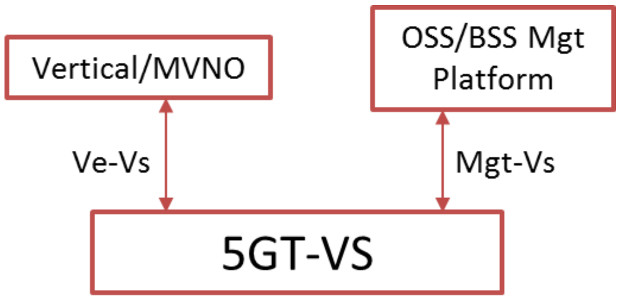


FIGURE 6: REFERENCE POINTS ON THE NORTHBOUND OF THE 5GT-VS

The 5GT-VS NBI, implementing both reference points, is a REST API based on the HTTP protocol and JSON messages, where the 5GT-VS acts as REST server and the verticals and the OSS/BSS Management Platform act as REST clients. The interface should also support asynchronous notifications from the 5GT-VS, for example based on web sockets. Suitable mechanisms for authentication and authorization of the entities issuing the requests should also be supported. However, this section specifies only the primitives and the content of the abstract messages exchanged at the reference points. Their encoding into specific protocol messages is reported in Section 5.10.1.

5.3.4.1 Ve-Vs reference point

The Ve-Vs reference point identifies the following operations:

- Query VSBs.
- Create, query, update, and delete VSDs.
- Instantiate, query, terminate, modify VSIs.
- Notifications about vertical service lifecycle events.
- Query monitoring parameters for VSIs.
- Subscriptions/notifications about vertical service monitoring parameters.
- Notifications about vertical service failures.

As an example of how these descriptions look like, see the Query VS blueprints operation in Section 5.3.4.1.1.

5.3.4.1.1 Query VS blueprints

This operation allows a vertical to retrieve one or more VSBs from the 5GT-VS catalogue. The blueprints are then used by the vertical to create the VSDs for the vertical services to be instantiated.

The Query VS blueprints messages are specified in Table 11.

TABLE 11: QUERY VS BLUEPRINTS MESSAGES

Message	Direction	Description	Parameters
Query VS blueprint request	Vertical → 5GT-VS	Request to retrieve one or more VSBs matching the given filter.	• Filter (e.g. VSB ID, …) Vertical ID.
Query VS blueprint response	5GT-VS → Vertical	Response including the details of the requested VSBs.	• List<VSB> (format specified in Section 5.4.1).

5.3.4.2 Mgt-Vs reference point

The Mgt-Vs reference point between the OSS/BSS Management Platform (Mgt in the following) and the 5GT-VS identifies the following operations:

- Create, query and delete tenants.
- Create, query, modify and delete SLAs.
- Create, query, modify and delete policies.
- Create, query and delete VSBs.

We provide a description of the operations including their parameters in an annex in Section 5.10.2.

Beyond these operations, the OSS/BSS Management Platform has also access in read mode to the information related to all the entities managed by the 5GT-VS, i.e. VSDs, VSIs, NSIs and NSSIs, which can be retrieved from the related catalogues and records.

5.3.5 Service Orchestrator NBI

The 5GT-SO NBI refers to the interface between the 5GT-VS and the 5GT-SO. It is based on the ETSI NFV IFA 013 interface (reference point *Os-Ma-nfvo*) [11]. This interface is used for: *(i)* network service lifecycle management; *(ii)* VNF package and NSD management; *(iii)* forwarding of NFV-NSI-related state information; *(iv)* policy management.

The interactions between the 5GT-VS and 5GT-SO implement four reference points (see Figure 3 in Section 5.3.2). The 5GT-SO NBI implements additional methods to allow the 5GT-SO to handle NFV-NSs extended to include also MEC Applications. In terms of interfaces, this has an impact mostly on the management of the catalogues and on the queries of the NFV-NS instances, where both of them should support the information related to the MEC applications:

- **Vs-So (-LCM)** is used for operation on NFV-NSs. It offers primitives to select, allocate, terminate, query, and reconfigure network services or receive notification about their lifecycle. In addition, this reference point should provide pricing primitive for the provided services. This reference point is implemented following the information model defined in the IFA 013 NS Lifecycle Management Interface [11]. It provides messages for creation and deletion of network service identifiers as well as instantiation, scaling, update, querying, and termination of a network service instance;
- **Vs-So (-MON)** is used for the monitoring of NFV-NSs and VNFs through queries or subscriptions and notifications about performance metrics. Additionally, this interface provides APIs for the fault management. This reference point is implemented following the information model defined in the IFA 013 NS Performance Management Interface [11] and in the IFA 013 NS Fault Management Interface [11]. The Performance Management Interface provides messages for creating, deleting, and querying Performance Monitoring jobs (PM jobs) as well as subscriptions to receive monitoring reports and notifications. The Fault Management Interface provides subscriptions, notifications, and queries for alarms related to network services and VNFs;
- **Vs-So (-CAT)** is used for the management of NFV NSD and VNF package descriptors, including the on-boarding, removal, updates and queries. This reference point is implemented following the information model defined in the IFA 013 NSD Management Interface [11], in the IFA 013 VNF Package Management Interface [11], and in the ETSI MEC 10-2 Application Package Management Interface [23], where the 5GT-SO acts as the Mobile Edge Orchestrator (MEO). Each of them provides mechanisms to on-board, enable, disable, update, delete and query NSDs (IFA 13 [11]), PNFDs (IFA 13 [11]), VNF Packages (IFA 13 [11]), and MEC Application Packages (MEC 10-2 [23]), as well as subscriptions and notifications to notify new on-boarding actions or changes in existing descriptors;
- **Vs-So (-POL)** is used for the management of policies. It offers primitives to transfer, delete, activate, deactivate, associate, disassociate policies and receive notifications about policy conflicts. This reference point is implemented through the IFA 013 policy management interface [11]. It provides messages to transfer, delate, activate, deactivate, associate, disassociate polices, and to notify about policy conflicts.

In ETSI NSD (IFA 014 [28]), some information elements are missing, thus it needs to be extended or be modified to fully support the services provided by the 5G-T project. They are: (1) AppD, which follows the information model defined in ETSI MEC 10-2 [23], and (2) latency and location constraints. NSD and VNF packages follow the information models defined in the IFA 014 [28] and IFA 011 [29], respectively. In 5G-T, the NSD is extended to include references to the AppDs, as shown in Figure 7 and Figure 8, as well as appD profiles to define how the MEC application are interconnected to the virtual links in the network service. See also Sections 5.4.3.1 and 5.4.3.2 for details on how this is integrated in NSDs.

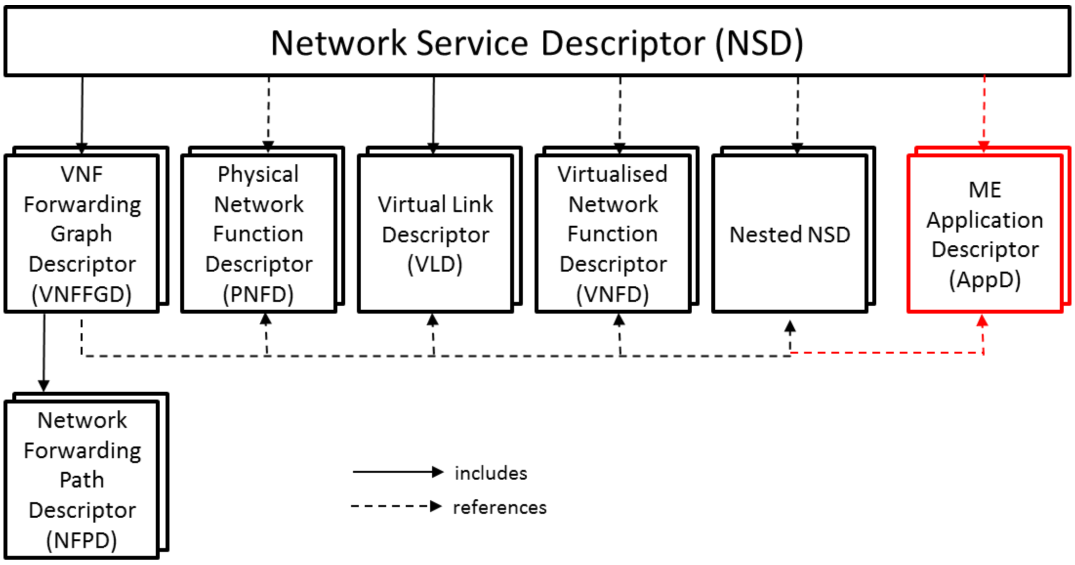

Network Service Descriptor (NSD)
VNF Forwarding Graph Descriptor (VNFFGD)
Physical Network Function Descriptor (PNFD)
Virtual Link Descriptor (VLD)
Virtualised Network Function Descriptor (VNFD)
Nested NSD
ME Application Descriptor (AppD)
Network Forwarding Path Descriptor (NFPD)
includes
references

FIGURE 7: NSD EXTENDED WITH APPD REFERENCES

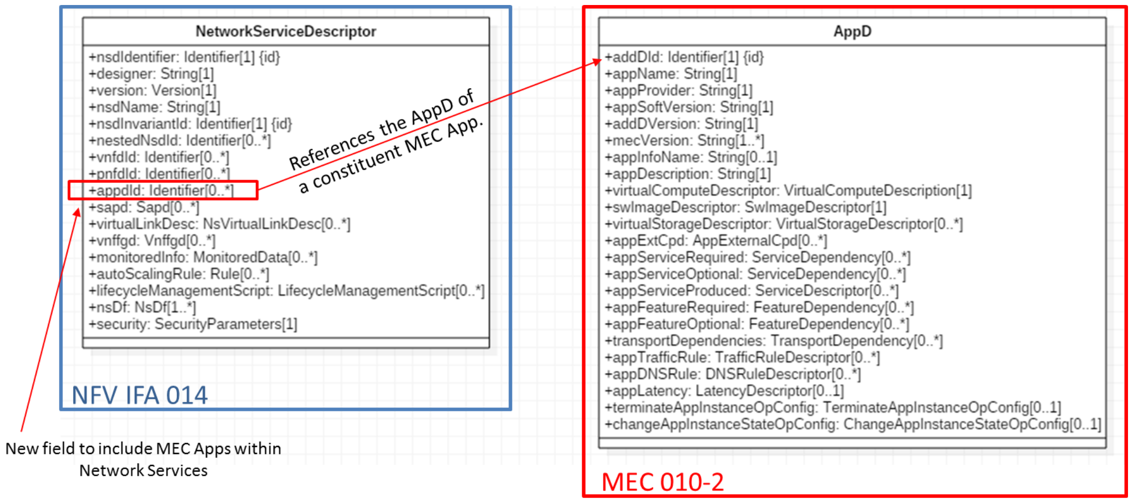

NetworkServiceDescriptor
+nsdIdentifier: Identifier[1] {id}
+designer: String[1]
+version: Version[1]
+nsdName: String[1]
+nsdInvariantId: Identifier[1] {id}
+nestedNsdId: Identifier[0..*]
+vnfdId: Identifier[0..*]
+pnfdId: Identifier[0..*]
+appdId: Identifier[0..*]
+sapd: Sapd[0..*]
+virtualLinkDesc: NsVirtualLinkDesc[0..*]
+vnffgd: Vnffgd[0..*]
+monitoredInfo: MonitoredData[0..*]
+autoScalingRule: Rule[0..*]
+lifecycleManagementScript: LifecycleManagementScript[0..*]
+nsDf: NsDf[1..*]
+security: SecurityParameters[1]
NFV IFA 014
References the AppD of a constituent MEC App.
New field to include MEC Apps within Network Services
AppD
+addDId: Identifier[1] {id}
+appName: String[1]
+appProvider: String[1]
+appSoftVersion: String[1]
+addDVersion: String[1]
+mecVersion: String[1..*]
+appInfoName: String[0..1]
+appDescription: String[1]
+virtualComputeDescriptor: VirtualComputeDescription[1]
+swImageDescriptor: SwImageDescriptor[1]
+virtualStorageDescriptor: VirtualStorageDescriptor[0..*]
+appExtCpd: AppExternalCpd[0..*]
+appServiceRequired: ServiceDependency[0..*]
+appServiceOptional: ServiceDependency[0..*]
+appServiceProduced: ServiceDescriptor[0..*]
+appFeatureRequired: FeatureDependency[0..*]
+appFeatureOptional: FeatureDependency[0..*]
+transportDependencies: TransportDependency[0..*]
+appTrafficRule: TrafficRuleDescriptor[0..*]
+appDNSRule: DNSRuleDescriptor[0..*]
+appLatency: LatencyDescriptor[0..1]
+terminateAppInstanceOpConfig: TerminateAppInstanceOpConfig[0..1]
+changeAppInstanceStateOpConfig: ChangeAppInstanceStateOpConfig[0..1]
MEC 010-2

FIGURE 8: NSD AND APPD INFORMATION MODELS

Furthermore, ETSI NSD allows the integration of existing PNFs, i.e. dedicated machines deployed for implementing some specific network function placed in a certain location, within an NSD. However, the ETSI NSD does not contain any information element to specify this location. Thus, the 5G-T proposes corresponding extensions of information elements within NSDs to express location as well as latency constraints (see Section 5.4.3.4.2).

The most relevant operations for policy management are the transfer and associate operations. The transfer operation conveys a policy from the 5GT-VS to the 5GT-SO and, if successful, it receives a reference to this policy. Remember, there is no standardized notation for policies, and 5GT-VS and 5GT-SO need to get a common understanding of the policies by different means. The associate policy links a policy with a specific NSI, this allows to associate different policies with different instances of even the same NSD. These operations are summarized in Table 12.

TABLE 12: RELEVANT POLICY MANAGEMENT OPERATIONS

Message	Direction	Description	Parameters
TransferPolicyRequest	5GT-VS → 5GT-SO	Transfer policy and retrieve reference	• designer • name • version • policy
AssociatePolicyRequest	5GT-VS → 5GT-SO	Associate policy with NSIs.	• policyInfoId • List<nsInstanceId>.

5.3.6 NFVIaaS and NSaaS Support

The 5GT-VS as defined before focused on the support for verticals to define and deploy vertical services, i.e. on a Service as a Service (SaaS) basis. We investigate two approaches to provide the support for the various levels of NFVIaaS and for NSaaS as defined below. In the first approach, the support would be provided more directly, whereas in the second the support is provided indirectly by considering NFVIaaS and NSaaS as specific vertical services themselves.

Network slice as a service (NSaaS): Unlike vertical services, an NSaaS is to provide a network along with the services that it may support. For instance, a network provider may provide a mIoT network slice as a service, which may support several services, including sensor monitoring, EVS as well as traffic management and warehouse automation. The NSaaS customer is aware of the NSI; it can, in turn, play the role of a provider itself, and offer to its own consumers its vertical services built on top of the services of the network slice (B2B2X). Regarding the mIoT NSaaS example, the consumer may fragment its mIoT NSI capabilities per criticality level and per user-defined area (such as factory premises, cities, and venues) in vertical services offered to its subscribers. The NSaaS consumer may have to negotiate with the NSaaS provider to limit the level of exposure of:

- NSI characteristics allowing the NSaaS customer building its own services, for instance, security functions are reachable according to a prior agreement,
- NSI management features, for instance, only commissioning/decommissioning operations on an NSI are allowed.

The NSaaS concept is present in the MVNO UC. The 5GT-VS MVNO implements the LCM of Network Slices. Thus, Network Slices can be created, instantiated, terminated and deleted from the 5GT-VS MVNO and the latter passes the requests to the 5GT-SO.

NFVI as a Service (NFVIaaS): this business type allows customers setting up their own applications and network functions over an infrastructure provider. For instance, an MVNO may bring its own Home Subscriber Server (HSS) that needs to be connected with the other vEPC functions provided by the Mobile Network Operator (MNO). This business type can be implemented as follows: The 5GT-VS provides a vEPC blueprint, wherein a field describing the HSS is filled up by the customer where the virtual machine image for the HSS is provided as a parameter. A MVNO can create its specific vEPC by completing this blueprint with the location of the virtual machine image for its HSS.

The direct approach allows specific customers, e.g. an MVNO, to request directly network infrastructure or an NSI and operate it itself. The direct approach was used in the NSaaS example, with explicit operations for NSI management operations at the 5GT-VS NBI.

The NFVIaaS example uses the indirect approach, the infrastructure is again described as a vertical service. The blueprint has a placeholder to provide the HSS, otherwise, this is a vertical service as, e.g., one for sensor monitoring.

5.4 Service Descriptions

This section focuses on the descriptors used in the 5GT-VS as such. We focus on the details of the information that shall be conveyed in each type of descriptor and the notations used. Shortened examples of descriptions are provided in Section 5.7 and Section 5.12. To facilitate the reading of this section, in Figure 9, we show the mapping between the interfaces and the descriptors used on top of the components already shown in Figure 2.

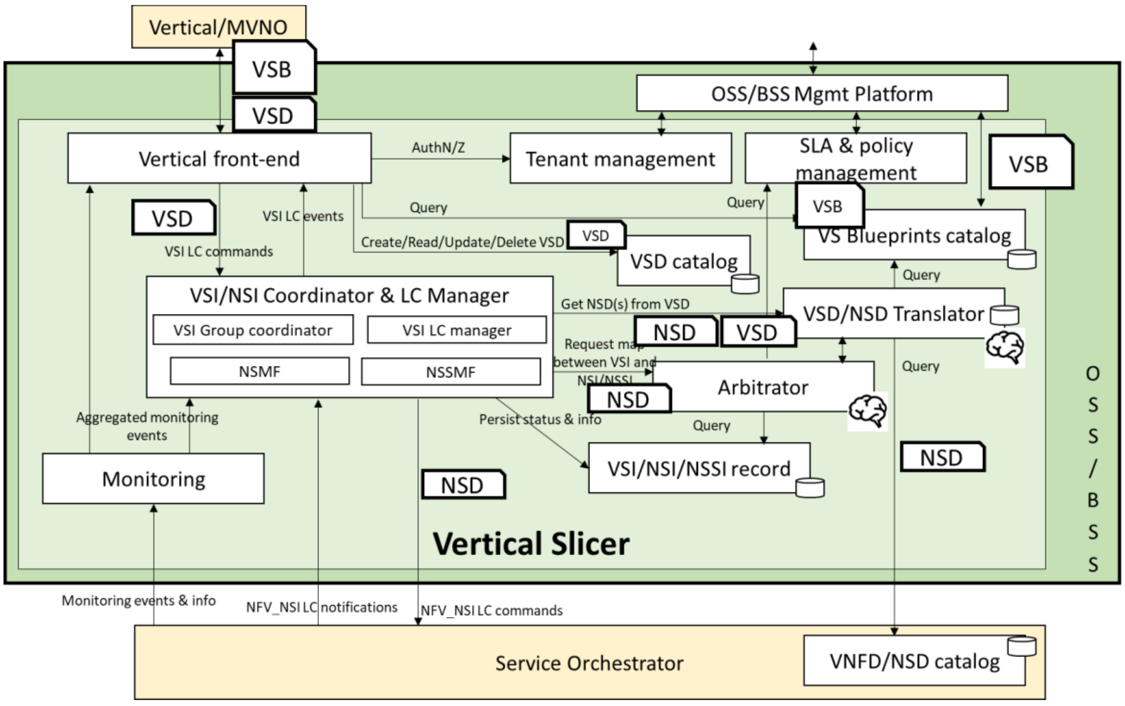


FIGURE 9: USAGE OF DESCRIPTORS IN 5GT-VS

The Vertical Service Blueprints (VSB) and Descriptors (VSD), which are main contributions of the project, enable modeling vertical services using parameters adapted to the knowledge and capabilities of the vertical (i.e. model services without detailing the infrastructure resources required). We use an extended version of ETSI NSDs [28] as a notation for network slice templates.

To instantiate a vertical service, a vertical first selects a blueprint, then it provides the missing information in the blueprint to prepare a VSD. The 5GT-VS maps this to a Network Service Descriptor (NSD), describing the network slice for this vertical service. In this section, we start with presenting VSBs in Section 5.4.1, focusing on those fields that are different from those in the VSDs. We consider VSBs as parameterized versions of VSDs, which we describe in Section 5.4.2. We describe NSDs and the extensions we propose in Section 5.4.3.

5.4.1 Vertical Service Blueprints

The 5GT-VS provides a list of VSBs to a vertical, designed to capture the service requirements specified by the vertical. Essentially, a VSB is a parameterized VSD, i.e. some aspects of a VSD are left open to be specified by the vertical.

A VSB, see Table 13, has fields to identify it and to describe the structure of the corresponding vertical service. In addition, it has fields to define SLA and other requirements. These fields are the same as those for VSDs and are shown in Table 14.

TABLE 13: VERTICAL SERVICE BLUEPRINT

Field	Description
Name	Name provided by the provider for the blueprint.
Description	Short description of the blueprint, e.g. “Sensor monitoring over 5G”.
Version	A version number.
Identity	Unique identifier for a blueprint. This is provided by the 5GT-VS when on-boarding the blueprint.
Parameters	List of parameters. The list provides for each parameter its name, type, description, and the field where it is used.
Atomic functional components involved	List of atomic functional components (i.e., network functions and virtual applications in general) needed to implement the VSB requested, including dimensioning of those functions, in terms of number of users or sessions, bandwidth, parameters to be monitored, etc. The identified functional components can include functions to support the service from the provider perspective (e.g., firewall, load balancer, etc) transparent to the vertical. Specific information for the atomic components is described.
Service sequence	Description of how the atomic functional components should interact. This can be provided as a VNF forwarding graph [38]. It is also possible to refer to the service sequence of another VSD, thereby allowing to create nested VSDs.
Connectivity service	Specification of the kind of connectivity service to be provided to the vertical (e.g., L2VPN, Ethernet-VPN (EVPN), L3VPN, etc.), and properties (QoS and/or flow management, protection, restoration, etc.) that can be necessary for the service instantiation and orchestration.
External interconnection	An indication of specific external connections to be provided (i.e., Internet, Network A, etc.). This includes the specification of connection endpoints, and it may include the indication of specific identifiers of the external networks to connect (e.g., IP addresses, Autonomous Systems, etc.). Corresponding identifiers - if needed

	- can be retrieved from the connection endpoints after vertical service instantiation.
Internal interconnection	An indication of specific internal connections to be provided (i.e., connections to other vertical service instances or slices, from the same vertical customer or not).

It is left to the implementation of the 5GT-VS Frontend 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, types of traffic probes, etc., could be left open in the blueprint. It is up to the 5GT-VS Frontend 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.

Examples of VSBs are defined in Section 5.12.

5.4.2 Vertical Service Descriptors

The VSDs are the descriptors obtained after providing the missing information in VSBs. VSDs drive the 5G-T service customer (TSC) (i.e. a vertical or an M(V)NO) to express its service request. A VSD has a number of fields that identifies it, and defines the SLA requirements and other properties of the vertical service. The structural information on the vertical service is derived from the corresponding VSB.

The following table presents the structure of a generic VSD.

TABLE 14: VERTICAL SERVICE DESCRIPTOR

Field	Description
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.
Blueprint	The identifier of the blueprint from which this VSD was derived.
Identity	Unique identifier for a VSD. This is provided by the 5GT-VS when on-boarding the VSD.
SST	For many vertical services it is possible to capture the SLA requirements by a predefined slice/service type (SST)[15], see Table 2 and Table 3. More complex, especially nested, vertical services might require SLA requirements in separate and more detailed statements, see the SLA field.
Service constraints	List of service constraints to be taken into consideration for the service orchestration and instantiation. The needed constraints

[15] It is possible to extend the single-value SSTs by the more fine-granular categorization as proposed by 3GPP.

	are based on the business requirements listed in Section 3. These constraints can be: • Geographical constraints on the deployment area and on the location interconnections. • Security, i.e. the level of network slice isolation (ReqVS.B21). • Priority in case of resource shortage, see Section 5.4 on arbitration. • Cost. • Synchronization. • Preferred/non-preferred, prohibited infrastructure providers (ReqVS.B19). • Lifecycle independence (ReqVS.B20). Some of these constraints could influence decisions with respect the usage of capabilities across the federation of SO domains.
Management and control capabilities for the tenant	Different options can be considered for control and management by the tenant of the slice to be provided (i.e., the vertical customer). Following [31], either provider managed or tenant managed slices are supported.
SLA	As an alternative to specifying a single SST, service level objectives, as agreed with the vertical customer can be provided. Any of the technical requirements as defined in [4], Section 4, can be defined. E2e latency refers to the latency among the connection endpoints in the service sequence and/or interconnections and may include also processing latency at the VNF/VF. An additional SLA requirement is, e.g., the required bandwidth of external and internal interconnections.
Monitoring	• List of network Key Performance Indicators (KPIs), e.g. throughput at Connection Points (CPs) or Service Access Points (SAPs), load of instantiated VNFs, etc. • Additional traffic probes to be instantiated, e.g. traffic mirroring at a defined CP.
Lifetime	Information about aging, i.e. lifetime and/or start/end dates required for the slice to realize the service request (short-lived slices vs long-lives slices).
Charging	Information about the charging aspects [32] to be taken into consideration.

The atomic functional components, the service sequence, and the external and internal interconnections can be described in the notation of an ETSI NFV NSD [28].The atomic functional components are considered as VNFs and VAs, information similar to VNF packages can be described for them, see Table 15.

TABLE 15: INFORMATION ON ATOMIC FUNCTIONAL COMPONENTS

Field	Description
Number of Application servers	The vertical may specify the number of application servers required for the service. If left open, this number is provided by the VSD/NSD Translator, see Section 5.3.3.1.
Images of virtual applications	The vertical specifies the path to the images of the virtual applications.
Virtual application connection end points	The vertical specifies the connection end points for the application.
Lifecycle operations	Scripts for instantiating, terminating, updating, healing, where applicable.
Scaling rules	Rules when to scale in/out the virtual application

Examples of VSDs are presented in detail in Section 5.12.

5.4.3 Network Service Descriptors

In this section, we describe the information elements that must be present in the Network Service Descriptors (NSDs) to properly define the NFV-NS behavior. The skeleton of the list of information elements presented below is tightly related to ETSI NFV IFA 014 [28], and it tries to gather those fields necessary for the NSs that are deployed in the context of the 5G-T. Here we describe the information elements of NSDs that are extended by 5G-T. For other information elements, we list their most relevant fields in an annex in Section 5.11.

5.4.3.1 Network Service Descriptor (NSD)

The Network Service Descriptor (NSD) entity is the root entity that references other information elements in the description of a network service. The entity allows the complete description of an NFV-NS that is deployed.

TABLE 16: NSD INFORMATION ELEMENT

Field	Description
NSD_id	Unique identifier for an NSD descriptor.
nested_NSDs	List of nested NSD_ids.
VNFD_ids	List of VNFD ids.
AppD_ids	List of AppD ids, see Section 5.4.3.2.
PNFD_ids	List of PNFD_ids.
SAPD_ids	List of SAPD_ids.
VNFFG	ID of the VNFFG.
flavours	List of NSDF ids.
signature	Authenticity signature to protect the descriptor.

5.4.3.2 MEC application Descriptor (AppD)

In 5G-T, an NFV-NS may include MEC applications, which are described using the Application Descriptor (AppD) as specified by ETSI MEC [23]. The AppD is very similar to the VNFD, however it includes specific information requirements for MEC applications. The following table summarizes the fields which indicates MEC applications requirement in terms of traffic redirection and requested latency.

TABLE 17: APPD INFORMATION ELEMENT

Field	Description
appServiceRequired	ServiceDependency: Describes the services a multi-access edge application requires to run.
appTrafficRule	List of TrafficRuleDescriptor: Describes the traffic rules the multi-access edge application requires.
appDNSRule	Describes the DNS rules the multi-access edge application requires.
appLatency	LatencyDescriptor: Describes the maximum latency tolerated by the multi-access edge application.

5.4.3.3 Network Service Descriptor (NSD) deployment flavour and instantiation level information elements

The NSD and, in particular, its Deployment Flavour element are extended with an additional field “mecAppProfile” that defines, for each MEC application included in the network service, how it is interconnected to the virtual links. In particular, a mecAppProfile information element has the format defined in Table 18.

TABLE 18: APPPROFILE INFORMATION ELEMENT

Field	Description
appProfileId	ID of the MEC application profile.
appdId	ID of the descriptor of the MEC application referred by the profile. This ID must be included in the list of AppD_ids defined in the root of the NSD.
appVirtualLinkConnectivity	This field defines how the MEC application is attached to the virtual links of the NS. In particular, this fields includes a “virtualLinkProfileId” and a list of “cpdId”. The virtualLinkProfileId identifies a particular virtual link within the network service, while the cpdId corresponds to the ID of the MEC application connection points (as defined in the AppD) that are attached to that virtual link.

Moreover, in order to define the number of MEC application instances that must be created for a given instantiation level, the nsInstantiationLevel information element within the Deployment Flavour is extended with an additional field “appToLevelMapping”, as described in Table 19.

TABLE 19: APPTOLEVELMAPPING INFORMATION ELEMENT

Field	Description
appProfileId	ID of the MEC application profile. It must correspond to one of the MEC application profiles defined in the deployment flavour attribute.
numberOfInstances	Number of instances that must be created for the MEC application associated to the given application profile.

5.4.3.4 ETSI-NSD Gaps

In this subsection we present gaps in ETSI NFV NSDs and propose solutions developed within 5G-T, as done for the case of MEC AppDs, see Sections 5.4.3.1 and 5.4.3.2.

5.4.3.4.1 Expressing location constraints

Although we expect that a vertical is not interested where exactly its VNFs or VAs are deployed, it still wants to express certain location constraints on a VSI. E.g. the vertical might want to express that the SAP to its VSI from the public internet or from its own company network is not too far away. As a second example, the vertical might require a certain coverage at the air interface of its VSI. In the definition of VSBs and VSDs, we have defined already some fields to specify such requirements, see Table 13 and Table 14.

ETSI NFV allows the definition of location information for PNFs and VNFs: ETSI NFV IFA014 [28] allows the definition of geographicalLocationInfo for PNFs as part of the PNFD. ETSI NFV IFA013 [11] allows the definition of VNFLocationConstraints for VNFs as part of the NsInfo information element. This allows to provide information at instantiation time of a network service. Specifying the location of VNFs and PNFs does not solve the problem, as in the case of nested service definitions, a vertical might not even be aware which VNFs and PNFs are contained in the network service. As a second problem, the location information of PNFs is described in the PNFD. This implies that for each new service instance with a different location, a new service descriptor has to be created.

Therefore, we propose to extend the definition of the SapData information element of ETSI NFV IFA 013 [11] with a field to express location constraints. The SapData information element can be provided also at instantiation time of a network service instance, therefore it is not necessary to create separate NSDs for almost similar network service instances.

TABLE 20: LOCATIONINFO ATTRIBUTE

Attribute	Qualifier	Cardinality	Content	Description
locationInfo	O	1	LocationInfo	Location of SAP, can be a point or an area. **Note**: type not specified further, common formats of geographical information systems would be the most convenient ones.

Also, as the SAPs are defined for the border of network services, a vertical can express its constraints on where to connect to the service without having to know the internal details of the service. For the air interface, we consider defining an SAP representing the air interface. The required air interface coverage can be expressed as location constraint of this SAP. Note that a location constraint can be either a single coordinate or a set of coordinates specifying one or several regions.

5.4.3.4.2 Expressing latency constraints on paths

So far, latency constraints among external or internal connection points of vertical services cannot be expressed adequately within NSDs. ETSI NFV IFA014 [28] allows the definition of QoS attributes such as latency or packet loss ratio for deployment flavours of individual VLs. Actually, it is necessary to express latency constraints across a path within the network service, not just a single VL.

Therefore, we propose to extend the descriptor of a Network Forwarding Path (NFP) with a similar QoS attribute as for virtual links.

TABLE 21: ADDITIONAL QOS ATTRIBUTE FOR NFPD

Attribute	Qualifier	Cardinality	Content	Description
qos	M	0..1	QoS	See ETSI NFV IFA014, Section 6.5.6

We assume that the QoS constraints are the same for different instances of the same vertical service. Therefore, we propose to provide this information as part of the NSD, not as part of the instantiation parameters of network service instances. These QoS constraints will be used by the 5GT-SO in the placement decisions of VNFs. Eventually, VNFs and VAs should be placed such that these QoS constraints are satisfied. It is left to the 5GT-SO to break down the latency along a path into latencies of VLs taking different deployment options into account. The 5GT-VS Monitoring Service can request performance monitoring data from the 5GT-SO to supervise whether these QoS constraints are satisfied.

5.4.4 Vertical-Driven Service Composition

In this section we describe an approach for composing services by verticals, which we name *vertical-driven service composition*. We aim to increase the flexibility for the verticals, without increasing complexity for them. Additionally, the approach has to fit into the existing 5G-T system, i.e. the 5GT-VS should handle the service composition and map this into vertical service and network service descriptors and onboard them as usual to the 5GT-SO. For the 5GT-SO and 5GT-MTP no changes should be needed.

Variations of a vertical service - consisting of a communication service and vertical applications - may occur within different parts of the vertical service:

- **Applications**: There may be variations among the applications themselves. E.g. there could be different kinds of content distribution systems for media applications or the data of remotely monitored patients could be evaluated by different algorithms. Additionally, a vertical might want to add its own VNFs and use them as building blocks in the definition of services.
- **Control-plane**: Different optimization algorithms could be used for the control plane of a communication service. E.g. handovers could be triggered simply

based on degrading radio conditions, but they could be triggered as well based on estimated trajectories of users or targeting a more balanced use of cells. Such variations could be more relevant for M(V)NOs.

- **Management-plane**: When providing an access network for end-users some kind of subscriber management is needed. This could range from a full-fledged application including customer relationship management for an M(V)NO with millions of end users to a simple web-based or spread-sheet based interface maintaining a few hundred end users for a mining or construction site. Different levels of control and monitoring of the communication part of a vertical service could be made available to verticals as variations of the same service.
- **General functionality**: There are also variations in general functions such as VPN gateways, firewall, intrusion detection systems, etc.

These variations are independent of each other, resulting in combinatorial explosion in the number of variations, which in turn increases the maintenance effort for the service provider.

We propose an approach, where a communication service is used as the backbone of a vertical service and which can be extended at different places with building blocks of different kinds. As an example, see the diagram in Figure 10. A 5G network is the communication service within this vertical service.

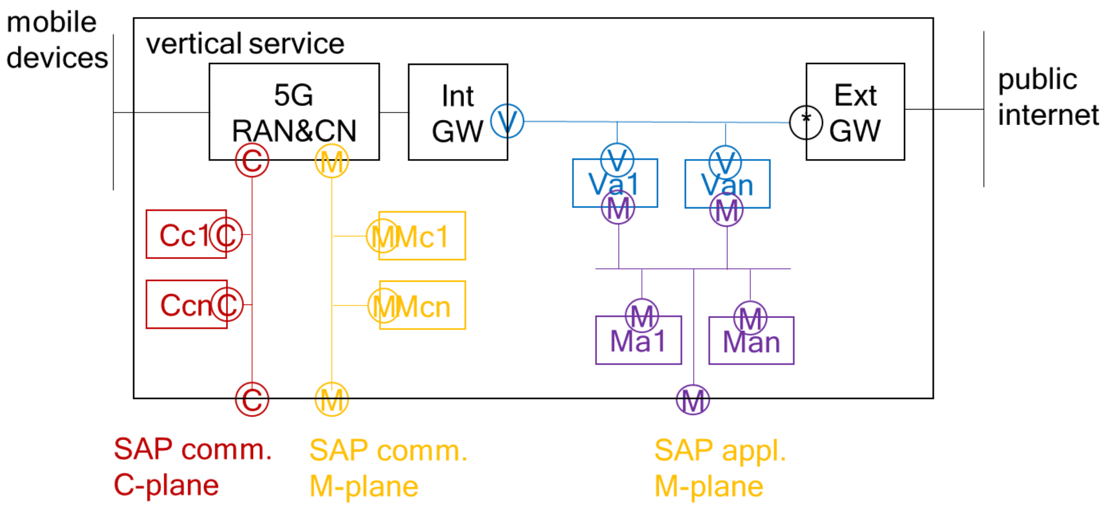


FIGURE 10: VERTICAL-DRIVEN SERVICE COMPOSITION GENERAL APPROACH

Different vertical applications - Va1 .. Van - are used and are complemented by functions to manage them. An internal and external gateway are general purpose functions. Control plane - Cc1 .. Ccn - and management plane functions - Mc1.. Mcn - for the 5G network could be used as well. Note that these different types of functions are connected to different virtual links, thereby establishing separate domains.

Subsequently, we present two incarnations of this general approach, one focusing on the vertical applications and another one considering the other type of functions as well.

5.4.4.1 Flow-based Programming of Applications

For some services, the vertical applications form a service function chain and the functions can be seen as passing data along this chain. Essentially, this is the approach taken by the service development kit (SDK) of 5G-City[16] [36] based on flow-based programming (FBP) [39]. Each function or building block has ports of two types - IN or OUT - which can be connected to the ports of other functions. When more than two building blocks are connected, then the virtual link is a tree-like structure, with an IN-port of one function as root and OUT-ports of several other functions as leaves.

An example with two functions Va1 and Va2 is shown in Figure 11. Whether ports are IN or OUT has been marked with arrows in this diagram.

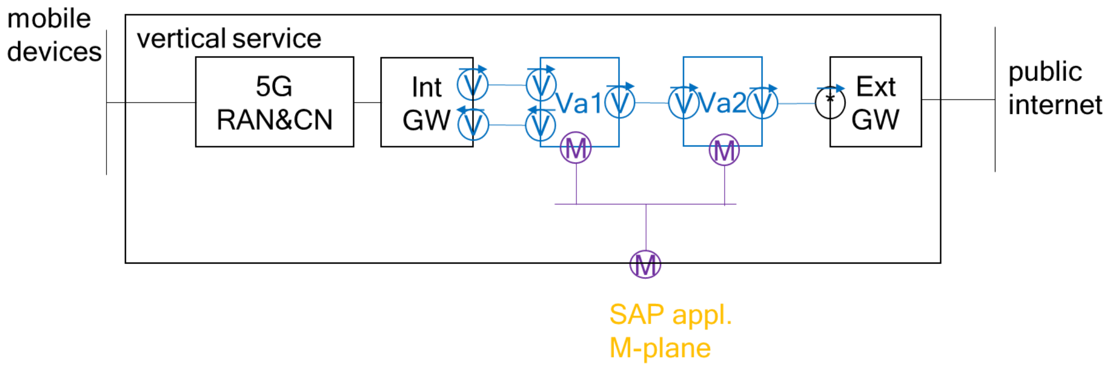


FIGURE 11: VERTICAL DRIVEN SERVICE COMPOSITION, FLOW-BASED

The internal and external gateways adapt the IP-based communication in the RAN or in the public internet to the FBP paradigm. Slightly extending the concept of the 5G-PPP project 5G-City, we have added management access to the functions, simply by providing a management connection point of each function to a corresponding SAP of the vertical service. The actual management functionality is considered outside of the vertical service here, nevertheless there should be configuration and monitoring possibilities.

On a technical level, each function corresponds to a VNF and it may even consist of several VNF components. Especially, such a function may consist of one component acting as load balancer and other ones as workers. Autoscaling rules based on CPU load can be defined for such functions as well.

A vertical may select these smaller functions from a separate catalogue of building blocks and combine their IN and OUT ports accordingly in an appropriate editor [36]. The network service descriptor including the communication service, management connectivity, and internal and external gateways can be created automatically.

A vertical may also express SLA requirements regarding bandwidth at connection points and latency and reliability on paths consisting of several connection points. For the sake of simplicity, SLA requirements regarding coverage, energy-efficiency, cost, etc., can be expressed on the level of the whole service, but not on the level of the building blocks.

5.4.4.2 Service mesh based building blocks

Extensions of the C-plane and M-plane of a communication service are typically not written in an FBP style, as used in Section 5.4.4.1 for the vertical applications. Therefore, we propose a more generic concept for vertical-driven service composition here. Note, this more generic concept is not limited to the C- and M-plane of communication services, it can also be used for the vertical applications themselves. As a consequence, two

[16] https://www.5gcity.eu/

approaches for vertical-driven service composition are available for the vertical applications.

This more generic concept is inspired by the concept of service meshes defined by the 5G-PPP project Matilda[17] [40]. According to this concept, each component of a vertical application is accompanied with a side-car component. Such a side-car component handles the difficulties of deploying a component in a heterogeneous and distributed environment. E.g. the side-car can extend each request to another component with a timer to make each such request synchronous. More importantly, the side-car can provide the functionalities to include the component into the service-mesh by registering produced services to a service registry and correspondingly retrieving the information from the service registry how to communicate with requested services. Differing from Matilda, where types of exposed and required interfaces are defined, we use here a service registry as defined by MEC [19]. It can be checked whether all services required in such a service mesh are actually produced by some component.

The application component together with example functions of a side-car are shown in Figure 12. Here three different functions are shown, an access control to limit access to the application, which can also be used to establish a sandbox, protecting other applications from a misbehaving one. A service registry proxy is a second function. A load balancer is an optional function, needed in case there are multiple instances of the application.

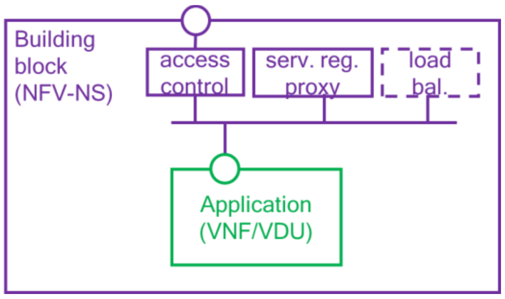


FIGURE 12: COMPONENT WITH SIDE-CAR

Defining the components as elements in a service mesh has the advantage that a simple topology can be chosen. Each of the components is connected to a multi-point to multi-point virtual link, in NFV terminology [29] this is a virtual link of type mesh. There are several such links, one for each type of component, to ensure isolation among the different types of components. Similarly, the service registry has to provide different namespaces for each of these virtual links.

The diagram in Figure 10 shows the different types of virtual links and different types of components attached to each of them. The diagram also shows that the different virtual links are kept separate. Note that this diagram shows all possibilities to extend the communication service, but a vertical is not forced to make use of all the possibilities. As the topology is simple, a vertical may just select different building blocks, this keeps the effort of defining the vertical service small.

5.4.4.3 Relation to the 5G-T architecture

In the previous sections, we extended the definition of vertical services by a pure selection from a catalogue to a composition of building blocks by the vertical itself. The

[17] http://www.matilda-5g.eu/

building blocks can again be defined by the service provider or they can even be defined and implemented by the vertical. They can be defined as a VNF for the FBP approach, or as an NFV NS for the service-mesh based approach.

The component in Figure 12 is defined as an NFV NS, where the actual application and the side-car functions are defined as VNFs. The communication service acting as a backbone for the vertical service is defined itself as an NFV NS. This allows the construction of both a VSD and a composite NSD for a vertical service. This NSD can be onboarded to the 5GT-SO.

Once the VSD and NSD have been constructed, they can be treated within the 5GT-VS in the same way as a VSD and NSD derived from a blueprint. If the building blocks have parameters, the Translator can map them to different DFs and ILs, the Arbitrator can do resource arbitration in case of conflicts among several VSIs, and the VSI/NSI Coordinator can handle the lifecycle of the vertical service once its instantiation has been triggered.

5.5 5GT-VS Workflows

As described in Section 5.3.1, the 5GT-VS consists of several components that interact locally and with external entities of the 5G-T architecture: The Vertical, OSS/BSS, and the 5GT-SO. In this section, we will describe via workflows the 5GT-VS component roles to ensure vertical service instantiation and management.

5.5.1 Blueprint On-boarding

Description: The workflows describes how the 5G-T service provider (TSP) onboards a VSB.

Prerequisite: none.

Assumptions: none.

Workflow: VSBs of vertical services are onboarded by the TSP through its OSS. As described in the workflow in Figure 13, the TSP provides in the request to onboard the VSB the description of the VSB, descriptions of referenced vertical and network services, and packages of the referenced VNFs (01).

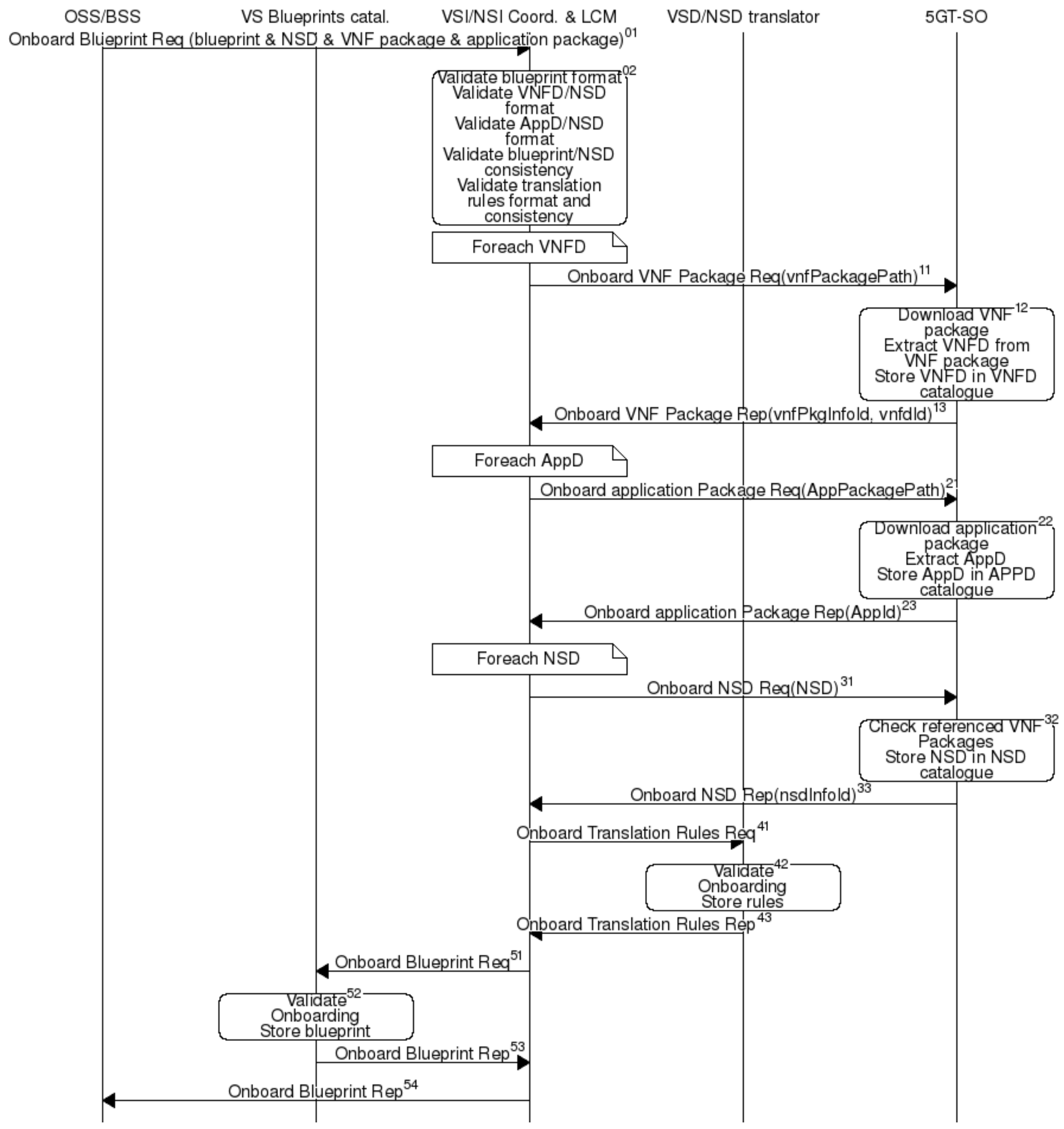

OSS/BSS
VS Blueprints catal.
VSI/NSI Coord. & LCM
VSD/NSD translator
5GT-SO
Onboard Blueprint Req (blueprint & NSD & VNF package & application package) 01
Validate blueprint format Validate VNFD/NSD format Validate AppD/NSD format Validate blueprint/NSD consistency Validate translation rules format and consistency 02
Foreach VNFD
Onboard VNF Package Req(vnfPackagePath) 11
Download VNF package Extract VNFD from VNF package Store VNFD in VNFD catalogue 12
Onboard VNF Package Rep(vnfPkgInfoId, vnfdId) 13
Foreach AppD
Onboard application Package Req(AppPackagePath) 21
Download application package Extract AppD Store AppD in APPD catalogue 22
Onboard application Package Rep(AppId) 23
Foreach NSD
Onboard NSD Req(NSD) 31
Check referenced VNF Packages Store NSD in NSD catalogue 32
Onboard NSD Rep(nsdInfoId) 33
Onboard Translation Rules Req 41
Validate Onboarding Store rules 42
Onboard Translation Rules Rep 43
Onboard Blueprint Req 51
Validate Onboarding Store blueprint 52
Onboard Blueprint Rep 53
Onboard Blueprint Rep 54

FIGURE 13: ON-BOARDING BLUEPRINT WORKFLOW

The provided input is validated by the VSI/NSI Coordinator & LCM (02). If the validation is successful, the referenced VNF packages, application packages and descriptions of network services are onboarded to the 5GT-SO. VNF packages and NSDs which have been onboarded before to the 5GT-SO do not have to be onboarded again (11 - 13, 21 - 23, 31 - 33).

The translation rules from VSDs based on this blueprint to the corresponding NSDs are validated and stored in the Translator component. On-boarding the descriptors to the 5GT-SO and the translation rules to the Translator could also be done concurrently, there is no need to keep a strict order (41 - 43).

Once all the constituent parts of the blueprint have been onboarded to the respective components, the blueprint itself is validated and stored in the blueprint catalogue (51 - 53). From this point onwards, the blueprint can be used by verticals to prepare vertical services, see Section 5.5.2.

In case a MEC application is referenced in the blueprint, the AppD/NSD validation ensures that a MEC application could be run, by checking that the necessary related VNFD or AppD or PNFD are present in the NSD. Indeed, 5G-T considers three scenarios to deploy a MEC application: (i) a MEC application requiring traffic redirection and consuming MEC services; (ii) a MEC application requiring only MEC services; (iii) MEC application requiring only traffic redirection. For the first scenario, the MEC application requires that also an EPC (MME, HSS, SPGW-C), SPGW-U, eNodeB(s), and Multi-access Edge Platform (MEP) are deployed. Specifically, the SPGW-U and the MEP should be run at the edge with the MEC application. For the second scenario, the MEC application needs eNodeB(s) and MEP. For the third scenario, the MEC application, requiring low latency access (reflected in the AppD), can be deployed without the need of other components. However, the 5GT-VS has to ensure in the VSD/NSD mapping that the traffic redirection as requested by the MEC application will be reflected in the Network Forwarding Path (NFP) included in the NSD. To summarize, the AppD/NSD validation of the 5GT-VS checks:

- Scenario 1: the presence of: VNFD or AppD of EPC elements, AppD of SPGW-U and MEP, and PNFD of eNodeB(s).
- Scenario 2: the presence of: AppD of MEP and PNFD of eNodeB(s).
- Scenario 3: the presence of NFP reflecting the traffic rule described in the AppD of the MEC application.

5.5.2 Vertical Service Preparation

Description: The workflows described how a vertical creates the description of a new vertical service from a blueprint.

Prerequisite: Blueprints have been onboarded by the provider.

Assumptions: The vertical service does not contain VAs, therefore no VAs have to be onboarded as part of service preparation.

Workflow: A vertical *prepares* a VSD by selecting a VSB and providing the missing parameters. As described in Figure 14, there are three steps taken by the vertical.

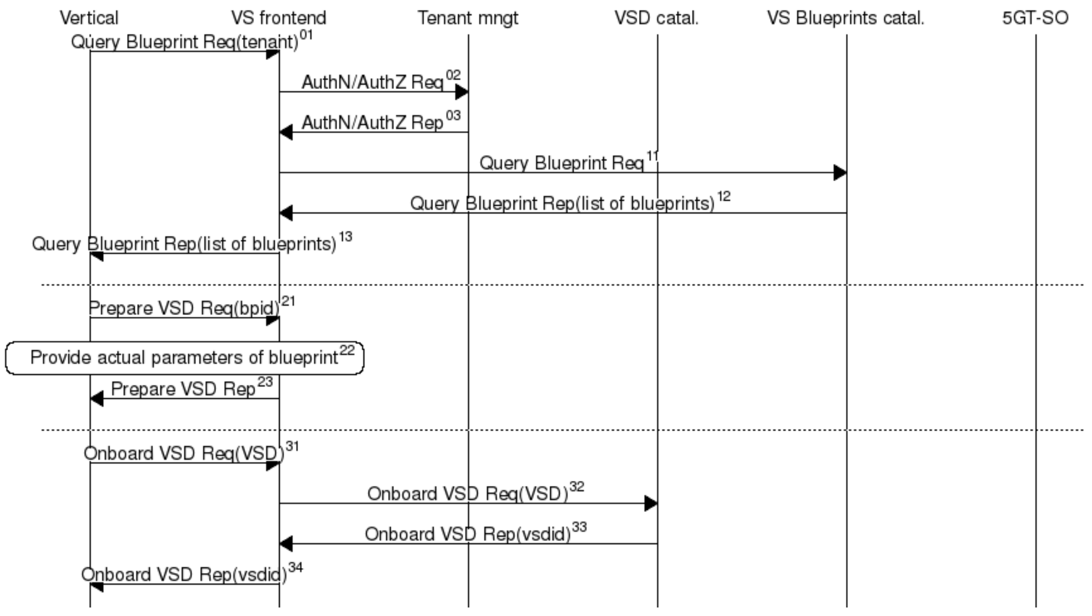


FIGURE 14: VERTICAL SERVICE PREPARATION WORKFLOW

First, the vertical requests the list of available blueprints (01). As part of this step the vertical is authenticated and it is checked whether it is authorized to make this request (02, 03). Within the 5GT-VS, the blueprint catalogue is checked (11, 12) and the list of available blueprints is provided to the vertical (13). Depending on the authorization, the 5GT-VS may present a subset of all the blueprints to a vertical.

Second, the vertical selects a blueprint from the list of available ones and requests the VS frontend to prepare the VSD (21). As part of this step, the missing parameters for this vertical service are provided, such as requirements on latencies, amount of UEs supported, information on VM images for VAs, etc. (22).

Third, the vertical requests to onboard this specific VSD to the 5GT-VS (31), which stores this VSD in the corresponding catalogue (32, 33).

Note that no interaction with the 5GT-SO takes place, this workflow is about creating the description of the vertical service, not about its instantiation.

5.5.3 Vertical Service Instantiation

Description: The workflow describes the instantiation of a vertical service, triggered by the vertical.

Prerequisites: The vertical has selected a vertical service description, which is already onboarded to the vertical slicer.

Assumptions: The vertical service is a simple one, meaning it can be deployed in one NSI; this service is deployed in a new NSI. The NSD to which the vertical service is mapped and which describes the NSI has been onboarded to the 5GT-SO before.

Workflow: The vertical triggers the workflow, see Figure 15, by requesting the instantiation of a vertical service, identified by an identifier for the VSD (01). Before the workflow proceeds further, the vertical is authenticated and its authorization checked with the tenant management component (02, 03). Assuming this is successful, the 5GT-VS checks the existence of the VSD and retrieves it (11, 12), creates an entry for the instance of this vertical service (23) and stores it in its repositories (24). Note, this is an internal bookkeeping of the 5GT-VS, the VSI has not been deployed yet.

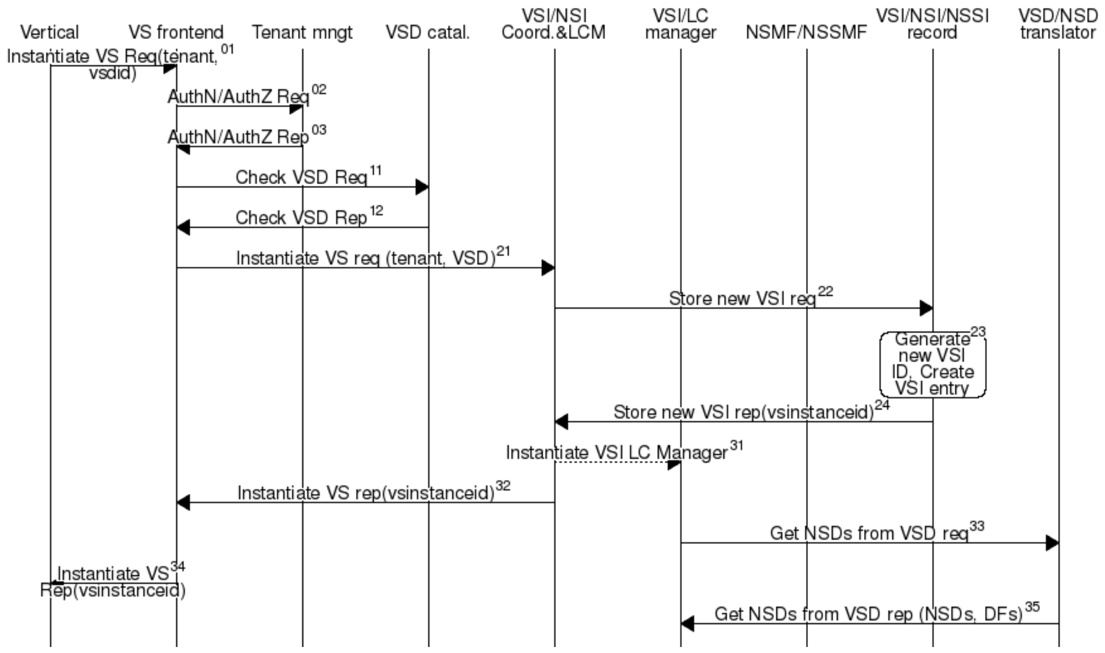

Vertical
VS frontend
Tenant mngt
VSD catal.
VSI/NSI Coord.&LCM
VSI/LC manager
NSMF/NSSMF
VSI/NSI/NSSI record
VSD/NSD translator
Instantiate VS Req(tenant, vsdid) 01
AuthN/AuthZ Req 02
AuthN/AuthZ Rep 03
Check VSD Req 11
Check VSD Rep 12
Instantiate VS req (tenant, VSD) 21
Store new VSI req 22
Generate new VSI ID, Create VSI entry 23
Store new VSI rep(vsinstanceid) 24
Instantiate VSI LC Manager 31
Instantiate VS rep(vsinstanceid) 32
Get NSDs from VSD req 33
Instantiate VS Rep(vsinstanceid) 34
Get NSDs from VSD rep (NSDs, DFs) 35

FIGURE 15: VERTICAL SERVICE INSTANTIATION WORFKLOW, PART 1

The VSI/NSI Coordinator & LCM creates a new instance of the lifecycle manager for this specific vertical service instance (31) and returns the instance id of this VSI to the Vertical frontend (32), which in turn returns it to the vertical (34). Concurrently, the VSI LC manager triggers the translation of the VSD into NSDs (33, 35).

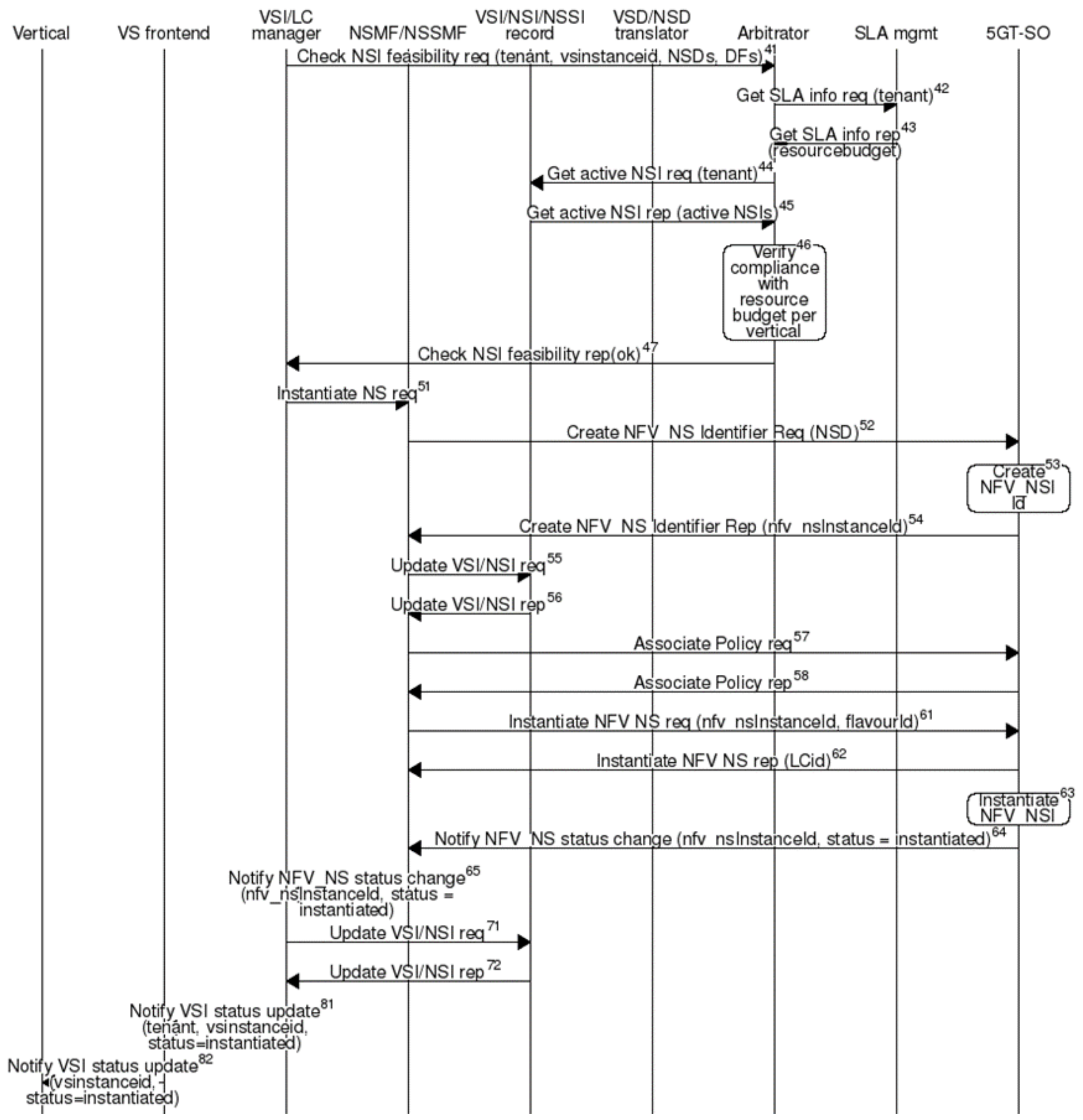

Vertical
VS frontend
VSI/LC manager
NSMF/NSSMF
VSI/NSI/NSSI record
VSD/NSD translator
Arbitrator
SLA mgmt
5GT-SO
Check NSI feasibility req (tenant, vsinstanceid, NSDs, DFs)41
Get SLA info req (tenant)42
Get SLA info rep43 (resourcebudget)
Get active NSI req (tenant)44
Get active NSI rep (active NSIs)45
Verify46 compliance with resource budget per vertical
Check NSI feasibility rep(ok)47
Instantiate NS req51
Create NFV_NS Identifier Req (NSD)52
Create53 NFV_NSI Id
Create NFV_NS Identifier Rep (nfv_nsInstanceId)54
Update VSI/NSI req55
Update VSI/NSI rep56
Associate Policy req57
Associate Policy rep58
Instantiate NFV NS req (nfv_nsInstanceId, flavourId)61
Instantiate NFV NS rep (LCid)62
Instantiate63 NFV_NSI
Notify NFV_NS status change (nfv_nsInstanceId, status = instantiated)64
Notify NFV_NS status change65 (nfv_nsInstanceId, status = instantiated)
Update VSI/NSI req71
Update VSI/NSI rep72
Notify VSI status update81 (tenant, vsinstanceid, status=instantiated)
Notify VSI status update82 (vsinstanceid, status=instantiated)

FIGURE 16: VERTICAL SERVICE INSTANTIATION WORKFLOW, PART 2

As next step, see Figure 16, the 5GT-VS checks whether this vertical service is feasible for the tenant. The VSI LC manager provides the information on the new vertical service and the NSDs, including the deployment flavours, to the Arbitrator (41). The Arbitrator receives the resource budget of this vertical from the SLA manager (42, 43) and the list of active NSIs from the VSI/NSI repository (44, 45), and decides whether the requested resources still fit within the resource budget[18] of this vertical (46).

If the resource budget has not been exhausted already, the VSI LC manager triggers the NSMF (51) to request a new network service identifier from the 5GT-SO (52 - 54). The VSI/NSI record is updated with this information (55, 56) and, if required, policies are associated with the NSI (57, 58). Thereafter, the NSMF requests the actual instantiation of the network slice from the 5GT-SO (61). The NSI is described by a specific deployment

[18] Note, this is a check against the resource budget of the vertical. This is not a check whether the infrastructure has sufficient resources or not.

flavour[19] of the NFV network service instance. The 5GT-SO makes the orchestration decisions on VNF placement and resource allocations (63).

Once the NFV network service instance (i.e. network slice instance) has been instantiated on the actual infrastructure, the 5GT-SO notifies the NSMF (64), which in turn notifies the VSI LC manager (65), which updates the VSI/NSI record (71, 72). Eventually, the vertical is notified[20] about the instantiation of the vertical service (82).

5.5.4 Vertical Service Modification

Description: The workflow describes the modification of a vertical service instance, triggered by the vertical. The modification is a simple one, changing, e.g., the amount of supported devices. In other terms, we assume that a vertical service modification corresponds to a vertical service scaling.

Prerequisites: The VSI has been instantiated.

Assumptions: The vertical service is a simple one, meaning it can be deployed in one NSI; this service is deployed in its own NSI and the service does not share the slice with another service.

Workflow: The workflow consists of two separate steps, see Figure 17 and Figure 18. Firstly, the VSD is changed; secondly, the VSD change is applied to a specific VSI. Note that several instances of the same vertical service could be instantiated and only some of them are modified.

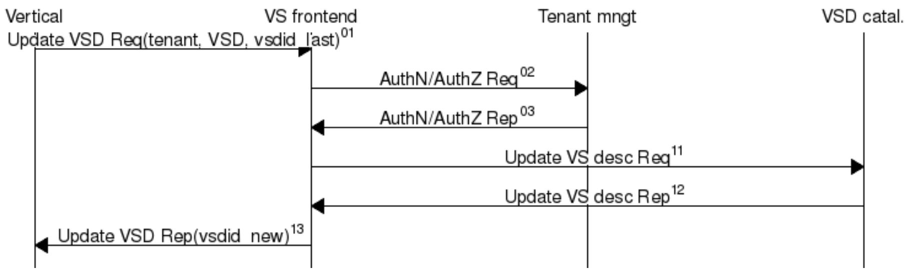


FIGURE 17: VERTICAL SERVICE MODIFICATION WORKFLOW, STEP 1

In the first step, the vertical provides the modified VSD. The relation to the other versions is kept by providing the identifier of the last version of this service descriptor as well (01). Firstly, the tenant is authenticated and its authorization is checked (02, 03). The new VSD is validated for correctness and stored in the corresponding databases (11, 12). An identifier for the new VSD is determined and provided to the vertical (13).

[19] Deployment flavour as defined in ETSI NFV IFA014 [28],
[20] At the implementation level, 5GT-VS GUI periodically polls the 5GT-VS updating the visualized VSI status.

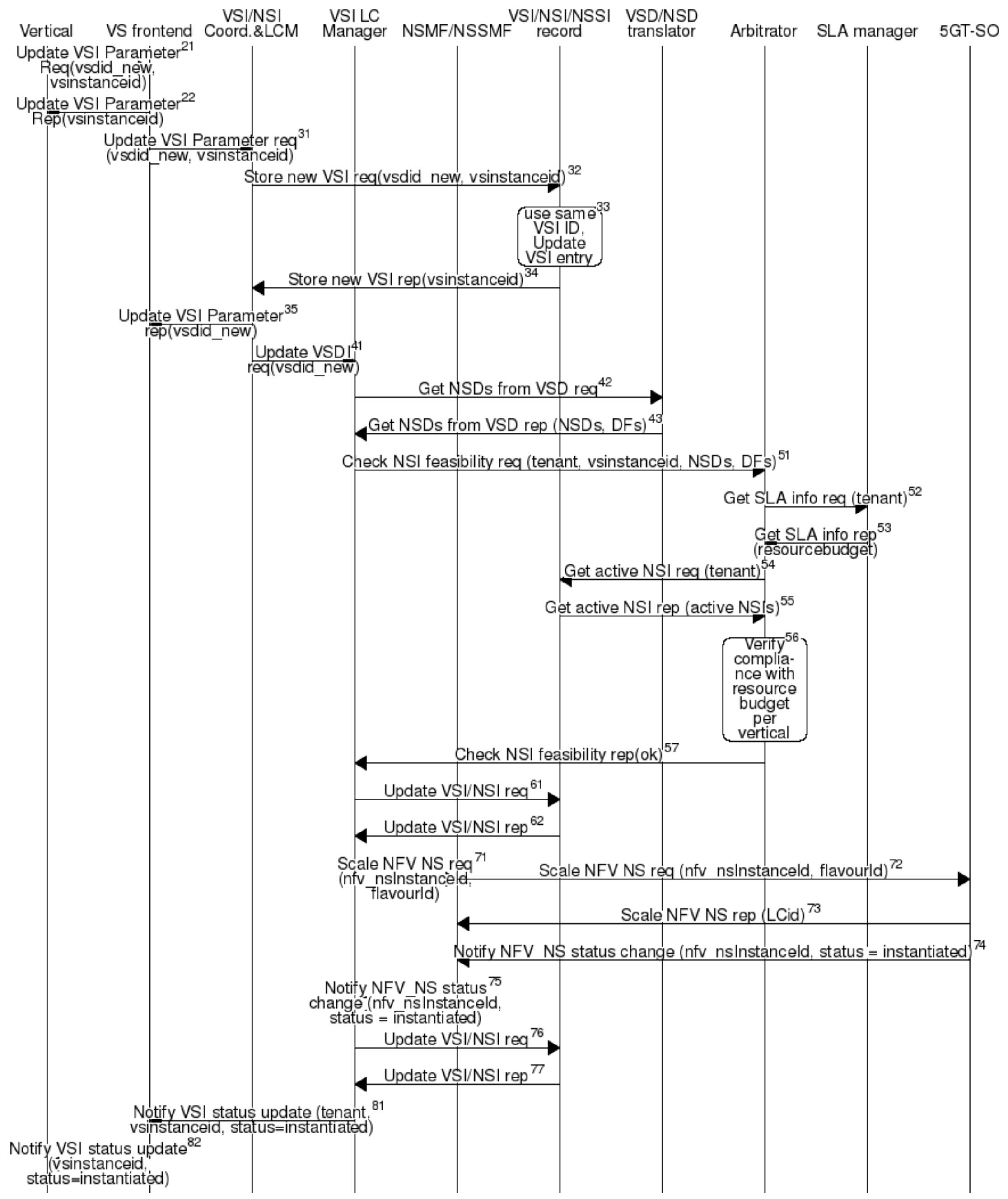

Vertical
VS frontend
VSI/NSI Coord.&LCM
VSI LC Manager
NSMF/NSSMF
VSI/NSI/NSSI record
VSD/NSD translator
Arbitrator
SLA manager
5GT-SO
Update VSI Parameter Req(vsdid_new, vsinstanceid) 21
Update VSI Parameter Rep(vsinstanceid) 22
Update VSI Parameter req (vsdid_new, vsinstanceid) 31
Store new VSI req(vsdid_new, vsinstanceid) 32
use same VSI ID, Update VSI entry 33
Store new VSI rep(vsinstanceid) 34
Update VSI Parameter rep(vsdid_new) 35
Update VSD req(vsdid_new) 41
Get NSDs from VSD req 42
Get NSDs from VSD rep (NSDs, DFs) 43
Check NSI feasibility req (tenant, vsinstanceid, NSDs, DFs) 51
Get SLA info req (tenant) 52
Get SLA info rep (resourcebudget) 53
Get active NSI req (tenant) 54
Get active NSI rep (active NSIs) 55
Verify complia-nce with resource budget per vertical 56
Check NSI feasibility rep(ok) 57
Update VSI/NSI req 61
Update VSI/NSI rep 62
Scale NFV NS req (nfv_nsInstanceId, flavourId) 71
Scale NFV NS req (nfv_nsInstanceId, flavourId) 72
Scale NFV NS rep (LCid) 73
Notify NFV_NS status change (nfv_nsInstanceId, status = instantiated) 74
Notify NFV_NS status change (nfv_nsInstanceId, status = instantiated) 75
Update VSI/NSI req 76
Update VSI/NSI rep 77
Notify VSI status update (tenant, vsinstanceid, status=instantiated) 81
Notify VSI status update (vsinstanceid, status=instantiated) 82

FIGURE 18: VERTICAL SERVICE MODIFICATION WORKFLOW, STEP 2

In the second step, the vertical applies the modified VSD to a VSI. The 5GT-VS can relate the vsinstanceid to the used version of the VSD (21). The operation returns immediately (22), while the actual processing continues.

The specific VSI is updated in the NSI/VSI record (31 - 35). Thereafter the VSI manager of this specific VSI triggers the remapping of the new VSD to the NSDs (42, 43) and the feasibility is checked as for newly instantiated service (51 - 57). Its state is updated in the VSI/NSI record (61, 62).

Thereafter, the VSI manager informs the NSMF (71), which triggers the changes to the NSI by scaling the corresponding NSD at the 5GT-SO. In our simple example this could be a change of the cardinality of VNFs or a change of the deployment flavour. Note, depending on the extent of change of the VSD, the changes to the NSDs could be also more severe and might trigger a more complex interaction between 5GT-VS and 5GT-SO. The NFV_NS Scale operation terminates (73), but the activities continue with applying the changes to the 5GT-MTP.Eventually the 5GT-SO notifies the 5GT-VS that the modification has finished (74, 75). In turn, the VSI LC manager updates the VSI/NSI record (76, 77) and notifies the VS frontend that the modification is finished (81). Finally, the VS frontend notifies the vertical (82).

5.5.5 Vertical Service Termination

Description: The workflow describes the termination of a VSI, triggered by the vertical.

Prerequisites: The VSI has been instantiated.

Assumptions: This VSI is deployed in a single NSI, which is mapped to a single NFV network service.

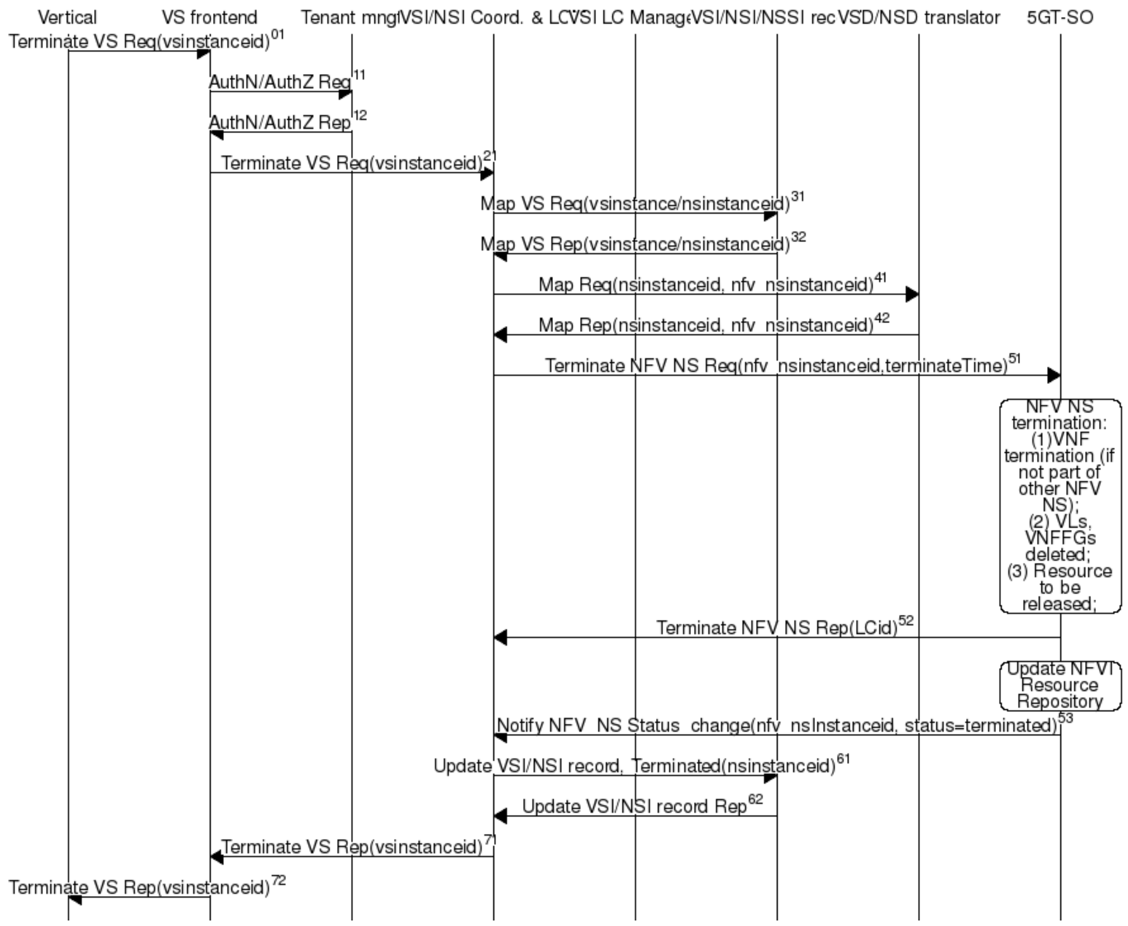

Vertical
VS frontend
Tenant mngt
VSI/NSI Coord. & LC
VSI LC Manage
VSI/NSI/NSSI rec
VSD/NSD translator
5GT-SO
Terminate VS Req(vsinstanceid)01
AuthN/AuthZ Req11
AuthN/AuthZ Rep12
Terminate VS Req(vsinstanceid)21
Map VS Req(vsinstance/nsinstanceid)31
Map VS Rep(vsinstance/nsinstanceid)32
Map Req(nsinstanceid, nfv_nsinstanceid)41
Map Rep(nsinstanceid, nfv_nsinstanceid)42
Terminate NFV NS Req(nfv_nsinstanceid,terminateTime)51
NFV NS termination: (1)VNF termination (if not part of other NFV NS); (2) VLs, VNFFGs deleted; (3) Resource to be released;
Terminate NFV NS Rep(LCid)52
Update NFVI Resource Repository
Notify NFV_NS Status_change(nfv_nsInstanceid, status=terminated)53
Update VSI/NSI record, Terminated(nsinstanceid)61
Update VSI/NSI record Rep62
Terminate VS Rep(vsinstanceid)71
Terminate VS Rep(vsinstanceid)72

FIGURE 19: VERTICAL SERVICE TERMINATION WORKFLOW

The vertical triggers to terminate the VSI by sending the Terminate request (01) using the vsinstanceid via the VS frontend. First, the tenant is authenticated (11, 12). Then, the Terminate request is forwarded to VSI/NSI Coordinator & LC Manager (and VSI manager instance) (21), which maps the vsinstanceid to nsinstanceid, by sending a request to the VSI/NSI record (31, 32).

The VSI/NSI Coordinator & LC Manager sends a request (41, 42) to the VSI/NSD translator to obtain the corresponding nfv_nsInstanceid of the nsinstanceid. Now the VSI Coordinator/NSI Coordinator & LC Manager can send the termination request for the nfv_nsInstanceid with a specific timeout (according to ETSI NFV IFA 013 [11]). The 5GT-SO initiates VNFs termination, VL and VNFFG deletions, and release of orchestrated resources. The 5GT-SO response for termination of the lifecycle for the nfv_nsInstanceId, including the LCid or lifecycleOperationOccuranceId (according to ETSI NFV IFA 013 [11]). Then the 5GT-SO updates the NFVI Resource Repository concerned by the recently released resource. The 5GT-SO sends a notification message with the terminated status of the nfv_nsInstanceId (53).

The VSI/NSI coordinator & LC Manager concludes the termination of the nsInstanceId that corresponds with the terminated nfv_nsInstanceId. It first updates (61, 62) the VSI/NSI record in order to remove the nsinstanceid, and informs the VS frontend (71). The latter (72) will in turn notify the Vertical about the success of the vertical service termination procedure.

5.5.6 Vertical Service Monitoring

This workflow corresponds to the use-case where the 5GT-VS monitors a VSI and consumes the monitoring data itself.

Description: The workflow describes procedures to allow the 5GT-VS to collect monitoring data related to a network service.

Prerequisites: All the descriptors have been on-boarded, the vertical has requested the instantiation of the vertical service and the 5GT-VS has already created an nsi_Id.

Assumptions: The MonitoredData in the NSD includes only MonitoringParameters, not VnfIndicatorInfo. There is a 1:1 relationship between VSI and network service instance (no sharing or nesting).

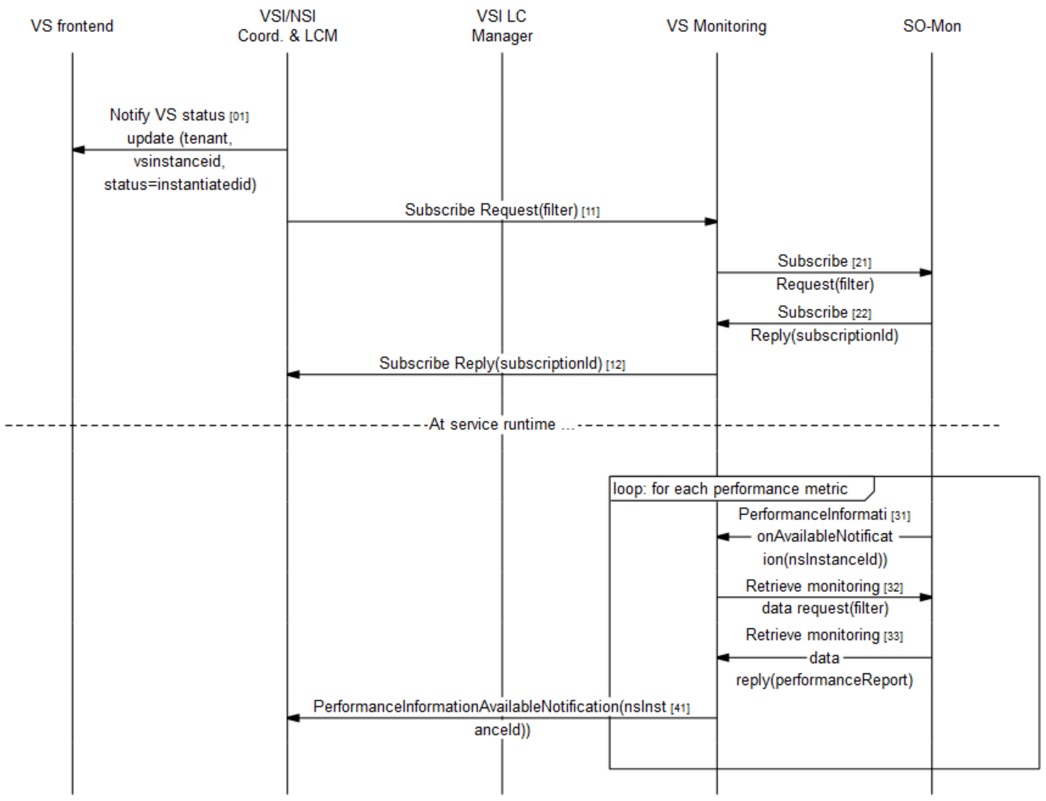

VS frontend
VSI/NSI Coord. & LCM
VSI LC Manager
VS Monitoring
SO-Mon
Notify VS status [01] update (tenant, vsinstanceid, status=instantiatedid)
Subscribe Request(filter) [11]
Subscribe [21] Request(filter)
Subscribe [22] Reply(subscriptionId)
Subscribe Reply(subscriptionId) [12]
At service runtime ...
loop: for each performance metric
PerformanceInformati [31] onAvailableNotificat ion(nsInstanceId))
Retrieve monitoring [32] data request(filter)
Retrieve monitoring [33] data reply(performanceReport)
PerformanceInformationAvailableNotification(nsInst [41] anceId))

FIGURE 20: VERTICAL SERVICE MONITORING WORKFLOW

The monitoring procedure should start only after the vsinstanceid has been instantiated (status=instantiated). The VSI/NSI Coordinator & LC Manager (and VSI manager instance) requests (11) the monitoring component of the 5GT-VS to subscribe to specific monitoring metrics. The monitoring component of the 5GT-VS sends a request to the 5GT-SO Monitoring Service indicating the monitoring metrics to subscribe to (21, 22).

At service runtime, the 5GT-SO Monitoring Service indicates the availability of monitoring data to the 5GT-VS monitoring element. The latter retrieves the data (32, 33), and informs the VSI coordinator (41).

5.5.7 Service Scaling

The workflow for the vertical-driven VSI scaling is reported in Section 5.5.4. It corresponds to the procedure triggered by the vertical when requesting the modification of a VSI, providing a new VSD that can be served through a scaled-up service. The procedure for the automated VSI scaling follows exactly the same workflow in terms of actuation. The only difference is the trigger that originates the scaling action: in this case the scaling is requested by the arbitrator, instead of the vertical through the 5GT-VS NBI. The automated NFV-NSI scaling is handled entirely at the 5GT-SO, both in terms of decision and actuation; for this reason, the related workflow is reported in D4.3 [37].

5.6 Vertical Slicer algorithms

5.6.1 Arbitrator algorithms

Resources are assigned to verticals by the Arbitrator based on their service requests. For each VSI of a vertical, the Arbitrator determines the associated <DFs, ILs>, such that the vertical's QoS requirements are met and accounting for the service priority level. The assignment of resources requires agreeing between the parties (vertical and slice provider, i.e., the 5GT-SO) on service objectives, needs and available resources. The vertical specifies its needs in terms of Service Level Objectives (SLOs). The 5G-T service provider, instead, will define service level classes and guarantees on resource availability through Service Level Agreement Templates (SLAT). The matching of SLOs desired by the vertical and SLATs offered by a provider will converge into an SLA.

The Arbitrator assigns resources among VSIs of the same vertical. As an example, we consider a vertical from the automotive domain, whose SLA foresees the availability of CPU (C), bandwidth (B), memory (M), and storage (S). In this example, the vertical wants to deploy two VSIs: an Extended Virtual Sensing (EVS) service - aimed at preventing collisions between vehicles -, and a Multimedia Streaming (MS) service, with EVS having priority over MS.

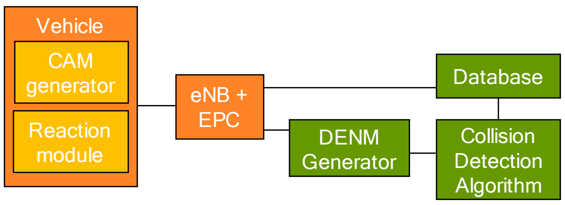


FIGURE 21: VNFFG OF THE EVS SERVICE

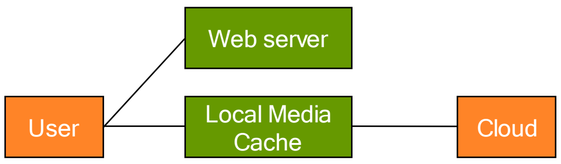


FIGURE 22: VNFFG OF THE MS SERVICE

At first, the Arbitrator considers the highest-priority service instance, EVS in our case, and allocates memory and storage based on the needs exhibited by the VNFs in the VNF Forwarding Graph (VNFFG) representing it (see Figure 21). CPU and bandwidth allocation, instead, is more complex, as it depends on the VNF placement decisions by the 5GT-SO. Let us assume that the main performance metric of service *s* is the maximum latency D_s, which depends on two components. Firstly, it depends on the processing time to execute the VNFs, given by the CPU allocated to the VNFs execution. Secondly, if the 5GT-SO chooses different servers for the implementation of the VNFs, it depends on the network travel time, due to the time needed to transfer data from one VNF to the next in the VNFFG, and on the bandwidth associated with the VL connecting the servers. Specifically, for a set of VNFs *V*, the Arbitrator will compute the allocation for the EVS service in terms of the CPU μ and bandwidth β that satisfy the following conditions:

$$\sum_{v\in V} \frac{1}{f_v \mu - \lambda_v} + \sum_{(u,v)\in E} \frac{d_{u,v}}{f_{u,v}\beta} \le D_s \qquad (3)$$

$$\mu \le C \,;\, \beta \le B \,;\, \mu/\beta = C/B \qquad (4)$$

where f_v is the relative computational requirement of VNF $v \in V$, $f_{u,v}$ is the relative bandwidth requirement for the VL connecting VNFs $u, v \in V$ and $d_{u,v}$ is the amount of data exchanged by *u* and *v*. (3) accounts for the latency due to both the VNF execution (modelling the generic VNF *v* as a FIFO M/M/1 queue with $f_v\mu$ as the output rate and λ_v as the service request rate input to *v*) and the travel time over the VLs connecting any two adjacent VNFs; (4) imposes that both the total CPU and bandwidth allocations do not exceed the corresponding budget available to the vertical; it also imposes that the ratio between μ and β be equal to the ratio of C and B in order to ensure a consumption of the different types of resources proportional to the corresponding budgets. Finally, it is worth pointing out that we neglected the delay due to memory/storage access.

Conditions (3) and (4) represent a flexible deployment, allowing to place VNFs in different PoPs. As network travel time has to be compensated by shorter computation, this is a worst-case scenario for compute resources. Vice versa, if all VNFs are deployed in the same PoP, the dependence on network travel time can be neglected, leading to a best-case scenario. Thus, the bandwidth required for data transfer can be set to zero and the allocated CPU can be computed as:

$$\sum_{v\in V} \frac{1}{f_v \mu - \lambda_v} \le D_s \qquad (5)$$

$$\mu \le C \qquad (6)$$

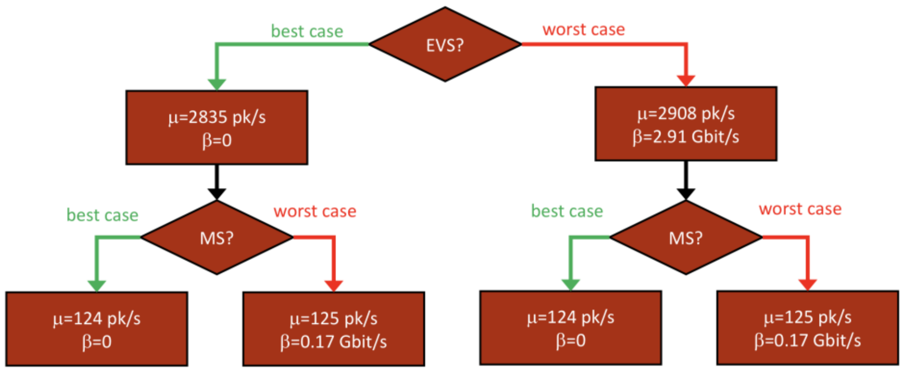


FIGURE 23: BEST AND WORSE-CASE ALLOCATIONS FOR THE EVS AND THE MS SERVICES

The Arbitrator will then proceed with the MS service (whose VNFFG is shown in Figure 22), following the same steps as above but using the remaining budget available to the vertical in terms of CPU and bandwidth. The MS uses only two virtual applications: one that caches local copies of the video files, and effectively serves the video files to the user; the other one is a web server providing the video player to the user through Javascript.

Let us now provide a numerical example. We start by listing the SLOs:

- EVS Service Level Objectives:
- Geographical coverage radius: 500 m
- Maximum latency D_{EVS}: 20ms
- Maximum service request rate: 60/s
- MS Service Level Objectives:
- Geographical coverage radius: 1000 m
- Maximum latency D_{MS}: 5s
- Minimum data rate: 200 kb/s
- Maximum number of simultaneous streams: 6.

We will also assume (considering experimental results from consumer-grade hardware) C=10000 packets/s and B=10Gbit/s and that the relative computational requirement f_v for the database and VNFs of the EVS service are both 0.05, which yields f_v for the collision detector equal to 0.9. For the MS service, we assume f_v = 0.95 for the media cache and 0.05 for the Web server.

The solutions of equations (3)-(6) using numerical methods for both the worst and the best case can be summarized by the tree structure of Figure 23. Note that the 5GT-VS will send to the 5GT-SO within DFs the minimum and maximum values of required CPU and bandwidth for EVS and MS, namely, $\mu_{\text{EVS}} \in [2835,2908]$ packets/s and $\beta_{\text{EVS}} \in [0, 2.91]$ Gbit/s and $\mu_{\text{MS}} \in [124,125]$ packets/s and $\beta_{\text{MS}} \in [0, 0.17]$ Gbit/s.

For services of several verticals, the Arbitrator decides on the deployment of services based on the SLAs, starting with the verticals that have highest priority in the agreements. Verticals of the same priority have the same probability to be deployed:

$$P(v) = \frac{1}{|I_v|} \tag{7}$$

where *P(v)* is the probability of selecting one vertical of this priority, and I_v is the set of verticals in the priority considered. This guarantees that all the verticals in the same priority have the same probability to be chosen to start the deployment. In addition, we remove the ones that are considered each time from the set to continue with the ones that have not been deployed already. This guarantees that a vertical is not selected several times while another one is not selected at all, which makes the probability non-dependent on the number of services of a vertical.

This algorithm can be extended to the case of VNFs shared among different VSIs, even of different verticals. In this case, the mapping of VSIs to NSIs and resource assignment is intertwined. If a vertical requests to instantiate a vertical service using VNFs already deployed for some other VSI, the Arbitrator will check SLAs for isolation requirements. If the new or the existing VSIs are required to be isolated, the Arbitrator cannot share these VNFs and will map the newly requested vertical service to a new NSI and the previous algorithm can be used. If there is no requirement to isolate the new and the existing VSIs, the Arbitrator can map the newly requested service to an existing NSI and share the VNFs. The Arbitrator will modify the DFs of shared VNFs increasing the amount of resources, such that the VNFs can handle the traffic of the new VSI as well. This is translated in the formulae by modifying the arrival and processing rates and comparing them against the minimum of the latency requirement of the VSIs.

$$\sum_{v\in V}\frac{1}{\sum_{s\in S}(\mu_v^s-\lambda_v^s)}+\sum_{(u,v)\in E}\frac{d_{u,v}}{\beta_v}\leq D_s \qquad (8)$$

The arrival and processing rates now depend on all the services that use the same instance of the VNF. S is the set of services that share at least one VNF instance with the demand we are placing now and it appears in the first part of the formula modifying the total delay.

An initial implementation of the 5GT-VS prototype shows that the overhead added by the Arbitrator's algorithm to the total provisioning time for an end to end service is minimal, in the order of 1%. For example, considering the provisioning of a simple virtual Content Delivery Network with one origin server and two caches, all deployed as VNFs, the 5GT-VS processing is in the order of 1.8 seconds, while the provisioning time at the NFVO and Virtual Infrastructure Manager level takes more than 2.5 minutes, with most time needed for the creation of the virtual machines and the configuration of the applications.

The performance advantage of the 5GT-VS. We now consider a synthetic scenario with three different services s_1, s_2 and s_3, each with the same requirements and whose priorities are 1, 2 and 3, respectively. We compare the arbitration algorithm implemented by the 5GT-VS with a simple first-come-first-serve (FCFS) policy, where:

- service requests are processed - and granted, if there are enough resources - in the same order they arrive
- services are assigned a uniform share of the SLA resources.

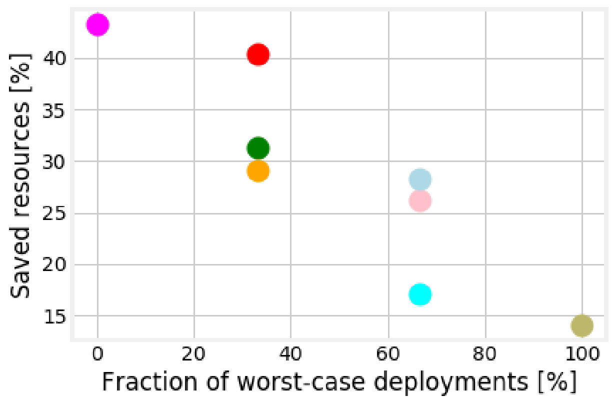


FIGURE 24: SLA RESOURCES SAVED WHEN USING THE 5GT-VS, COMPARED TO THE FCFS/UNIFORM CASE

In Figure 24, different dots correspond to different combinations of best- and worst-case deployment decision by the 5GT-SO, i.e., different leaves of the tree in Figure 23. We can clearly see that the 5GT-VS guarantees very significant savings, reaching 40% when all 5GT-SO decisions match the best-case perspective and being as high as 15% even when all worst-case decisions are made.

Next, we want to assess to which extent the 5GT-VS is able to honour service priorities, i.e., only rejecting the lowest-priority service. To this end, we reduce the allowable resource utilization and visualize the average priority of rejected services (recall that service priorities are between 1 and 3). As we can clearly see from Figure 25, the 5GT-VS is able to make better decisions as to which service to reject, i.e., to reject lower-priority services while serving higher-priority ones.

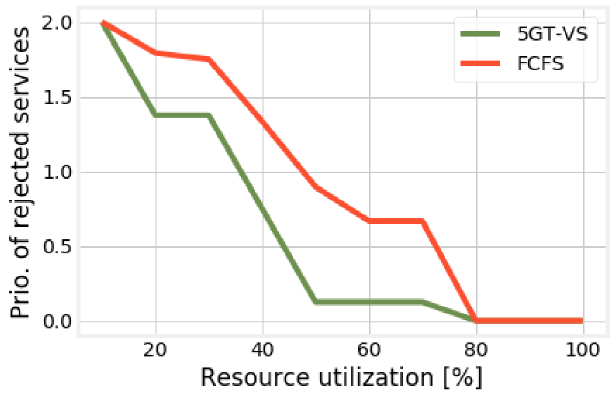


FIGURE 25: PRIORITY OF REJECTED SERVICES WITH THE 5GT-VS AND UNDER THE FCFS/UNIFORM STRATEGY

5.6.2 Arbitrator algorithm to handle service fallback to alternative VNF graphs

The 5GT-VS also supports the case where multiple VNFFGs exists for the same service, possibly associated with different levels of preference for the vertical, different revenues for the mobile operators, or different requirements in terms of SLA resources.

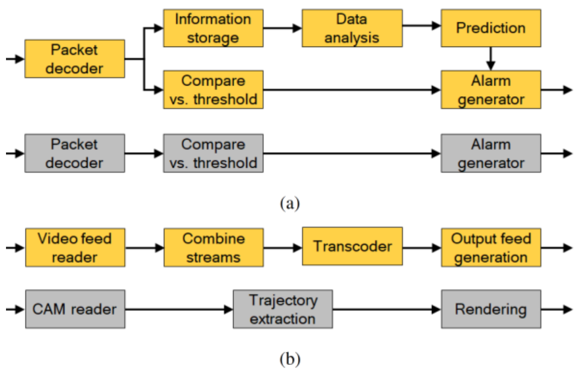


FIGURE 26: PRIMARY (GOLD BACKGROUND) AND SECONDARY (SILVER BACKGROUND) VNF GRAPHS ASSOCIATED WITH A SENSOR MONITORING (A) AND A VEHICULAR SEE-THROUGH (B) SERVICE

A good example is the sensor monitoring service presented in Figure 26(a): in ordinary conditions, sensor readings are checked against static thresholds and used for prediction. An alarm is generated if current values exceeded the static threshold or the predicted values are detected as anomalous. However, if a resource shortage prevents the primary VNF graph from being deployed, there is a benefit in *at least* being able to raise an alarm if thresholds are exceeded, by implementing the bottom VNF graph in Figure 26(a). Implementing such a *secondary* graph is preferable, for both the vertical and the mobile operator, to not implementing the service at all.

The vehicular see-through service presented in Figure 26(b) provides a second example of such a situation. Large vehicles, e.g., trucks, are equipped with cameras capturing their view of the road. Such video streams are transmitted to the infrastructure where they are read, combined, transcoded and served to the vehicles whose view would otherwise be obstructed. The VNFs required by such a service, represented in the top part of Figure 26(b), require significant computational and bandwidth resources. If such resources are not available, the see-through service can be implemented through the VNFs in the lower part of Figure 26(b). Therein, instead of relying on video streams, cooperative awareness message (CAMs) are leveraged to construct a schematic view of the positions of the preceding vehicles. The two VNF graphs serve the same purpose - warning following vehicles of potential hazards - in different ways, associated with different levels of resource consumption.

Note that, as exemplified above, the secondary graph is either a subset of the primary one, or it includes a (smaller) number of VNFs, each of which characterized by lower requirements. It follows that deploying the secondary graph of a certain service *in lieu* of the primary one will always lead to shorter delays and a reduced resource consumption, at the price of a lesser quality of experience for the user.

Figure 27 presents a simplified vision of how 5G-T entities interact when making and enacting decisions concerning which VNF graph to use for each service. In steps 1-2, the vertical informs the 5GT-VS of the requirements associated with the different versions of its services. In steps 3-8, the vertical requests to the 5GT-VS the deployment of services s_1 and s_2. After checking the available SLA resources, the 5GT-VS decides to deploy the primary graph of s_1 and the secondary one of s_2, and instructs the 5GT-SO accordingly.

In step 9, the monitoring platform detects a resource shortage situation and informs the 5GT-SO, which relays the warning to the 5GT-VS. Such a situation requires to switch a service to a lesser VNF graph, and the 5GT-VS decides to move s_1 from primary to secondary. The decision is then notified to the 5GT-SO, which enacts it by removing the VNFs associated with the primary graph of s_1 and deploying those of the secondary graph.

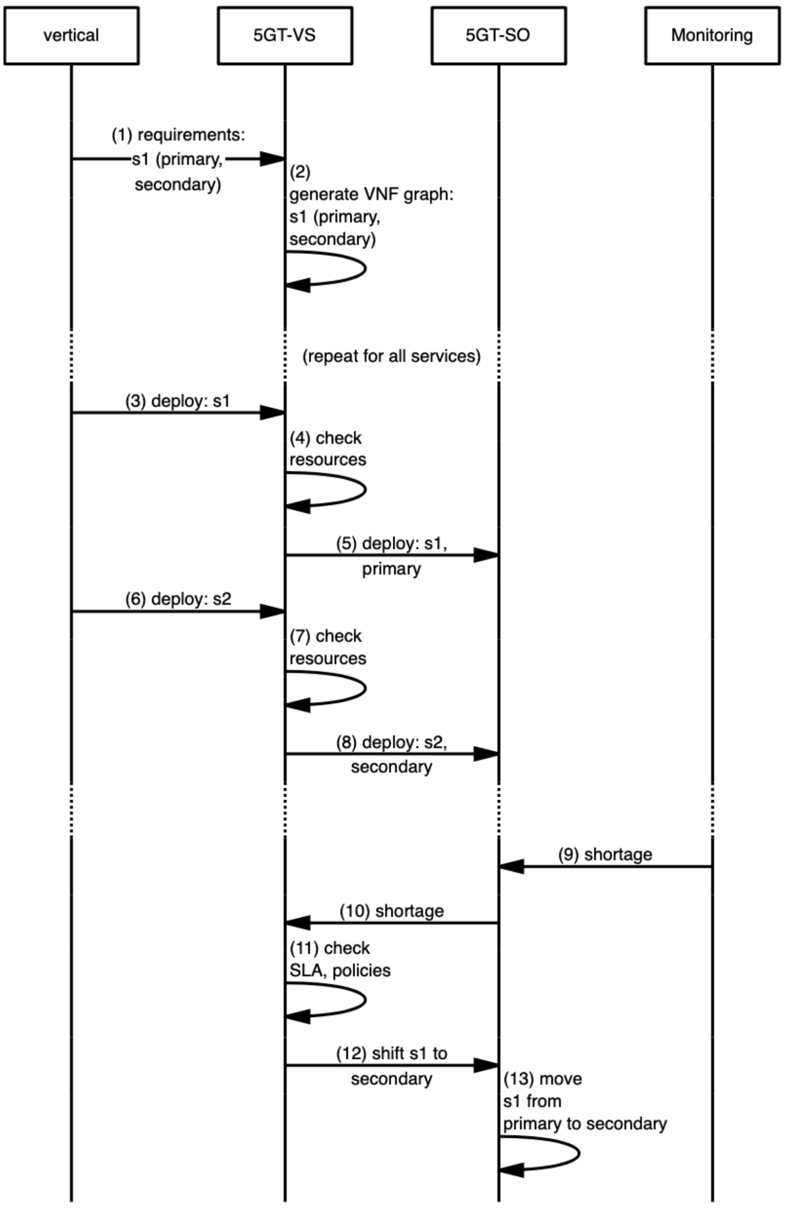

vertical
5GT-VS
5GT-SO
Monitoring
(1) requirements: s1 (primary, secondary)
(2) generate VNF graph: s1 (primary, secondary)
(repeat for all services)
(3) deploy: s1
(4) check resources
(5) deploy: s1, primary
(6) deploy: s2
(7) check resources
(8) deploy: s2, secondary
(9) shortage
(10) shortage
(11) check SLA, policies
(12) shift s1 to secondary
(13) move s1 from primary to secondary

FIGURE 27: SWITCHING BETWEEN VNF GRAPHS: SIMPLIFIED SEQUENCE DIAGRAM

5.7 Applicability of Vertical Slicer to 5G-T use cases

This section provides a brief description of 5G-T use cases, focusing on the 5GT-VS functionalities that are required for each use case and the characteristics of the related service blueprints and descriptors. The service blueprints in JSON format are reported in appendix, in section 5.13.

5.7.1 5GT-VS in automotive use case

The automotive industry is currently undergoing key technological transformations, as more and more vehicles are connected to the Internet and to each other, and advances towards higher automation levels. To deal with increasingly complex road situations, automated vehicles will have to rely not only on their own sensors, but also on those of other vehicles, and will need to cooperate with each other, rather than to make decisions on their own.

These trends pose significant challenges to the underlying communication system, as information must reach its destination reliably within an exceedingly short time frame - beyond what current wireless technologies can provide.

The automotive use case presented below addresses the class of services referred to as Extended Virtual Sensing (EVS). With EVS, the network infrastructure collects and makes available measurements gathered by sensors aboard vehicles, as well as by smart city sensors, to improve road safety and passengers/driver comfort. The EVS selected in the 5G-T project consists of a set of applications to be activated in the proximity of an intersection, aiming at avoiding the risk of collisions between approaching vehicles. The service is designed to alert drivers about the presence of unseen vehicles and, possibly, it is the enabler for vehicles to autonomously activate the emergency braking system. Indeed, as mentioned above, although vehicles can count on data coming from multiple information sources (e.g., onboard Advanced Driver-Assistance Systems (ADAS) sensors), such data can be enhanced with the information the vehicle receives from the infrastructure, which, having a centralized view of the monitored area, can lead to better decisions by the vehicle control logic.

The EVS application exploits real-time data transmitted periodically by vehicles (namely, the so-called Cooperative Awareness Messages (CAM) messages) and collected by a third-party entity, the Cooperative Information Manager (CIM). The CAMs include position, heading, speed, and acceleration of the transmitting vehicle at the monitored crossing. Based on such information, the EVS algorithm evaluates each pair of vehicles and estimates possible collisions with a trajectory-based algorithm. If a possible collision is detected, the EVS application generates an "Alert" string, which is then encoded into a Decentralized Environmental Notification Message (DENM) message. DENMs are sent by the infrastructure to the vehicles that have been detected to be on collision course. It should be noted that alerts should be generated in real-time and received at the vehicle with a very low latency, which gives the car enough time to react and avoid the collision. For this reason, the EVS service components should run close to the vehicles, at the edge of the network. As consequence, they are modelled as MEC applications.

In 5G-T automotive use case, the EVS service is instantiated in combination with a video streaming service, which represents a low-priority service. In this scenario, the 5GT-VS needs to arbitrate between the two services, in order to maintain the compliance with the resource budget declared in the SLAs established with the vertical. Moreover, the EVS service can automatically scale, according to its real-time performance. In summary, the list of 5GT-VS features that are required for the automotive use case is the following:

- Provisioning and termination of vertical service instances.
- Monitoring of vertical service instances.
- Arbitration among multiple services with different priorities.
- Service scaling (automated NFV-NSI scaling option).

- Service configuration.
- Support of MEC Applications.

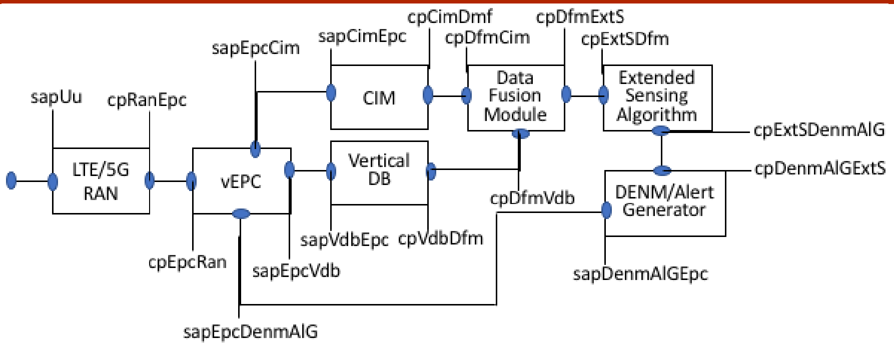
The VSB of such an EVS service is detailed below, along with the workflows of the procedures involving the 5GT-VS entity.

5.7.1.1 Blueprint for the EVS automotive service

The main fields of the EVS VSB are reported in Table 22 (the full JSON format is reported in section 5.13.1), while the complete VSBs and VSDs of the automotive use case are available in the Github repository at the following link:

`https://github.com/5g-transformer/5gt-vs/tree/master/blueprints/automotive/`

TABLE 22: VSB FOR AUTOMOTIVE SERVICE

Field	Description
Service graph	 cpCimDmf cpDfmExtS sapCimEpc cpDfmCim cpExtSDfm sapEpcCim Data Fusion Module Extended Sensing Algorithm sapUu cpRanEpc CIM cpExtSDenmAlG Vertical DB cpDenmAlGExtS LTE/5G RAN vEPC cpDfmVdb DENM/Alert Generator sapVdbEpc cpVdbDfm cpEpcRan sapEpcVdb sapDenmAlGEpc sapEpcDenmAlG
SST	URLLC
Atomic functional components	Mobile Communication Transport (eNB and vEPC) Automotive vertical's DB: database that stores the extended CIM information, specific for the given car maker, as retrieved from the CAM messages generated by the vehicles of the car maker. CIM: database that stores the standard CIM information, as retrieved from the CAM messages generated by all the vehicles. Data fusion module: component that aggregates data from the CIM databases. Extended sensing algorithm: EVS algorithm estimating potential collisions between vehicles. DENM/Alert generator: generator of alerts and DENM messages towards the vehicles.
Parameters	Expected number of vehicles Geographical area coordinates Geographical area coverage Maximum service end-to-end latency (latency from the CAM transmission by the car to DENM reception at the car, in ms)

	Size of the vertical DB
SLA	Security level: High Priority level: High Reliability level: 99% Resilience level: 99% Maximum target latency: 20 ms

5.7.1.2 5GT-VS workflows for automotive service

In the case of the EVS service, the 5GT-VS operates following the same workflow as for any other service requested by a vertical. However, given the high priority characterizing the EVS service, it is worth specifying the procedure followed by the Arbitrator when multiple services with different priority levels are requested by the automotive vertical.

Consider the specific case where the automotive vertical requests both the EVS service and the video streaming services for its vehicular users. Since the EVS has higher priority than video streaming, the Arbitrator will process it first. Let us then assume that the initial expected traffic load for the EVS is quite low, thus the services is instantiated using a “small” Instantiation Level (IL) (see Figure 28). Other values of IL are computed considering that a moderately increase of the traffic may take place during the service instance lifetime. Instead, the video streaming service is instantiated using a “large” IL, due to the expected high-definition video traffic load (see Figure 29). Next, the vehicular traffic at the intersection increases, and in order to keep ensuring the target EVS service latency, the 5GT-SO will have to scale up/out the service according to the scaling rules encoded in the NSD. However, the 5GT-SO scaling operation may fail, e.g., applying the most favourable IL, the monitored latency of the EVS does not drop below the maximum target values. If such an event occurs, the 5GT-SO notifies the 5GT-VS about the impossibility to properly scale the EVS and a new arbitration between the video streaming and the EVS services is performed at the 5GT-VS (see Figure 30). As a result, the Arbitrator changes the IL for the EVS service from “small” to “large”, and the one for the video streaming service, from “large” to “small”. Thus, the 5GT-VS instructs the 5GT-SO to first scale down/in the video streaming, according to the newly provided IL, and then to scale up/out the EVS service, again according to the newly provided IL.

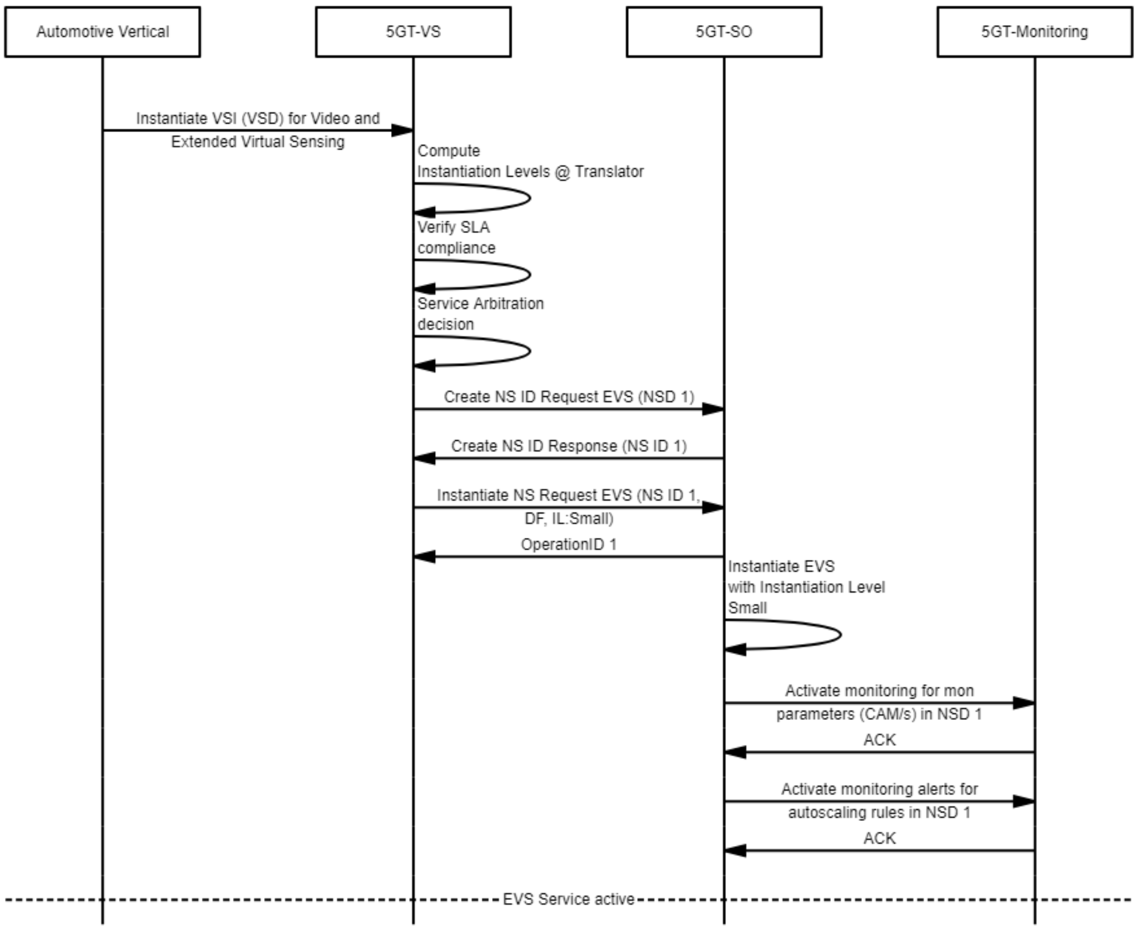

Automotive Vertical
5GT-VS
5GT-SO
5GT-Monitoring
Instantiate VSI (VSD) for Video and Extended Virtual Sensing
Compute Instantiation Levels @ Translator
Verify SLA compliance
Service Arbitration decision
Create NS ID Request EVS (NSD 1)
Create NS ID Response (NS ID 1)
Instantiate NS Request EVS (NS ID 1, DF, IL:Small)
OperationID 1
Instantiate EVS with Instantiation Level Small
Activate monitoring for mon parameters (CAM/s) in NSD 1
ACK
Activate monitoring alerts for autoscaling rules in NSD 1
ACK
EVS Service active

FIGURE 28: EVS INSTANCE REQUEST

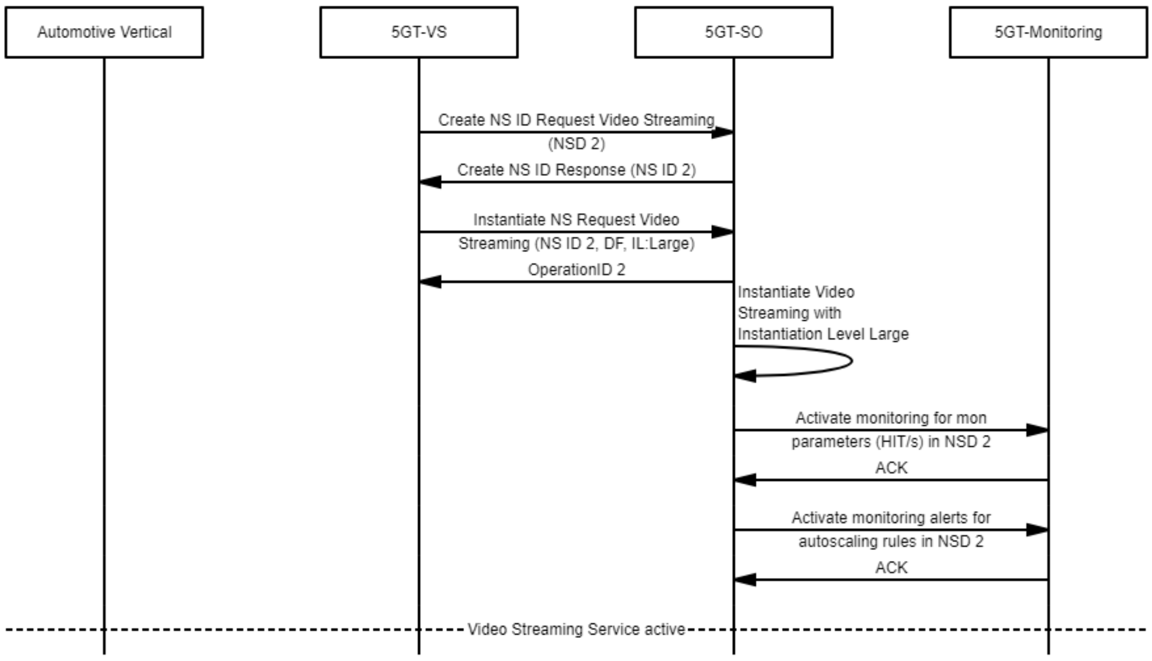

Automotive Vertical
5GT-VS
5GT-SO
5GT-Monitoring
Create NS ID Request Video Streaming (NSD 2)
Create NS ID Response (NS ID 2)
Instantiate NS Request Video Streaming (NS ID 2, DF, IL:Large)
OperationID 2
Instantiate Video Streaming with Instantiation Level Large
Activate monitoring for mon parameters (HIT/s) in NSD 2
ACK
Activate monitoring alerts for autoscaling rules in NSD 2
ACK
Video Streaming Service active

FIGURE 29: VIDEO STREAMING INSTANTIATION

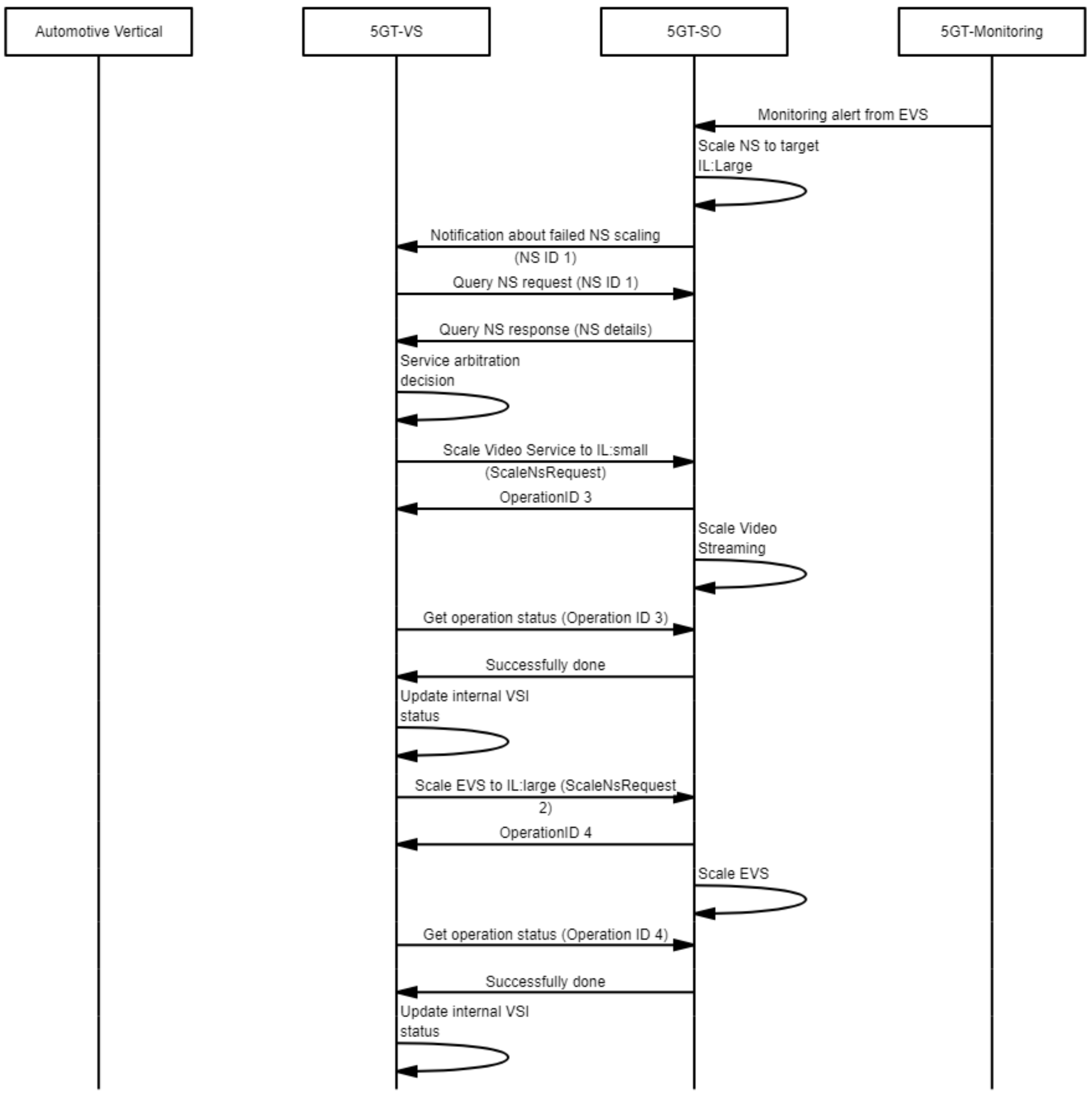

Automotive Vertical
5GT-VS
5GT-SO
5GT-Monitoring
Monitoring alert from EVS
Scale NS to target IL:Large
Notification about failed NS scaling (NS ID 1)
Query NS request (NS ID 1)
Query NS response (NS details)
Service arbitration decision
Scale Video Service to IL:small (ScaleNsRequest)
OperationID 3
Scale Video Streaming
Get operation status (Operation ID 3)
Successfully done
Update internal VSI status
Scale EVS to IL:large (ScaleNsRequest 2)
OperationID 4
Scale EVS
Get operation status (Operation ID 4)
Successfully done
Update internal VSI status

FIGURE 30: SERVICE ARBITRATION IN AUTOMOTIVE USE CASE

5.7.2 5GT-VS in entertainment use case

The entertainment industry is experiencing a great challenge to deliver media-rich content providing an optimum quality of experience to a great number of users. It is for this reason that the entertainment industry has plenty of interest in 5G technologies since there is a need to improve network infrastructures to address the challenge of new media services with demanding data rates and latency to be able to deliver an immersive experience to the users. 5G technologies supply the service providers with network information that can be very relevant to deploy successfully the services that the industry is demanding. The entertainment use case aims to implement the FAN ENGAGEMENT trend providing high-quality content and giving the user a more interactive experience. In particular, a relevant case is represented by the situation where this service experiences a high demand in terms of media-rich content that should be provided to users with an optimum quality of experience.

In 5G-T the entertainment use case includes the instantiation, monitoring, and dynamic scaling of a virtual Content Delivery Network (vCDN) service, where the video caches are deployed at the edge of the network. In particular, the list of 5GT-VS features that are required for this use case is as follows:

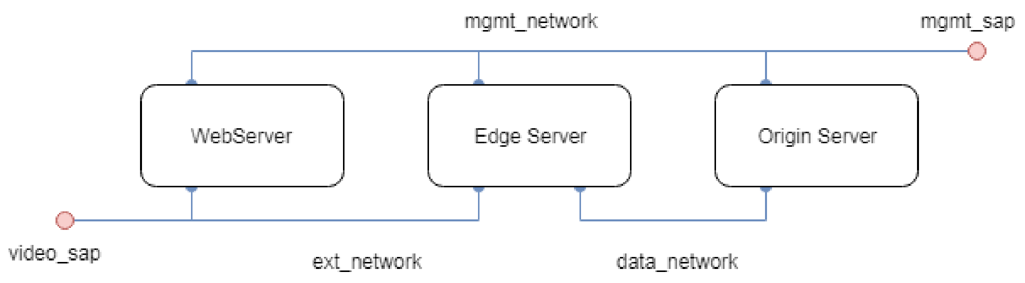
- Provisioning and termination of vertical service instances.
- Monitoring of vertical service instances.
- Service scaling (automated NFV-NSI scaling option).
- Support of MEC Applications.

5.7.2.1 Blueprint for entertainment services

The main fields of the vCDN VSB are reported in Table 23, while the complete VSB and VSD of the entertainment use case are available in the Github repository at the following link:

`https://github.com/5g-transformer/5gt-vs/tree/master/blueprints/entertainment/`

TABLE 23: VSB FOR ENTERTAINMENT SERVICE

Field	Description
Service graph  mgmt_network mgmt_sap WebServer Edge Server Origin Server video_sap ext_network data_network	
SST	eMBB
Atomic functional components	Origin server: to store the video content. Edge server: cache serving end user requests. Webserver: storing web-based video player and static content to be downloaded by the end user.
Parameters	Number of users
SLA	Maximum storage limit: 100 GB Maximum CPU limit: 4 Maximum memory limit: 8192 MB

5.7.2.2 5GT-VS workflows for entertainment services

The entertainment service follows the instantiation workflow of any other vertical and it is fully aligned with the Vertical Slicer workflows described in Section 5.5. Moreover, in the entertainment service, the number of caches (edge servers) can be dynamically modified by the 5GT-SO according to the traffic load. As shown in Figure 31, the scaling action (from an initial Instantiation Level IL “A” to a target Instantiation Level IL “B”) is regulated through the autoscaling rules defined in the NSD and based on the measurements of monitoring parameters (also specified in the NSD) collected by the 5GT-SO Monitoring

Platform. The autoscaling procedure is then notified from the 5GT-SO to the 5GT-VS, in order to keep the synchronization between 5GT-VS and 5GT-SO about the status and the amount of resources for the different NFV-NSs.

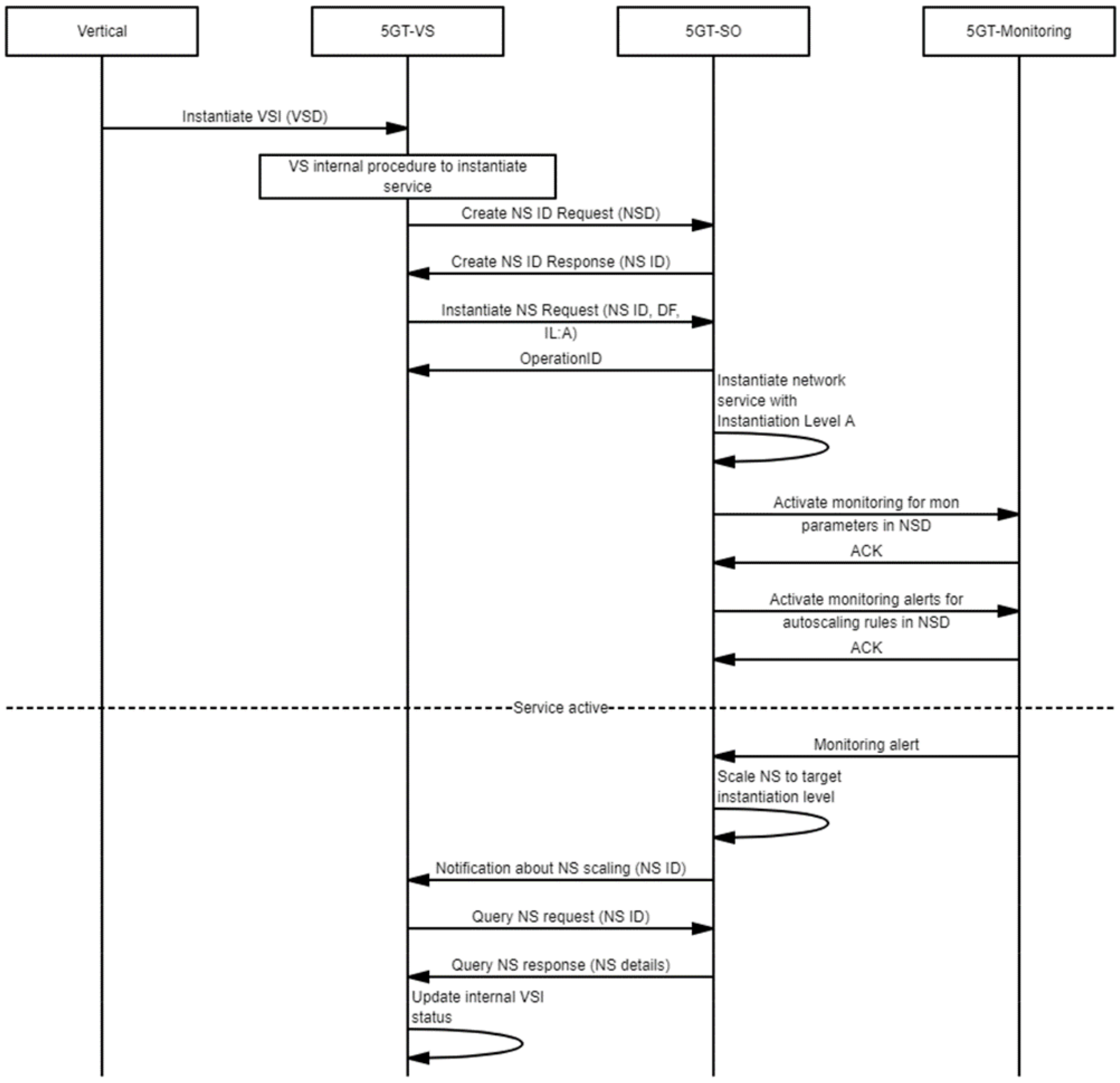

Vertical
5GT-VS
5GT-SO
5GT-Monitoring
Instantiate VSI (VSD)
VS internal procedure to instantiate service
Create NS ID Request (NSD)
Create NS ID Response (NS ID)
Instantiate NS Request (NS ID, DF, IL:A)
OperationID
Instantiate network service with Instantiation Level A
Activate monitoring for mon parameters in NSD
ACK
Activate monitoring alerts for autoscaling rules in NSD
ACK
Service active
Monitoring alert
Scale NS to target instantiation level
Notification about NS scaling (NS ID)
Query NS request (NS ID)
Query NS response (NS details)
Update internal VSI status

FIGURE 31: WORKFLOW FOR ENTERTAINMENT USE CASE

5.7.3 5GT-VS in e-Health use case

The e-Health use case is one of the most critical verticals in the 5G-T project, since it deals with people's health and life. The health industry can effectively take advantage of the future 5G networks to improve the quality of life and medical assistance of people in emergency situations. It aims to detect and assist people in emergencies in the minimum possible time in order to assure the maximum probability of people to recover from an emergency.

The ability of 5G networks to support high demands of traffic with low delay requirements perfectly meets the e-Health requirements as it allows discovering and attending emergencies in short time. In particular, there are two main targets for this use case: e-Infrastructure and e-Health application.

The e-Infrastructure use case focuses on how critical emergencies are detected and ambulances alerted and instructed to reach the emergency location. Also, it is important to provide the medical staff with access in real time to the clinical history of the patient at the place of the incident, in order to give the patient a better medical attention. To realize these goals, the 5G-T system allows scaling up the resources of the e-Health slice in extreme cases where the network is overloaded like in big events. In this way, the appropriate amount of resources will be allocated dynamically, adapting them to the amount of traffic to be managed and being able to meet the service requirements in spite of the high load. Note that, the e-Health use case will need to be marked as a high-priority and low-latency service in the 5G-T system.

The e-Health application, instead, aims to study how new technologies such as Edge Computing can help improving the speed of response. This application tries to reduce the response time and automate the communication between the patient and the medical personnel and among the medical staff members. The idea is to have an application for automating the collection of data from wearables, the detection of problems and the ambulance call, which requires mechanisms for patient feedback (call back). In case of emergency, we will take advantage of the edge resources to deploy servers close to the patients, in order to provide video feed between the emergency team and then doctors at the hospital. This will enable doctors to give instructions to the personnel in the ambulance, which is not specialized, to deal with some emergencies such as an urgent surgery. In this case, for instance, a doctor has to monitor and guide the process over real-time 4K video.

The e-Health use case involves two types of services: the e-Health Monitoring and the e-Health Emergency services. The e-Health Monitoring service provides an infrastructure to detect emergency situations. Such service is always running and it is the one to which users with wearables subscribe. This service monitors the patients in order to process their vital signs and detect anomalies that allow doctors to act quickly in emergency cases. The e-Health Emergency service is instantiated on-demand when an emergency is detected and it is deployed in a restricted geographical location to deal with the specific emergency.

The two services share some of their functional components, so that the 5GT-VS can apply sharing strategies to optimize the usage of the infrastructure resources. In detail, the list of 5GT-VS features adopted in the e-Health use case is the following:

- Provisioning and termination of vertical service instances;
- Composition of vertical services;
- Management of network slice subnets shared among multiple end-to-end network slices;
- Programmable request of vertical services from vertical's applications;
- Service configuration.

5.7.3.1 Blueprint for e-Health services

The main fields of the e-Health monitoring and emergency VSBs are reported in Table 24 and Table 25 respectively, while the complete VSBs and VSDs of the e-Health use case are available in the Github repository at the following link:

`https://github.com/5g-transformer/5gt-vs/tree/master/blueprints/eHealth/`

TABLE 24: VSB FOR E-HEALTH MONITORING SERVICE

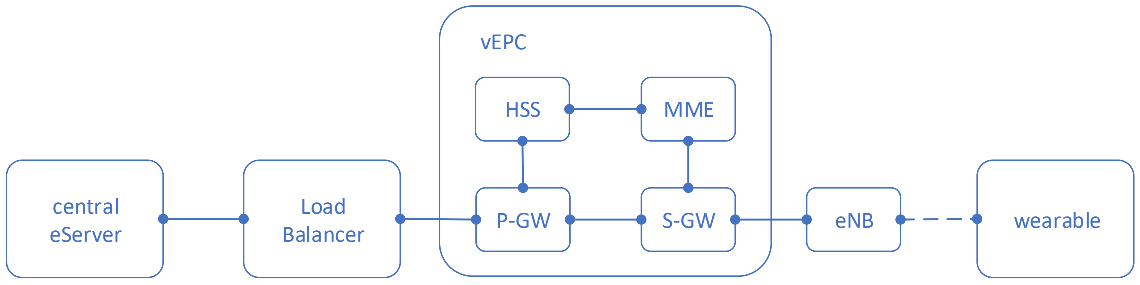
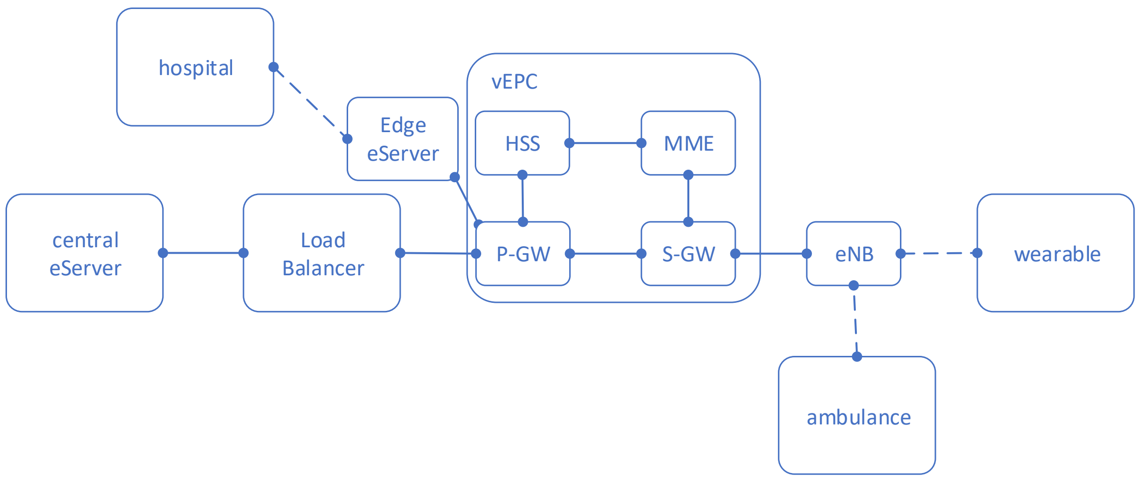
Field	Description
Service graph  vEPC HSS MME central eServer Load Balancer P-GW S-GW eNB wearable	
SST	mIoT
Atomic functional components	vEPC, which provides the mobile access to the service. The vEPC is composed of several subcomponents such as a P-GW, S-GW, MME, HSS. Load balancer, to dispatch the service requests in case of multiple instances of the central eServer. Central server, where the monitoring data from the wearables is processed.
Parameters	Geographical area: coordinates wearables Maximum Latency Minimum Bandwidth
SLA	Security level: high Priority level: medium Reliability level: high Maximum latency: low

TABLE 25: VSB FOR E-HEALTH EMERGENCY SERVICE

Field	Description
Service graph  hospital Edge eServer vEPC HSS MME central eServer Load Balancer P-GW S-GW eNB wearable ambulance	
SST	mIoT
Atomic functional components	All the components of the e-Health emergency service. A P-GW deployed close to the emergency, to connect the ambulance to the hospital. An edge server deployed close to the new P-GW, for local processing of monitoring data.
Parameters	Geographical area: coordinates wearables and ambulances MaxLatency MinBandwidth Number of available ambulances
SLA	Security level: high Priority level: high Reliability level: high Maximum latency: low

5.7.3.2 5GT-VS workflows for e-Health services

In the case of the eHealth services, the 5GT-VS operates following the same workflow as for any other service requested by a vertical. However, it is worth to mention that in terms of the arbitrator priorities, the eHealth emergency service will undertake the same procedure that is applied in the EVS service, since both are highly critical services dealing with people's safety.

Therefore, when both eHealth services, monitoring and emergency, are requested, the emergency one will have higher priority and thus, it will be processed first by the arbitrator. The monitoring service is deployed all the time to ensure continuous exchange of data between the users' wearables and the central eServer that process them. This exchange of data does require neither very stringent amount of resources nor very low latency, but the resources that are required must be provided to ensure a proper service functionality. The increase in the resources needed by the eHealth services occurs when an emergency takes place and the emergency service has to be deployed, triggered by the eHealth application itself through the 5GT-VS NBI. In this case, the Arbitrator has to notify the 5GT-SO about the high priority of the service to be deployed and the 5GT-SO has to provide the resources that are needed. When other emergencies occur in the same area, the resources of the already instantiated emergency service will be scaled up by the 5GT-SO and there is no need to deploy a completely new service. Thus, in this case the 5GT-SO scales up the resources and, when emergencies are resolved, the 5GT-SO scales down the unneeded resources.

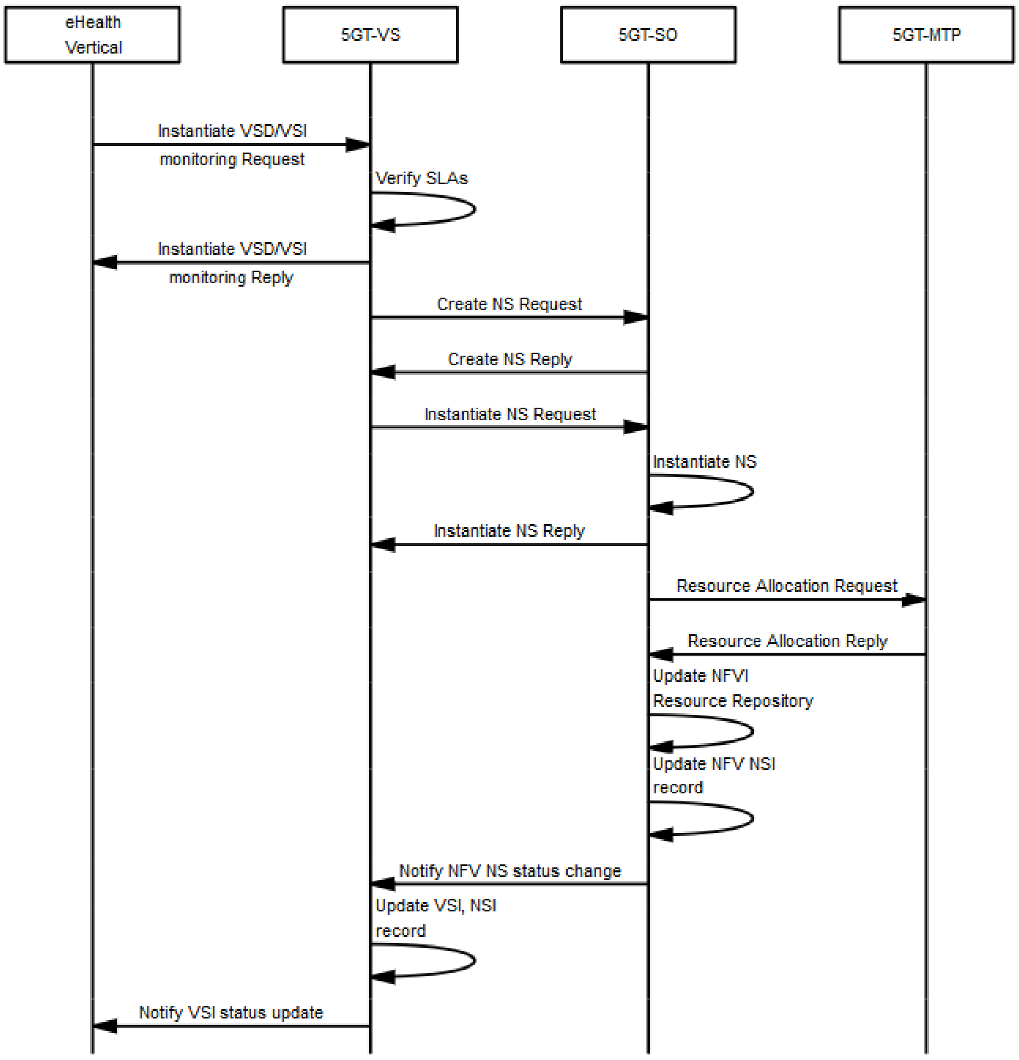


FIGURE 32: E-HEALTH USE CASE: MONITORING SERVICE INSTANTIATION

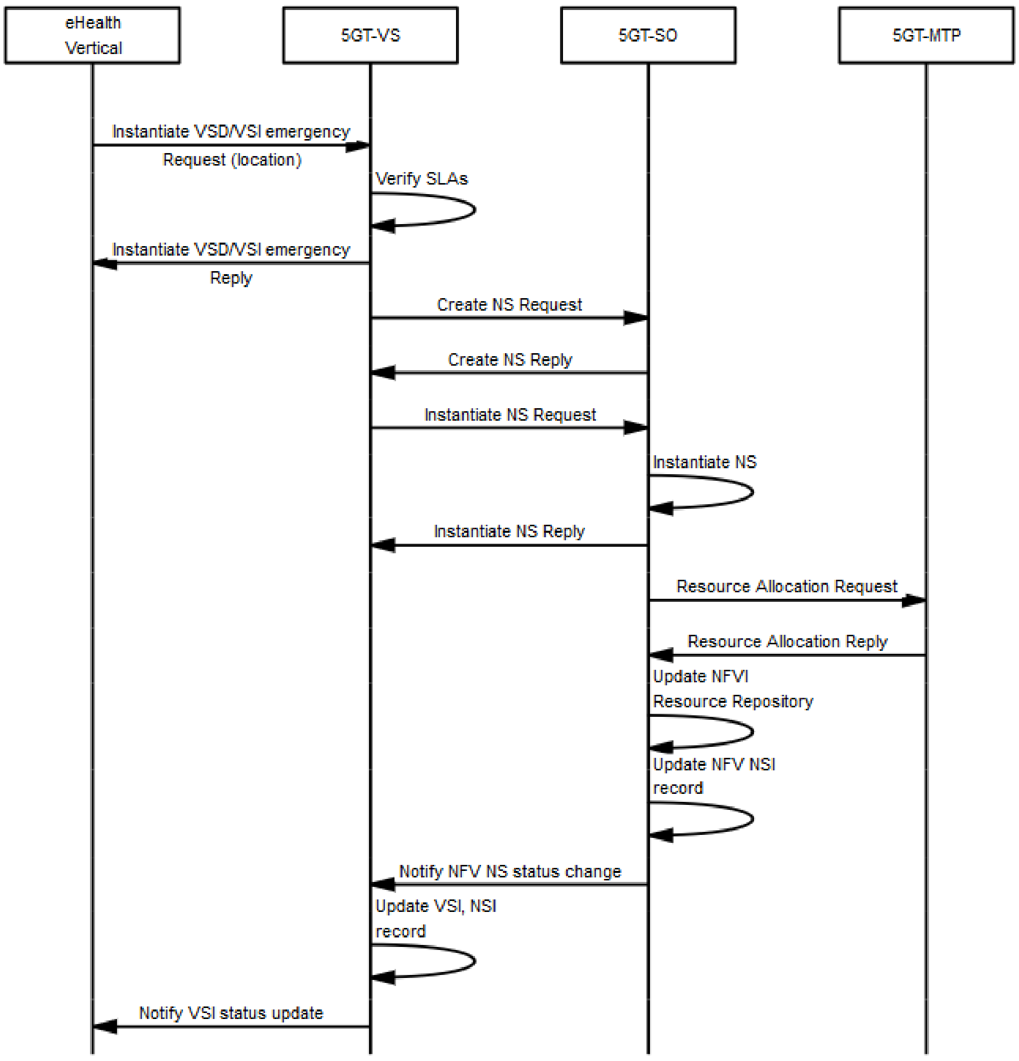


FIGURE 33: E-HEALTH USE CASE: EMERGENCY SERVICE INSTANTIATION

5.7.4 5GT-VS in e-Industry use case

The e-Industry use case intends to assess the role of 5G technologies and the impact of full digitalization and automation of industrial processes, as foreseen by the Industry 4.0 paradigm. It is anticipated that 5G connectivity will bring several benefits to such business area, including reduced installation and maintenance cost, agile and reconfigurable deployments, enhanced reliability and lower latency. In 5G-T we aim to demonstrate the advantages of moving the control of the production processes and robot capabilities into the cloud and leveraging on wireless connectivity to optimize the processes and increase deployment flexibility.

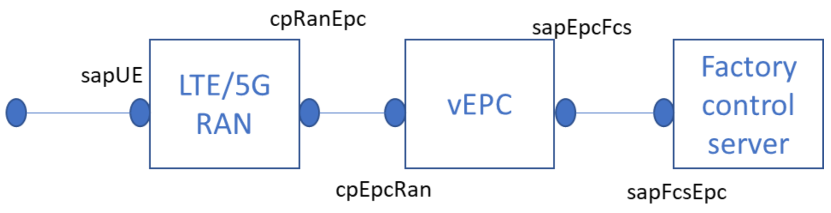
The 5GT-VS feature specifically involved in the provisioning and management of an e-Industry vertical service is the provisioning and termination of vertical service instances.

5.7.4.1 Blueprint for e-Industry service

The main fields of the e-Industry VSB are reported in Table 26, while the complete VSBs and VSDs of the e-Industry use case are available in the Github repository at the following link:

`https://github.com/5g-transformer/5gt-vs/tree/master/blueprints/eIndustry/`

TABLE 26: VSB FOR E-INDUSTRY SERVICE

Field	Description
Service graph	 cpRanEpc sapEpcFcs sapUE LTE/5G RAN vEPC Factory control server cpEpcRan sapFcsEpc
SST	URLLC
Atomic functional components	Mobile Communication Transport (eNB and vEPC) Factory control server: control plane of the factory system
Parameters	Number of End Devices (e.g., robots, sensors) Density of End Devices (per sq. meter) Geographical area End-to-end Latency (msec) Highest Acceptable jitter (msec) Bitrate required (Mbps)
SLA	End-to-end Latency < 10ms Bitrate needs per access point: 15-70 Mpbs Reliability: high (> 99%) Availability (related to coverage): high (> 99%)

5.7.4.2 5GT-VS workflows for e-Industry service

The workflow for the e-Industry service requires the instantiation of the related VSI and it is fully aligned with the Vertical Service Instantiation workflow reported in Section 5.5.3.

5.7.5 5GT-VS in MNO/MVNO use case

The role of the Mobile Network Operator (MNO) hosting a Mobile Virtual Network Operator (MVNO) is built on the offering of a Network Slice as a Service (NSaaS); for instance, the MNO would rely on network slicing combined to services like EPCaaS and IaaS in order to set up a virtual mobile network and provide connectivity network services to the MVNO. In addition, verticals can be seen as customers of an MNO or an MVNO.

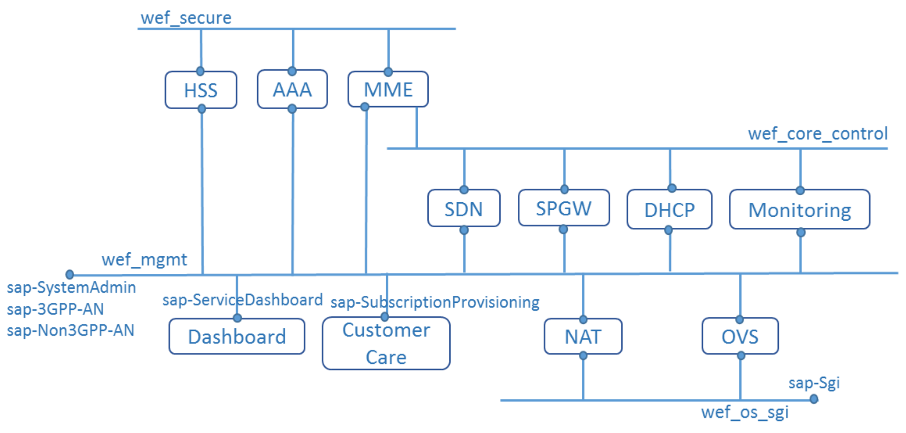
In 5G-T, we chose to focus on the vEPCaaS UC. The MNO/MVNO UC aims to demonstrate the deployment and the operation of a Network Slice with vEPC in “as a service” mode in order to build a MVNO service.

5.7.5.1 Blueprint for MNO/MVNO service

The main fields of the vEPC VSB are reported in Table 27, while the complete NSDs and VNFDs of the MNO/MVNO use case are available in the Github repository at the following link:

`https://github.com/5g-transformer/5gt-vs/tree/master/blueprints/mvno/`

TABLE 27: VSB FOR vEPC SERVICE

Field	Description
Service graph	 wef_secure HSS AAA MME wef_core_control SDN SPGW DHCP Monitoring wef_mgmt sap-SystemAdmin sap-3GPP-AN sap-Non3GPP-AN sap-ServiceDashboard sap-SubscriptionProvisioning Dashboard Customer Care NAT OVS sap-Sgi wef_os_sgi
SST	Not available
Atomic functional components	AAA (Authentication, Authorization and Accounting) Customer Care (subscription management and provisioning) Dashboard (displays connexions state and presence of UEs) DHCP (Dynamic Host Configuration Protocol) HSS (Home Subscriber Service) MME (Mobility Management Entity) Monitoring (monitoring of the WEF services) NAT (Network Address Translation) OVS (Open Virtual Switch) SDN (Software Defined Networking) controller SPGW (Serving and PDN Gateway)
Parameters	Number of users Number of sessions under processing Number of sessions set up per second

SLA	Sap: QoS class: gbr, priority (real time, high, medium, low) Availability: low (<95% is required) Reliability: medium (95%-99%) Traffic Density: 1000 users/km²

5.7.5.2 5GT-VS workflows for MNO/MVNO service

The workflows for the MNO/MVNO use case refer to the instantiation of the related service. In particular, the MNO/MVNO use case focuses on the NSaaS offer and it is demonstrated through a specific 5GT-VS implementation, described in section 5.8.3, built around the Network Slice Manager (NSM) component. The workflow for the creation of the Network Slice is reported in Figure 34.

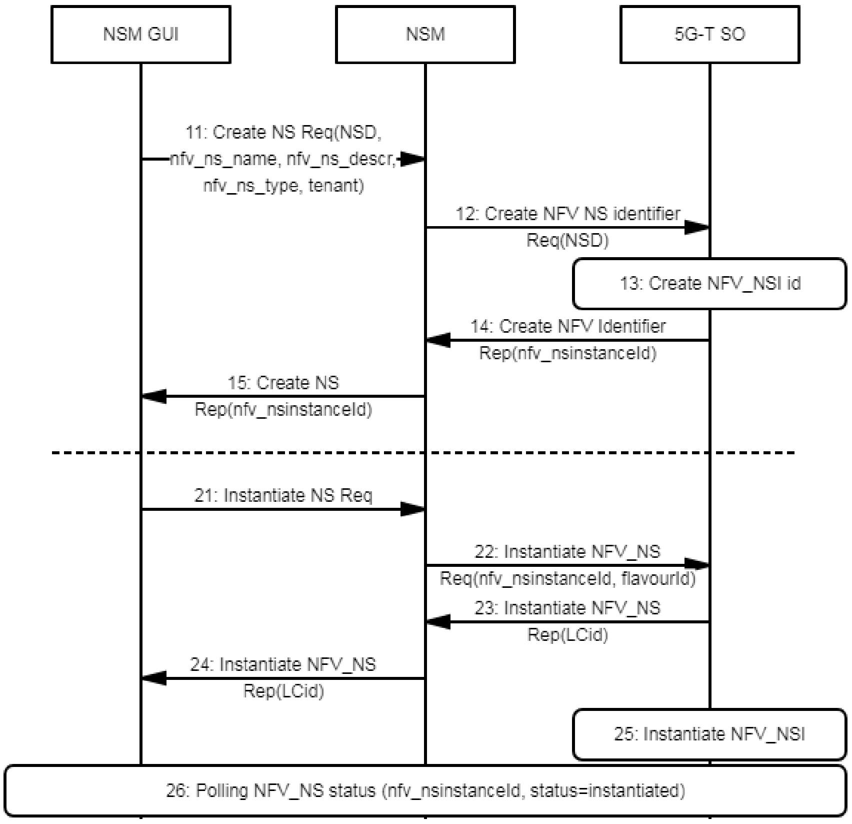

NSM GUI
NSM
5G-T SO
11: Create NS Req(NSD, nfv_ns_name, nfv_ns_descr, nfv_ns_type, tenant)
12: Create NFV NS identifier Req(NSD)
13: Create NFV_NSI id
14: Create NFV Identifier Rep(nfv_nsinstanceId)
15: Create NS Rep(nfv_nsinstanceId)
21: Instantiate NS Req
22: Instantiate NFV_NS Req(nfv_nsinstanceId, flavourId)
23: Instantiate NFV_NS Rep(LCid)
24: Instantiate NFV_NS Rep(LCid)
25: Instantiate NFV_NSI
26: Polling NFV_NS status (nfv_nsinstanceId, status=instantiated)

FIGURE 34: MNO/MVNO USE CASE: NETWORK SLICE INSTANTIATION

Contrary to the Vertical Service Instantiation workflow, the workflow above starts directly with the creation of the Network Slice by the customer through the NSM GUI. The NSM in this workflow is equivalent to the NSMF/NSSMF in the Vertical Service Instantiation workflow. Another difference between the two workflows is that, in the workflow above, Create NS and Instantiate NS requests going to the NSM are 2 different requests, while for the Vertical Service in the Figure 16 only the Instantiate NS request is called, which will then call the Create NFV NS and the Instantiate NFV NS requests.

5.8 Vertical Slicer prototypes

In 5G-T, the 5GT-VS has been implemented in three different prototypes. The first prototype provides the 5GT-VS reference implementation, integrating all the major features and functionalities required to demonstrate 5G-T use cases and thus constituting the major WP3 input towards WP5. The second prototype is a complementary one, still based on the overall 5GT-VS architecture, but more focused on testing the communication services based on the slice and service types (SST) defined by 3GPP, without addressing the specific requirements of 5G-T use cases. The third one focuses mostly on the NSMF/NSSMF components and targets the MNO/MVNO use case, with the NSaaS scenario. In this particular case, the mobile (virtual) operator is perfectly aware of the network requirements for the desired services and is able to request directly a network slice, without the need to choose from a range of simplified blueprints. The related 5GT-VS prototype offers an NBI to directly request and operate network slice instances, without implementing the functionalities related to the management of vertical services and their translation into network slices.

This section describes the three prototypes, listing their functionalities and providing the documentation in terms of user-guide and development-guide. For the latter, the focus is mostly on the aspects that allow to extend the system and to interact with it, in order to facilitate its adoption and integration within other contexts and communities external to 5G-T. In this sense, it is worth to mention that the 5GT-VS reference implementation has been already adopted as baseline for a network slicing solution in two 5G-PPP phase 2 projects (Slicenet[21] and blueSPACE[22]), while the components related to VSB/VSD catalogues and to the translation between VSDs and NSDs will be extended and integrated into the platform under development in the 5G-PPP phase 3 5G-EVE[23] project.

5.8.1 Vertical Slicer Reference Implementation

The 5GT-VS reference implementation, called SEBASTIAN (SErvice BAsed Slice Translation, Integration and AutomatioN), is an open source software prototype developed in Java and based on the Spring framework,[24] which provides all the major 5GT-VS functionalities required by 5G-T use cases. As such, it will be used as reference for all the WP5 demonstration activities that involves 5GT-VS features. The prototype is released as part of D3.4 [3] and it is available in the 5G-T public Github repository at the following link:

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

Further details about the structure of the repository, software licenses and dependencies, as well as installation and configuration guidelines are provided in D3.4 Release Notes [3].

5.8.1.1 Functionalities

The list of 5GT-VS functionalities and features implemented in SEBASTIAN is reported in Table 28, specifying the software releases (R1 for D3.2 [2] and R2 for D3.4 [3]) where each functionality is available.

[21] https://slicenet.eu/
[22] https://www.bluespace-5gppp.eu/
[23] https://www.5g-eve.eu/
[24] https://spring.io/

TABLE 28: SEBASTIAN FUNCTIONALITIES

Functionality	Description	Release
Administrative REST API	REST-based NBI for administrative functions: • Management of tenants, groups, SLAs and policies • On-boarding of VSBs • Retrieval of NSIs	R1/R2 (policies from R2)
Operational REST API	REST-based NBI for verticals: • Retrieval of VSBs • Definition of VSDs • Management of VSIs	R1/R2
Authentication and authorization	Mechanisms to authenticate the user and authorize REST API requests based on tenant's profile.	R1/R2
Web-based GUI for 5GT-VS administrators	Web GUI to access the functionalities exposed by the administrative REST API.	R1/R2 (policies from R2)
Web-based GUI for verticals	Web GUI to access the functionalities exposed by the operational REST API.	R1/R2
Web-based GUI for monitoring	Web GUI page to visualize monitoring data on a per-service basis.	R2
Translation between VSD and NSD	Implementation of 5GT-VS Translator.	R1/R2
Arbitration for 1:1 mapping between VSIs and NSIs	Basic implementation of 5GT-VS Arbitrator: • SLA verification based on resources consumed by each tenant • Mapping of VSI on single NSI (and vice versa)	R1
Arbitration for N:M mapping between VSIs and NSIs	Full-featured implementation of 5GT-VS Arbitrator: • SLA verification based on resources consumed by each tenant • Management of NSI composition • Mapping of multiple VSIs on single, composite NSIs • Sharing of NSSIs among multiple, composite NSIs • Arbitration among multiple vertical service instances belonging to the same tenant and with different priorities (including triggering of scaling actions) • Management of service isolation requirements	R2
Basic lifecycle management of "non-composite" services	Management of instantiation and termination procedures for "non-composite" services (i.e. VSIs mapped on NSIs without NSSIs)	R1/R2
Basic lifecycle management of "composite" services	Management of instantiation and termination procedures for "composite" services (i.e. VSIs mapped on NSIs including one or more NSSIs)	R2

Policy management	Management of policies to provide the 5GT-SO with directives about NFV-NSIs service instantiation	R2
Service scaling	Management of scaling procedures in VSI and NSI lifecycle	R2

5.8.1.2 User guide

The SEBASTIAN user guide is available on github, at the following link:

`https://github.com/5g-transformer/5gt-vs/tree/master/documentation/`

5.8.1.3 Software architecture

As mentioned, SEBASTIAN prototype is developed in Java and it is based on the Spring framework [25], adopting Apache Maven [26] as build automation tool. The software architecture, shown in Figure 35, follows a modular approach designed to simplify the introduction of future extensions in the system. The internal modules of SEBASTIAN implement Java interfaces to provide their functions towards other entities. A RabbitMQ[27]-based message bus (with messages encoded in JSON format) enables the asynchronous interactions among the different software modules. This choice allows an easy extension of such communications to external entities that may use the same message bus to interact with specific system components, following a subscription/notification approach. A PostgreSQL [28] database is used to implement SEBASTIAN catalogues and records, as well as to maintain its internal information persistency. The management of the databases within the software is wrapped through the usage of the Java Persistence API (JPA). On top of the system core, SEBASTIAN implements also the web-based GUI described in Section 5.8.1.2.

SEBASTIAN north-bound interface (NBI) is implemented through a set of REST controllers that implement the 5GT-VS REST APIs and invoke the corresponding services in the system core. The CRUD (Create-Read-Delete-Update) requests received at the management REST API (e.g. related to user groups, tenants, SLAs, or policies configuration and VSB creation) are elaborated through services that interact in read/write with the associated databases, modelled as JPA repositories. The REST requests for operational actions on VSIs are handled through the *VS operation service*, which in turn invokes the Engine dispatcher. From here, the processing of the requests is handled in an asynchronous manner, coordinating the exchange of ordered messages between the software modules cooperating for the execution of the given command. For example, a request to create a new VSI is initially processed at the corresponding *VSI Lifecycle (LC) Manager,* which invokes (synchronously) the *VSD-NSD Translator* and then the *Arbitrator*. The arbitrator implements algorithms embedding the logic of network slice sharing and multi-service arbitration. It provides in output the specification of the NSI to be deployed and, if needed, information about the NSSI to be re-used and if/how they must be scaled. Moreover, in case of arbitration decisions involving service

[25] https://spring.io/
[26] https://maven.apache.org/
[27]https://www.rabbitmq.com/
[28] https://www.postgresql.org

arbitration, the arbitrator output specifies also the actions (i.e. scaling or termination) that must be performed on existing VSIs/NSIs before instantiating the new one.

The requests for new NSIs are handled at the *NSI LC Manager*, which invokes the *NFVO Service* to request the creation of the corresponding NFV-NSI to the underlying 5GT-SO. If an NSI includes existing NSSIs, the NFV-NSI request will include the references to the IDs of the nested NFV-NSIs. In all the cases where actions on existing VSIs must be performed before instantiating the current one (and thus its NSI), a new instance of *VSI Group Coordinator* is created, with the responsibility of handling the ordered list of needed actions.

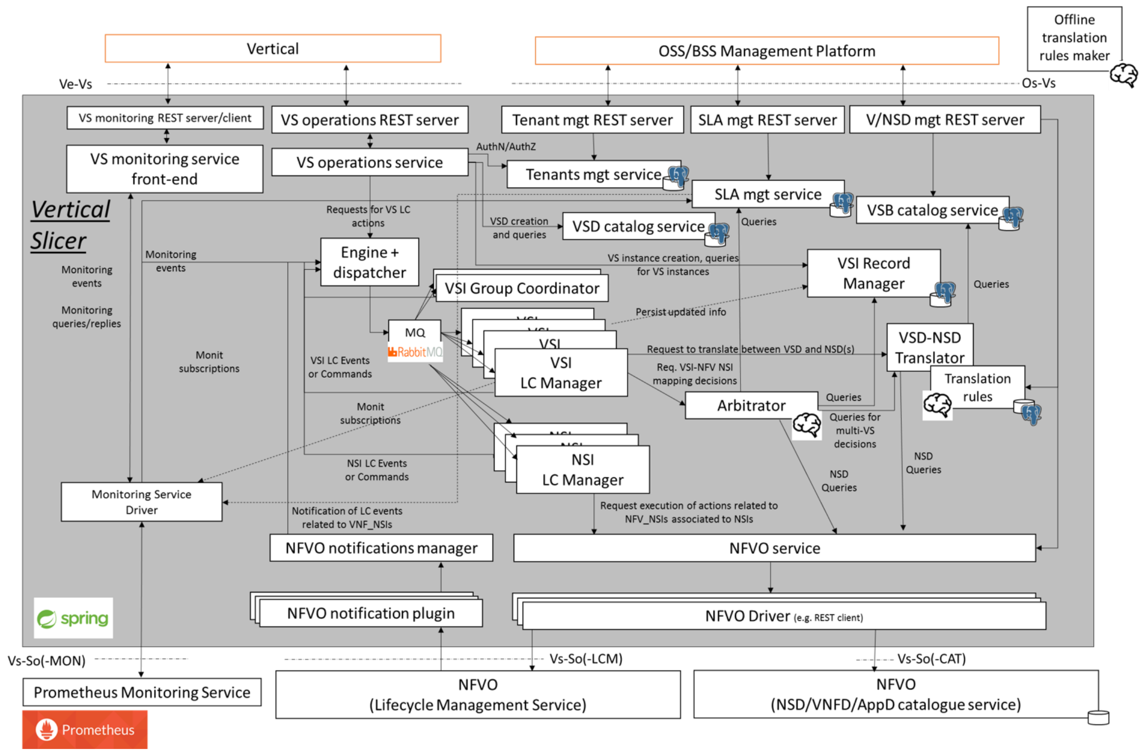

Offline translation rules maker
Vertical
OSS/BSS Management Platform
Ve-Vs
Os-Vs
VS monitoring REST server/client
VS operations REST server
Tenant mgt REST server
SLA mgt REST server
V/NSD mgt REST server
VS monitoring service front-end
VS operations service
AuthN/AuthZ
Tenants mgt service
SLA mgt service
VSB catalog service
Vertical Slicer
Requests for VS LC actions
VSD creation and queries
VSD catalog service
Queries
Monitoring events
Engine + dispatcher
VS instance creation, queries for VS instances
VSI Record Manager
VSI Group Coordinator
Persist updated info
MQ
RabbitMQ
Monitoring queries/replies
Monit subscriptions
VSI LC Events or Commands
VSI LC Manager
Request to translate between VSD and NSD(s)
Req. VSI-NFV NSI mapping decisions
VSD-NSD Translator
Translation rules
Arbitrator
Queries for multi-VS decisions
NSI LC Events or Commands
NSI LC Manager
NSD Queries
Monitoring Service Driver
Notification of LC events related to VNF_NSIs
Request execution of actions related to NFV_NSIs associated to NSIs
NFVO notifications manager
NFVO service
spring
NFVO notification plugin
NFVO Driver (e.g. REST client)
Vs-So(-MON)
Vs-So(-LCM)
Vs-So(-CAT)
Prometheus Monitoring Service
NFVO (Lifecycle Management Service)
NFVO (NSD/VNFD/AppD catalogue service)
Prometheus

FIGURE 35: SEBASTIAN SOFTWARE ARCHITECTURE

At the southbound interface (SBI), the interaction with the 5GT-SO is handled through a specific driver which embeds a REST-based implementation of the consumer side of the ETSI NFV IFA 013 interface, selected as reference NBI of the 5GT-SO. Different *NFVO drivers* can be supported, enabling the interaction with other NFVOs (see Section 5.8.1.3.2 about how to extend SEBASTIAN to integrate new NFVO drivers). The communication between the internal modules of the 5GT-VS and different NFVO drivers is mediated through the *NFVO Service*, which provides a single service access point and dispatches the incoming requests towards the correct driver.

5.8.1.3.1 Software components

The list of software components is fully described in deliverable D3.4 [3] - Release Notes. Here we report only the summary:

- Engine dispatcher
- VS operations service
- VSI LC Manager
- Arbitrator

- NSI LC Manager
- VSD/NSD Translator
- VSI Group Coordinator
- NFVO Service and NFVO Drivers

5.8.1.3.2 How to extend the Vertical Slicer prototype

The SEBASTIAN software development guide is available on github, at the following link:

`https://github.com/5g-transformer/5gt-vs/tree/master/documentation/`

5.8.1.4 Adoption of Vertical Slicer Reference Implementation in other projects

SEBASTIAN software code has been adopted as baseline for the prototypes of a network slicing platform in two 5G-PPP phase 2 projects (SliceNet[29] and blueSPACE[30]). Moreover, the VSB/VSD catalogues and the VSD-NSD Translator implemented in SEBASTIAN will be extended and integrated into the experimenters' portal that is currently under development in the 5G-PPP phase 3 5G-EVE[31] project. In the following, we provide some details about the adoption and extensions of SEBASTIAN in these three projects.

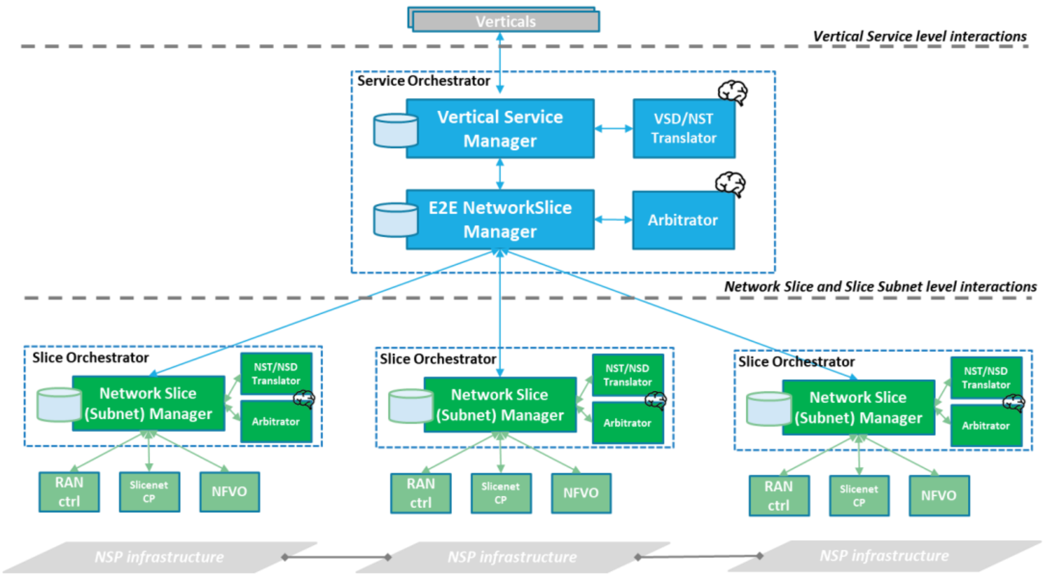


FIGURE 36: MULTI-DOMAIN NETWORK SLICING SOLUTION IN SLICENET PROJECT [41]

SliceNet is developing a solution for the management of network slices across multiple administrative domains, each of them handled by a Network Service Provider (NSP) that offers network slices to its own customers. The technical approach proposed by SliceNet is based on a hierarchical architecture, where a centralized Service Orchestrator coordinates the instantiation of multi-domain vertical services requesting the creation of network slices in different NSP domains. As shown in Figure 36, the SliceNet Service Orchestrator will integrate SEBASTIAN components for the VSI Manager, NSMF, Translator, and Arbitrator. In particular, this implementation will extend Arbitrator and NSMF to deal with the decomposition of a network slice in network slice subnets

[29] https://slicenet.eu/
[30] https://www.bluespace-5gppp.eu/
[31] https://www.5g-eve.eu/

belonging to different domains and will enhance the network slice information model to guarantee the compliance with the 3GPP Network slice Resource Model. The intra-domain Slice Orchestrators in SliceNet will include SEBASTIAN's NSSMF, Arbitrator, Translator, and NFVO service components. In this case, extensions will be implemented to enable the interaction with the OpenSource MANO (OSM) [32] NFVO (through a dedicated NFVO driver) and to manage the RAN side of the slice.

In the blueSPACE project, SEBASTIAN is used to implement a Network Slicing Manager. As shown in Figure 37, this entity operates on top of an NFVO that is capable of instantiating MEC applications and configuring a Spatial/Wavelength Division Multiplexing (SDM/WDM)-based fronthaul network. The major updates in SEBASTIAN code are related to the introduction of resource allocation algorithms for network slices, implemented in the Arbitrator component. Moreover, new VSBs and VSDs have been defined for the target blueSPACE services (a virtual EPC and a virtual CDN).

Finally, the 5G-EVE project is designing a portal that allows verticals to define experiments to be executed and validated in virtualized 5G environments customized for the requirements of their services. In this context, the VSB/VSD information models and catalogues included in SEBASTIAN are applied to model the vertical services of the experimenters. Moreover, the same approach is also extended to blueprints and descriptors for experiment tools (e.g. definition of traffic generators, network probes, traffic conditioners, etc.). An extended version of SEBASTIAN translator is then used to map the experiment descriptor (i.e. the composition of vertical service and experiment tools) into the <NSD; DF; IL> triple that describes the NFV-NSI to instantiate in the 5G-EVE multi-site facility.

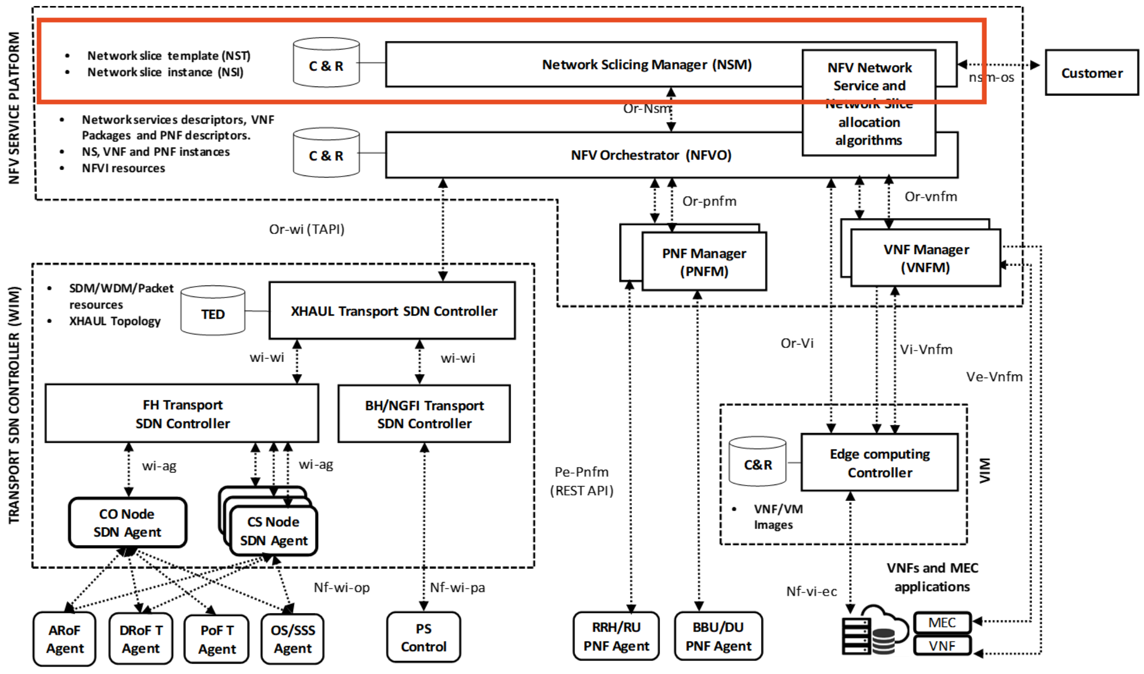


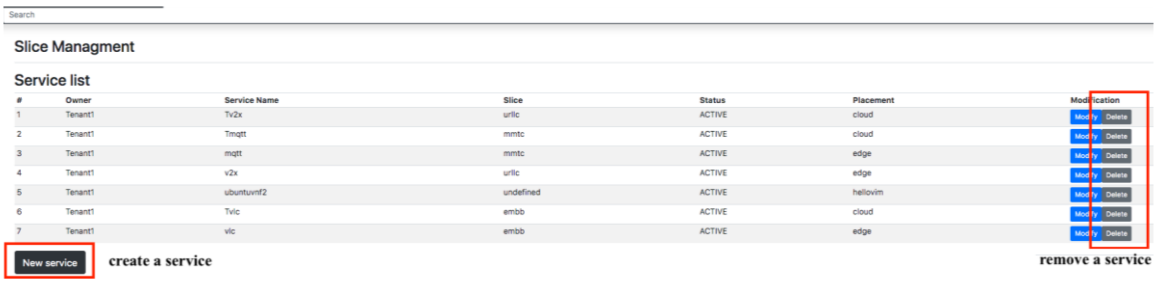
FIGURE 37: NETWORK SLICING MANAGER IN BLUESPACE PROJECT [42]

[32] https://osm.etsi.org

5.8.2 Vertical Slicer SST Implementation

The 5GT-VS SST implementation is part of a wider software release that includes both the 5GT-SO SST and 5GT-VS SST prototypes and it is called Joint Edge and Central Resource Slicer (JECRS). The JECRS framework comprises five key elements: an Operations, Administration, and Maintenance (OAM) portal, a resource coordinator, a template database, a configuration file maker and an infrastructure orchestrator communicator.

The JECRS procedures present the following workflow: The vertical (tenant) specifies the capability requirements of the vertical service in the request. The user requirement translator searches for an appropriate baseline configuration in the database. If no suitable template exists, the module translates the tenant's service request into the corresponding capacity requirement and adds the mapping to the database. After determining an appropriate mapping, the request arbitrator in the resource coordinator considers whether or not the request can be accommodated based on the free resources (an information provided by the infrastructure orchestrator communicator) and the Service-level Agreements (SLAs) of the tenants. If the request is accepted, the 2-tier resource allocator determines the resources to be provided by each tier of the MEC infrastructure.



Slice Managment

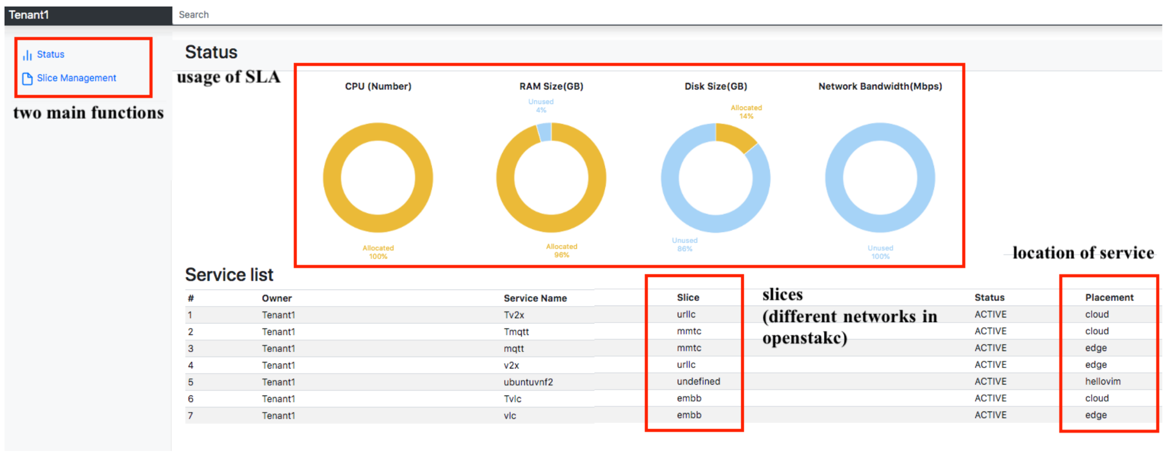
Service list

#	Owner	Service Name	Slice	Status	Placement	Modification
1	Tenant1	Tv2x	urllc	ACTIVE	cloud	Modify Delete
2	Tenant1	Tmqtt	mmtc	ACTIVE	cloud	Modify Delete
3	Tenant1	mqtt	mmtc	ACTIVE	edge	Modify Delete
4	Tenant1	v2x	urllc	ACTIVE	edge	Modify Delete
5	Tenant1	ubuntuvnf2	undefined	ACTIVE	hellovim	Modify Delete
6	Tenant1	Tvlc	embb	ACTIVE	cloud	Modify Delete
7	Tenant1	vlc	embb	ACTIVE	edge	Modify Delete

FIGURE 38: USER PORTAL FOR SLICES MANAGEMENT

The different components of 5GT-VS SST are the following:

- **User portal**: In the management webpage, the system administrator can see all the current slices/service instances for all the verticals/MVNOs (see Figure 38). Also, the slices status can be visualized (Figure 39). In particular, the SLA Configuration module is used to check the tenant SLA. The Slice/Service LCM module manages the lifecycle of each NSI and VSI with CRUD operations, as shown in Figure 40 and Figure 41.



#	Owner	Service Name	Slice	Status	Placement
1	Tenant1	Tv2x	urllc	ACTIVE	cloud
2	Tenant1	Tmqtt	mmtc	ACTIVE	cloud
3	Tenant1	mqtt	mmtc	ACTIVE	edge
4	Tenant1	v2x	urllc	ACTIVE	edge
5	Tenant1	ubuntuvnf2	undefined	ACTIVE	hellovim
6	Tenant1	Tvlc	embb	ACTIVE	cloud
7	Tenant1	vlc	embb	ACTIVE	edge

FIGURE 39: USER PORTAL FOR SLICES STATUS

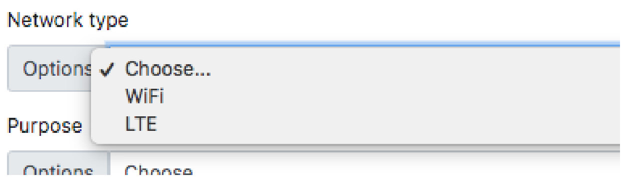


FIGURE 40: 5GT-VS SST NETWORK SELECTION

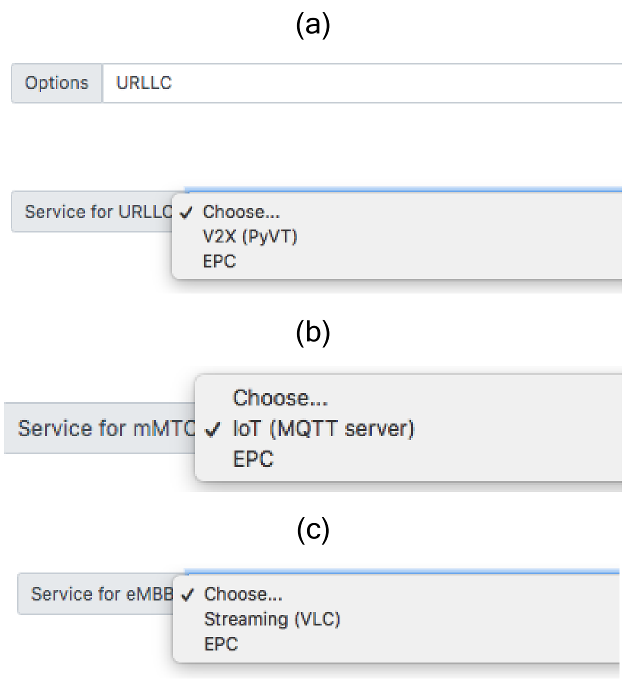


FIGURE 41: 5GT-VS SST SLICE TYPES: (A) URLLC; (B) MMTC; (C) EMBB

- **The Translator module** is responsible of transforming the high-level parameters provided in the request into VNFs, VLDs, and QoS parameters in the NSD. In particular, given a service j from tenant i (i.e. $r_{i,j}$), to facilitate the probing process for the computing resources, several break points are placed on the components of VNF(s) required by service j $VNF^{r_{i,j}}$ when running, and a count is made of the number of instructions processed in each case. The probing mechanism is deployed on a testing instance. On receipt of a new tenant request $r_{i,j}$, Upper-tier First with Latency-bounded Overprovisioning Prevention (UFLOP) searches for an appropriate baseline configuration in a database consisting of a set of default templates accumulated from previous probing rounds. If no suitable template can be found, UFLOP sets up $VNF^{ri,j}$in the testing instance and conducts a further probing process. In particular, UFLOP evaluates $VNF^{ri,j}$ for different traffic required by service j ($\lambda^{ri,j}$) as follow 0% of $\lambda^{ri,j}$, 25% of $\lambda^{ri,j}$, 50% of $\lambda^{ri,j}$, 75% of $\lambda^{ri,j}$, and 100% of $\lambda^{ri,j}$, respectively, in order to accurately estimate the relationship between the number of instructions needed and the actual value of $\lambda^{ri,j}$. Regarding the network resource probing process, UFLOP monitors the usage of the bandwidth at *TN*and *RN*during running and evaluates $VNF^{ri,j}$ using the same break points as those considered when probing the computing resource in order to determine the relationship between the bandwidth required and the actual value of $\lambda^{ri,j}$. The probing results (both computing and network) are then passed to the second step of UFLOP to determine the optimal allocation of the required resources which meets delay required by service j ($D^{ri,j}$).

- **VSB/VSD Catalogue module** is the database that stores the VSB and VSD information and is accessed by the translator module. In particular, the translator module will choose a proper VSD and forward it to the Slice/Service LCM module for further operations, e.g., creation. The resource allocation result is then passed to the configuration file maker, which creates a corresponding Network Service Descriptor (NSD) with a format dynamically determined by the infrastructure orchestrator. The NSD is then transferred via the infrastructure orchestrator communicator to the orchestrator (e.g., Tacker, OpenMANO, or Cloudify). On receipt of the NSD, the orchestrator sends a command to the VNFM to deploy the corresponding slice to the tenant such that they can build their service.
- **The Arbitrator module** decides on the creation of a new slice/service instance according to the resources available for the tenant, based on the SLA configuration. If the appropriate mappings (templates) are determined, the request arbitrator in the resource coordinator module evaluates whether or not the request can be accommodated based on an inspection of the free resource information provided by the infrastructure orchestrator communicator and the SLA of the tenant. If the request is accepted, the resource allocator in the resource coordinator module determines the resources to be provided by each tier of the 2-tier infrastructure. In Release 2, arbitration deals with situations where the tenant's request exceeds the SLA requirements, e.g. compute, network bandwidth. In this case, it will arbitrate among the services inside same or different slices, also providing suggestions to the tenant to recommend which service could be removed. This reflects the feature of Release 2 to support the dynamic combination of services.

5.8.3 Vertical Slicer MVNO Implementation

The third 5GT-VS implementation targets specifically the MNO/MVNO use case that offers NSaaS and, for this reason, it focuses mostly on the implementation of the Network Slice Manager (NSM) component. In fact, we assume that a MVNO does not need (or want) to adopt blueprints to define the required service, but prefers to specify directly the desired network slice, since he is perfectly aware of its requirements and its infrastructural details.

The NSM (Network Slice Manager) for the MNO/MVNO use case has been developed in Java and implements the NS LCM (Network Slice LifeCycle Management). As mentioned above, the MNO/MVNO UC does not use the whole 5GT-VS system. Indeed, in the MNO/MVNO UC, the M(V)NO already knows the network requirements for the services he wants. Thus, he can directly request a network slice, without the need to choose from a range of simplified blueprints as shown in Figure 42. So, this use case provides NSaaS (Network Slice as a Service) rather than offering vertical services. Since NSaaS are provided to the customer, we are at the level of the Network Slice Management Function (NSMF) and not of the Communication Service Management Function (CSMF), which is represented by the VSC (Vertical Slicer Customer), as shown in the Figure 42. Thus, we directly propose the Network Slice creation.

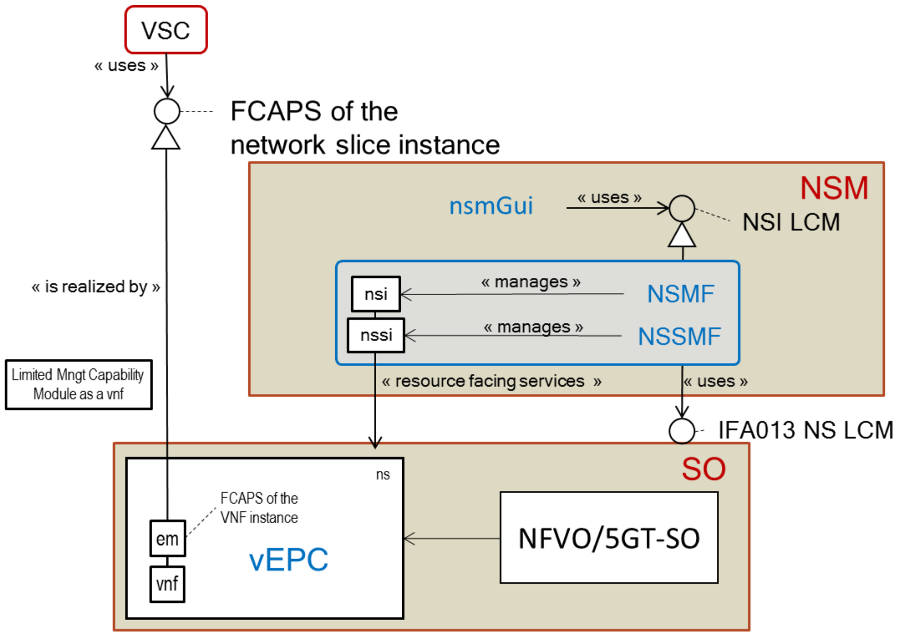


FIGURE 42: NETWORK SLICE AS A SERVICE

The NSM implementation is an alternative of the NSMF/NSSMF of the 5GT-VS to manage the lifecycle of Network Slices by exposing an NBI RESTful API with the separation of creation and activation operations.

5.8.3.1 Functionalities

The list of functionalities implemented in the NSM is reported in the Table 29. They will be available in the software release R2.

TABLE 29: NSM FUNCTIONALITIES

Functionality	Description
Administrative REST API	REST-based NBI for administrative functions: • Management of slice types and SLAs • Creation of NSTs (Network Slice Template)
Operational REST API	REST-based NBI for customers: • Definition of NSs based on slice type and SLA

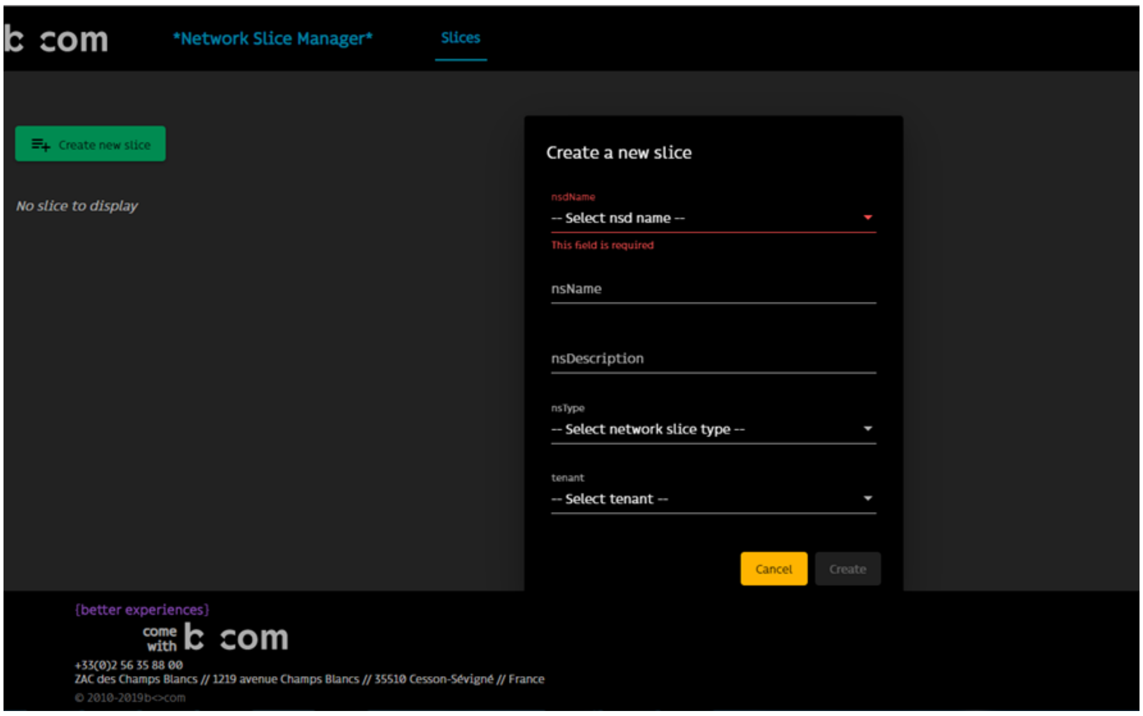
	• Management of NSIs (creation, allocation, deallocation, deletion)
Web-based GUI for customer	Web GUI (NSM GUI) to access the functionalities exposed by the operational REST API.
Translation between NST and (NSD, flavour)	Based on slice type and SLA, a script returns a NSD and a flavour.
Basic lifecycle management of "non-composite" services	Management of instantiation and termination procedures for "non-composite" services

5.8.3.2 Software architecture

The NSM prototype is developed in Java and use Apache Maven as build automation tool. The software architecture is shown in Figure 42. It follows a modular approach in order to be able to add new NFVO drivers and GUI.

An Apache Derby database is used to maintain the internal information persistency of the NSM prototype. The management of the databases within the software is wrapped through the usage of the Java Persistence API (JPA).

The NSM LCM northbound interface is implemented through a REST API. The NSM GUI, as shown in Figure 43, allows the customer to deploy a Network Slice by choosing an NSD, a slice type (eMBB, URLLC, mMTC) and a tenant where deploying the network slice.



No slice to display

Create a new slice

nsdName
-- Select nsd name --
This field is required

nsName

nsDescription

nsType
-- Select network slice type --

tenant
-- Select tenant --

+33(0)2 56 35 88 00
ZAC des Champs Blancs // 1219 avenue Champs Blancs // 35510 Cesson-Sévigné // France
© 2010-2019b<>com

FIGURE 43: NSM GUI: NETWORK SLICE CREATION

The northbound REST API implements the following functions:

- allocateNsi
- deallocateNsi
- activateNsi
- deactivateNsi

- getOperationStatus
- listAllNetworkSliceInstances
- getNsds
- getTenants.

At the southbound interface, the interaction with the 5GT-SO is handled through a specific driver.

The southbound REST API implements the following functions:

- createNsIdentifier
- instantiateNs
- terminateNs
- deleteNs
- getOperationStatus
- queryNs
- queryNsd.

5.8.3.2.1 Addition of a NFVO driver

In the context of 5GT, the default NFVO driver used interacts with the 5GT-SO but it can also be used to directly interact with orchestrators like Open Source Mano or Cloudify. New drivers can be added in the Maven project. Only one at a time can be used. The NFVO driver used must be specified in a properties file located in the business implementation part. New drivers must implement the OrchestratorManager Java interface.

5.9 Annex I: Reference Architectures

A few Standard Development Organizations (SDOs) and fora are contributing to the design of management systems for 5G that have many common design principles: (i) flexibility, (ii) adaptability, and (iii) cost-efficiency. Despite the many common design objectives across these working groups (as presented below), there are various relevant architectural concepts (e.g., slicing, federation/multi-domain, edge computing) that are specific to individual groups. In this context, 5G-T strives to bridge the gaps across such heterogeneous ecosystem in order to harmonically integrate these concepts under a single architecture.

This section introduces ongoing architectural work at 3GPP and ETSI NFV and MEC that fulfills two objectives:

- Served as basis to define the 5G-T architecture;
- Set the framework in which 5G- TRANSFORMER must be integrated to maximize its impact, i.e., by seeking as much as possible compliance with what is already defined.

Despite the fact that there are other organizations discussing about slicing and architectural concepts related with 5G-T, we focus on the ones below because they are those ones with a more complete definition of their architecture and building blocks, and so, they go well beyond requirements and high-level concepts.

5.9.1 3GPP

The most relevant working groups inside 3GPP related to 5G-T are SA2 (Architecture) and SA5 (Telecom management).

5.9.1.1 3GPP SA2

The 3GPP SA2 Working Group (WG), responsible for overall system architecture, is currently working on specifying the 5G Core (5GC) architecture with network slicing being a main feature of 5GC. Technical Specification (TS) 23.501 [9] defines Stage-2 system architecture for the 5G system, which includes network slicing. Figure 44 depicts an example of network slicing from 3GPP's perspective.

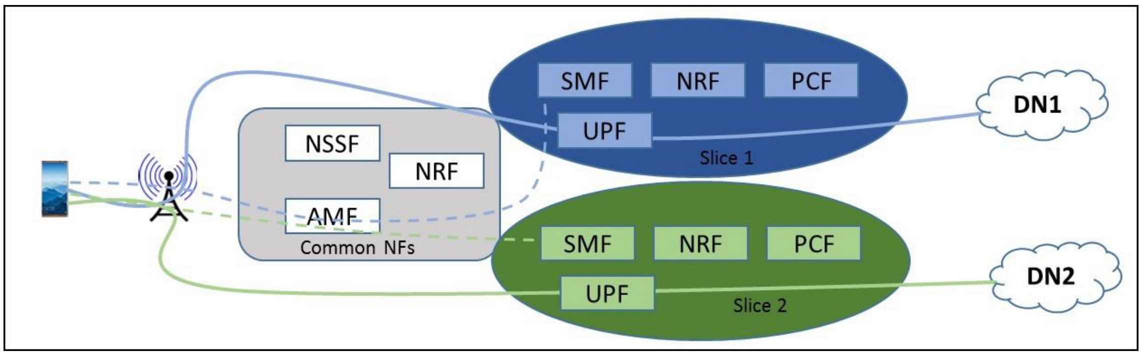


FIGURE 44: EXAMPLE OF NETWORK SLICES FROM 3GPP SA2 PERSPECTIVE

A network slice is viewed as a logical end-to-end network that can be dynamically created. A given User Equipment (UE) may access to multiple slices over the same Access Network (e.g. over the same radio interface). Each slice may serve a particular service type with agreed SLA. In the following, we provide highlights of 3GPP network slicing as being defined in TS 23.501 [9] in SA2.

A network slice is defined within a Public Land Mobile Network (PLMN) and includes the Core Network Control Plane and User Plane Network Functions as well as the 5G Access Network (AN). The 5G Access Network may be a Next Generation (NG) Radio Access Network described in 3GPP TS 38.300 [13], or a non-3GPP Access Network.

TS 23.501 [14] defines Network Function, Slice, and Slice Instance as follows:

- Network Function: A 3GPP adopted or 3GPP defined processing function in a network, which has defined functional behavior and 3GPP defined interfaces. (Note: A network function can be implemented either as a network element on a dedicated hardware, as a software instance running on a dedicated hardware, or as a virtualized function instantiated on an appropriate platform, e.g. on a cloud infrastructure.);
- Network Slice: A logical network that provides specific network capabilities and network characteristics;
- Network Slice instance: A set of network function instances and the required resources (e.g. compute, storage and networking resources) which form a deployed network slice.

5.9.1.2 3GPP SA5

3GPP SA5 Working Group (WG) is the 3GPP telecom management working group. 3GPP SA5 specifies the requirements, architecture, and solutions for provisioning and management of the network, including Radio Access Network (RAN) and Core Network (CN) and its services.

SA5 has completed a study on management and orchestration on network slicing (3GPP 28.801 [8]) and completed the normative specification work for release 15 based on this study. In particular, this work includes:

- Network slice concepts, use cases, and requirements (3GPP 28.530 [10]);
- Provisioning of network slicing for 5G networks and services (3GPP 28.531 [14]);
- Assurance data and performance management for 5G networks and network slicing;
- Fault supervision for 5G network and network slicing.

The following description highlights management and orchestration aspects of network slicing in 3GPP 28.801 [8]. However, these may be updated in the SA5 normative specifications based on the ongoing development of the SA2 technical specifications.

- General management and orchestration aspects of network slicing defined in 3GPP 28.801 [8]. Based on 3GPP 23.501 [9], SA5 has defined different management aspects for network slices in 3GPP 28.801 [8], as listed below:
    - Managing a complete Network Slice Instance (NSI) is not only managing all the functionalities but also the resources necessary to support certain set of communication services.
    - An NSI not only contains Network Functions (NFs), e.g belonging to AN and CN, but also the connectivity between the NFs. If the NFs are interconnected, the 3GPP management system contains the information relevant to connections between these NFs such as topology of connections, individual link requirements (e.g. QoS attributes), etc. For the part of the Transport Network (TN) supporting connectivity between the NFs, the 3GPP management system provides link requirements to the management system that handles the part of the TN supporting connectivity between the NFs.
    - NSI can be composed of network slice subnets of physical network functions and/or virtualized network functions.
- Network Slice Instance lifecycle management. 3GPP 28.801 [8] has introduced the network slice instance lifecycle management as depicted below in Figure 45, considering it independent of the network service instance that is using the network slice instance. Typically, a network slice instance is designed (preparation phase), then it is instantiated (Instantiation, Configuration and Activation phase), then it is operated (Run-time phase), and finally it may be decommissioned when the slice is no longer needed (Decommissioning phase). 3GPP 28.801 [8] introduces 3 management logical functions:
    - Communication Service Management Function (CSMF): Responsible for translating the communication service-related requirement to network slice related requirements.
    - Network Slice Management Function (NSMF): Responsible for management and orchestration of NSI and derive network slice subnet related requirements from network slice related requirements.
    - Network Slice Subnet Management Function (NSSMF): Responsible for management and orchestration of network slice subnet instances (NSSI).

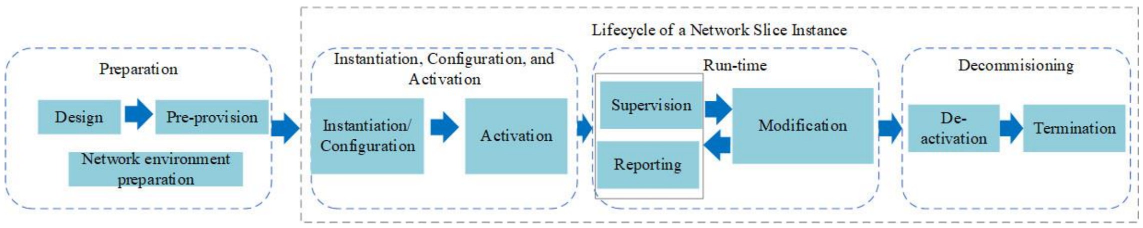

Lifecycle of a Network Slice Instance
Preparation
Design
Pre-provision
Network environment preparation
Instantiation, Configuration, and Activation
Instantiation/ Configuration
Activation
Run-time
Supervision
Reporting
Modification
Decommisioning
De-activation
Termination

FIGURE 45: 3GPP VIEW ON NETWORK SLICE INSTANCE LIFECYCLE

5.9.2 ETSI

5.9.2.1 ETSI NFV

Work is also ongoing inside ETSI NFV on how the NFV architecture in general, but more specifically, the ETSI MANO components can support network slicing.

In this respect, the Evolution and Ecosystem (EVE) working group has carried out activities that map NFV and 3GPP network slicing concepts (see EVE012 [30]). On the one hand, ETSI NFV EVE012 establishes the correspondence between a network slice (3GPP) and a network service (ETSI NFV). There, ETSI describes that an NFV Network Service (NFV-NS) can be regarded as a resource-centric view of a network slice, for the cases where an NSI would contain at least one virtualized network function. According to 3GPP 28.801 [8], an NSSI can be shared by multiple NSIs. The virtualized resources for the slice subnet and their connectivity to physical resources can thus be represented by the nested network service concept defined in ETSI NFV-IFA 014 [28] (right hand side of Figure 46), or one or more VNFs and PNFs directly attached to the Network Service used by the network slice. Figure 46 illustrates the relationship between 3GPP's network slice and ETSI NFV network service.

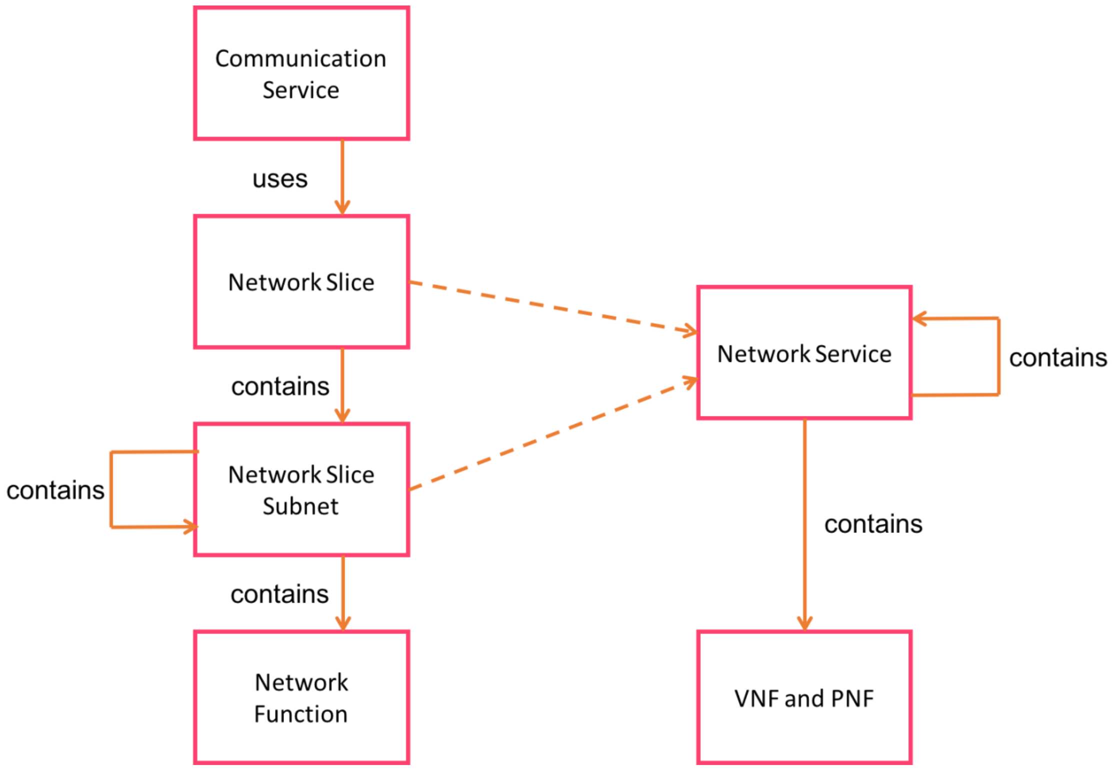


FIGURE 46: RELATION BETWEEN 3GPP AND ETSI INFORMATION MODELS (FROM [30])

As mentioned before, 3GPP 28.801 [8] identifies three management functions related to network slicing management: Communication Service Management Function (CSMF), Network Slice Management Function (NSMF), and Network Slice Subnet Management Function (NSSMF).

As shown in Figure 47, the Os-Ma-nfvo reference point can be used for the interaction between 3GPP slicing related management functions and NFV MANO. To properly interface with NFV MANO, the NSMF and/or NSSMF need to determine the type of network service or set of network services, VNF and PNF that can support the resource requirements for a NSI or NSSI, and whether new instances of these network services, VNFs, and the connectivity to the PNFs need to be created or existing instances can be reused.

From a resource management viewpoint, NSI can be mapped to an instance of a simple or composite network service instance or to a concatenation of such network service instances. From a resource management viewpoint, different NSIs can use instances of the same type of network service (i.e. they are instantiated from the same Network Service Descriptor or NSD) with the same or different deployment flavours. Alternatively, different NSIs can use instances of different types of network services. The first approach can be used if the NSIs share the same types of network functions (or a large common subset) but differ in terms of the performance expected from these network functions (and from the virtual links connecting them) and/or the number of instances to be deployed for each of them. If slices differ more significantly, mapping to different network services, each with its own NSD, can be considered. The same mapping principles might apply to NSSIs.

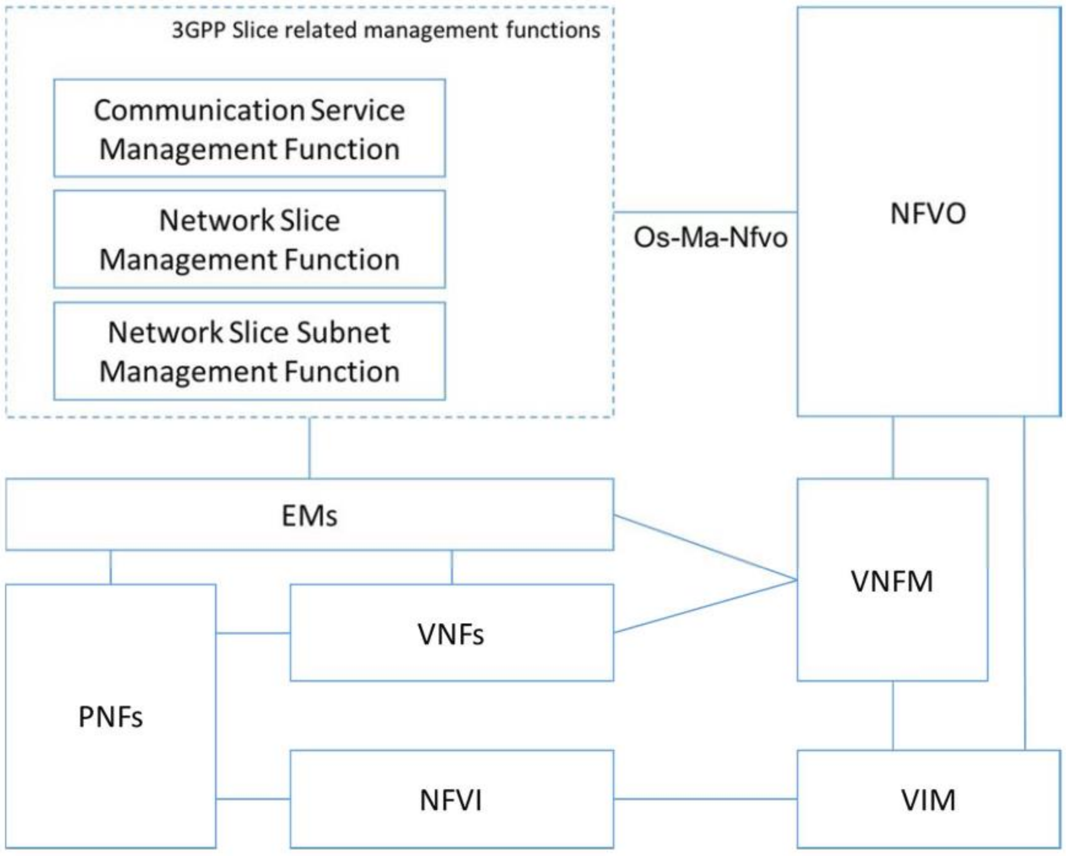

3GPP Slice related management functions
Communication Service Management Function
Network Slice Management Function
Network Slice Subnet Management Function
Os-Ma-Nfvo
NFVO
EMs
VNFM
VNFs
PNFs
NFVI
VIM

FIGURE 47: NETWORK SLICE MANAGEMENT IN AN NFV FRAMEWORK (FROM [30])

Also, as described before, 3GPP 28.801 [8] describes the lifecycle of a network slice, which is comprised of the four following phases: (i) Preparation; (ii) Instantiation, configuration, and activation; (iii) Run-time; and (iv) Decommissioning.

The preparation phase includes the creation and verification of Network Slice Template(s) (NST(s)). From an NFV perspective, the resource requirement for a NST can be realized by one or more existing NSDs that have been previously on-boarded on the NFVO. The creation of a new NST can lead to requiring update of an existing NSD or generation of a new NSD followed by on-boarding the new NSD if the slice requirements do not map to an already on-boarded NSD. Indeed, the NFV-NS for the multiple NSIs may be instantiated with the same NSD, in order to deliver exactly the same optimizations and features but dedicated to different enterprise customers. However, a network slice intended to support totally new customer facing services is likely to require a new NS and thus the generation of a new NSD. The network slice instantiation step in the second phase triggers the instantiation of the underlying NSs. NFV-MANO functions are only involved in the network slice configuration phase if the configuration of virtualisation-related parameters is required on one or more of the constituent VNF instances. Configuration of the network applications embedded in the constituent network functions involves the NSMF or NSSMF and/or other parts of the OSS/BSS, and the element managers (if any) associated to these functions. NFV-MANO functions can be triggered during the network slice activation step. If explicit activation of VNFs is required, the NSMF or the NSSMF can change the operational state of those VNFs through an Update NFV-NS operation defined in ETSI NFV-IFA 013 [11]. The involvement of NFV-MANO in the run-time phase is limited to the operations related to the performance management, fault management, and lifecycle management of virtualised resources (e.g. scaling an underlying NFV-NS to expand a NSI). The decommissioning phase triggers the termination of the underlying network service instances.

Additionally, and given the multiple administrative boundaries of the 5G-T architecture, the Interfaces and Architecture (IFA) working group is of particular interest for our project. ETSI NFV IFA028 [15] reports on potential architecture options to support the offering of NFV MANO services across multiple administrative domain. NFV-MANO services can be offered and consumed by different organizations, e.g. by different network operators or by different departments within the same network operator. Administrative domains as defined in ETSI NFV IFA010 [16] can be mapped to such different organizations. Examples of use cases for NFV-MANO service offerings across multiple administrative domains are described in ETSI NFV 001 [17]. Furthermore, ETSI NFV IFA022 [18] reports on the functional architecture needed to provision and manage multi-site network services. To this end, a set of multi-site use cases are studied.

Furthermore, compliance with widely accepted standards of the 5G-T architecture is also relevant to maximize its impact. Therefore, in a more general architectural context than that defined by the previous documents (which focus on specific issues) the interfaces already defined in ETSI NFV MANO are also relevant:

- IFA013 [11] defines the interfaces supported over the Os-Ma-nfvo reference point of the NFV MANO architectural framework as well as the information elements exchanged over those interfaces;

- IFA005 [12] defines the interfaces supported over the Or-Vi reference point of the NFV MANO architectural framework as well as the information elements exchanged over those interfaces;
- IFA006 [33] defines the interfaces supported over the Vi-Vnfm reference point of the NFV MANO architectural framework as well as the information elements exchanged over those interfaces;

5.9.2.2 ETSI MEC

Multi-Access Edge Computing (MEC) is one of the key concepts for fulfilling some of the requirements of vertical services, and therefore its integration in the 5G-T architecture is nexus in its design. MEC and its integration in an NFV context was studied in ETSI MEC017 [19] document. A reference architecture is provided with the following key observations:

- The mobile edge platform is deployed as a VNF and, therefore. the procedures defined by ETSI NFV for this means are used;
- ETSI NFV MANO sees mobile edge applications as regular VNFs allowing for reuse of ETSI MANO functionality (with perhaps some extensions);
- The virtualization infrastructure is deployed as an NFVI and its virtualized resources are managed by the VIM. For this purpose, the procedures defined by ETSI NFV infrastructure specifications, i.e. ETSI NFV INF003 [20], ETSI NFV INF004 [21], and ETSI NFV INF005 [22] can be used.

The concept of network slicing and its impact is being studied in ETSI MEC024 [47], to which the 5G-TRANSFORMER project contributed significantly. The report is expected to be published in July 2019 and covers several aspects:

- An overview of the network slicing concepts as defined by distinct standard organizations (i.e., NGMN, ONF, 3GPP, ETSI NFV). ETSI MEC gives priority to 3GPP and ETSI NFV concepts;
- The deployment scenario of MEC in NFV (ETSI MEC017 [19]) is considered as a baseline;
- The impact of 3GPP and ETSI NFV network slicing concepts is assessed for the various ETSI MEC components;
- No criticalities are identified on the data plane. However, the control plane requires some extensions to properly support the network slicing concept and SLA enforcement in the different network slices;
- The MEC Platform Manager (MEPM-V) and the MEC Application Orchestrator (MEAO) are part of the ETSI MANO and are the components mainly impacted by the network slicing concept when applied to ETSI MEC.

5.10 Annex II: Vertical Slicer NBI Operations

The operations at the 5GT-VS NBI, see Section 5.3.4, are listed in the following subsections together with their parameters.

5.10.1 Ve-Vs reference point

5.10.1.1 Query VS blueprints

This operation is described in Section 5.3.4.1.1, see Table 11.

5.10.1.2 Create VSD

This operation allows a vertical to create a new VSD that is added to the 5GT-VS catalogue. The VSD is then used by the vertical to request the instantiation of a new vertical service.

The Create VSD messages are specified in Table 30.

TABLE 30: CREATE VSD MESSAGES

Message	Direction	Description	Parameters
Create VSD request	Vertical → 5GT-VS	Request to create a new VSD.	• VSD (format specified in Section 5.4.2). • Vertical ID. • Is Public (set to true if the VSD is visible to other verticals).
Create VSD response	5GT-VS → Vertical	Response with the result of the operation.	• Result Code. • VSD ID.

5.10.1.3 Query VSD

This operation allows a vertical to retrieve one or more VSDs from the 5GT-VS catalogue, based on the given filter. A vertical can retrieve only the VSDs that have been created by the vertical itself or the VSDs that are declared as public.

The Query VSD messages are specified in Table 31.

TABLE 31: QUERY VSD MESSAGES

Message	Direction	Description	Parameters
Query VSD request	Vertical → 5GT-VS	Request to retrieve one or more VSDs matching the given filter.	• Filter (e.g. VSD ID). • Vertical ID.
Query VSD response	5GT-VS → Vertical	Response including the details of the requested VSDs.	• Result Code. • List<VSD> (format specified in Section 5.4.2)

5.10.1.4 Update VSD

This operation allows a vertical to update a VSD from the 5GT-VS catalogue, providing their IDs. A vertical is allowed to update only the VSDs that have been created by the vertical itself.

The Update VSD messages are specified in Table 32. The updated VSD has a new VSD ID, the previous version is still accessible with the old VSD ID.

TABLE 32: UPDATE VSD MESSAGES

Message	Direction	Description	Parameters
Update VSD request	Vertical → 5GT-VS	Request to update a VSD	• VSD (format specified in Section 5.4.2). • Vertical ID. • VSD ID.

			• Is Public (set to true if the VSD is visible to other verticals).
Update VSD response	5GT-VS → Vertical	Response with the result of the operation.	• Result Code. • VSD ID.

5.10.1.5 Delete VSD

This operation allows a vertical to delete one or more VSDs from the 5GT-VS catalogue, providing their IDs. A vertical is allowed to delete only the VSDs that have been created by the vertical itself.

The Delete VSD messages are specified in Table 33.

TABLE 33: DELETE VSD MESSAGES

Message	Direction	Description	Parameters
Delete VSD request	Vertical → 5GT-VS	Request to delete one or more VSDs with the given IDs.	• List<VSD ID>. • Vertical ID.
Delete VSD response	5GT-VS → Vertical	Response with the result of the operation.	• Result Code.

5.10.1.6 Instantiate VSI

This operation allows a vertical to instantiate a new VSI, given its VSD. Upon the reception of an Instantiate VSI request, the 5GT-VS creates a new entry in its repository, generating a unique VSI ID that is immediately returned to the vertical. This VSI ID can then be used by the vertical to retrieve information about the status and the characteristics of the VSI, using the Query VSI primitive.

The procedures to actually instantiate the VSI are then handled by the 5GT-VS in an asynchronous manner and notifications about the instantiation result, as well as any other relevant lifecycle event, are sent later to the vertical, if a suitable URL was provided in the instantiation request. The corresponding workflow is described in Section 5.5.3.

The Instantiate VSI messages are specified in Table 34.

TABLE 34: INSTANTIATE VSI MESSAGES

Message	Direction	Description	Parameters
Instantiate VSI request	Vertical → 5GT-VS	Request to instantiate a new VSI based on the specified VSD.	• VSI name. • VSI description. • VSD ID. • Vertical ID. • Notification URL (URL where the vertical wants to receive notifications about the lifecycle or failure events of the VSI). • Additional parameters (e.g. configuration

			parameters provided by the vertical, sharing permissions).
Instantiate VSI response	5GT-VS → Vertical	Response with the preliminary result of the operation and, if successful, the VSI ID.	• Result Code. • VSI ID.

5.10.1.7 Query VSI

This operation allows a vertical to retrieve the information about an existing VSI previously requested by the vertical itself. The target VSI is identified through its unique ID, as returned by the 5GT-VS in the Instantiate VSI Response. This method can be used by the vertical to get information about the current status of the VSI, its characteristics (through a reference to the current VSD) and the value assigned by the system to the output parameters, when they become available.

The Query VSI messages are specified in Table 35.

TABLE 35: QUERY VSI MESSAGES

Message	Direction	Description	Parameters
Query VSI request	Vertical → 5GT-VS	Request to retrieve information about the VSI matching the given ID.	• VSI ID. • Vertical ID.
Query VSI response	5GT-VS → Vertical	Response including the details of the requested VSI.	• Result Code. • VSI ID. • VSI name. • VSI description. • VSD ID. • VSI status (i.e. instantiating, instantiated, under modification, terminating, terminated, failed). • External interconnections (SAPs including the values assigned by the system to IP addresses, VLAN IDs, etc.). • Internal interconnections (CPs including the values assigned by the system to IP addresses, VLAN IDs, etc.).

5.10.1.8 Terminate VSI

This operation allows a vertical to terminate an existing VSI previously requested by the vertical itself. The target VSI is identified through its unique ID, as returned by the 5GT-VS in the Instantiate VSI Response. The 5GT-VS returns synchronously a preliminary result of the operation, but the procedure to actually terminate the VSI is handled by the 5GT-VS in an asynchronous manner. A notification about the termination result is sent to the vertical when the termination process is finished, if a suitable URL was provided by the vertical in the instantiation request. The corresponding workflow is described in Section 5.5.5.

The Terminate VSI messages are specified in Table 36.

TABLE 36: TERMINATE VSI MESSAGES

Message	Direction	Description	Parameters
Terminate VSI request	Vertical → 5GT-VS	Request to terminate a VSI matching the given ID.	• VSI ID. • Vertical ID.
Terminate VSI response	5GT-VS → Vertical	Response with the preliminary result of the operation.	• Result Code.

5.10.1.9 Modify VSI

This operation allows a vertical to modify an already instantiated VSI, providing a new VSD. The procedure to actually modify the VSI is handled by the 5GT-VS in an asynchronous manner and, when the procedure is completed, a notification about the modification result is sent to the vertical, if a suitable URL was provided by the vertical in the instantiation request. The corresponding workflow is described in Section 5.5.4.

The Modify VSI messages are specified in Table 37.

TABLE 37: MODIFY VSI MESSAGES

Message	Direction	Description	Parameters
Modify VSI request	Vertical → 5GT-VS	Request to modify a VSI based on a new VSD.	• VSI ID. • VSD ID (reference to the new VSD). • Vertical ID.
Modify VSI response	5GT-VS → Vertical	Response with the preliminary result of the operation.	• Result Code.

5.10.1.10 Notify VSI lifecycle event

This operation allows the 5GT-VS to inform the vertical about lifecycle events related to VSIs owned by the vertical itself. These notifications are asynchronous and can report the results of actions triggered by the vertical (e.g. the result of the instantiation, termination or modification procedures) or events triggered by an autonomous decision of the 5GT-VS (e.g. scaling in and out of the VSI as a consequence of Arbitrator's decision about a contention among vertical services with different priorities).

The Notify VSI lifecycle message is specified in Table 38.

TABLE 38: NOTIFY VSI LIFECYCLE EVENT MESSAGES

Message	Direction	Description	Parameters
Notify VSI lifecycle event	5GT-VS → Vertical	Notification of a lifecycle event.	• VSI ID. • LC notification type (e.g. instantiation result, termination result, modification result, scaling result). • Result Code. • VSD (only for notifications about scaling results, it indicates the new VSD currently associated to the VSI).

5.10.1.11 Query VSI monitoring parameter

This operation allows a vertical to retrieve monitoring parameters about an existing VSI previously requested by the vertical itself. The requested monitoring parameters should be included in the ones specified in the original VSD.

The Query VSI monitoring parameter messages are specified in Table 39.

TABLE 39: QUERY VSI MONITORING PARAMETER MESSAGES

Message	Direction	Description	Parameters
Query VSI monitoring parameter request	Vertical → 5GT-VS	Request to retrieve monitoring parameters about the VSI matching the given ID.	• VSI ID. • Vertical ID. • List<Monitoring Parameter>.
Query VSI monitoring parameter response	5GT-VS → Vertical	Response including the details of the requested VSI.	• Map<Monitoring Parameter; Value>. • Timestamp.

5.10.1.12 Subscription/Notification about VSI monitoring parameters

This operation allows a vertical to subscribe with the 5GT-VS to receive notifications about monitoring parameters, e.g. providing rules to describe thresholds for which notifications are generated when crossed or a time interval for periodical notifications.

The Subscription/Notification messages about VSI monitoring parameters are specified in Table 40.

TABLE 40: SUBSCRIBE/NOTIFY VSI MONITORING PARAMETERS MESSAGES

Message	Direction	Description	Parameters
Create VSI monitoring subscription request	Vertical → 5GT-VS	Request to create a new subscription for receiving notifications about monitoring parameters.	• VSI ID. • Vertical ID. • Map<Monitoring Parameter; Threshold>. • Time interval.

Create VSI monitoring subscription response	5GT-VS → Vertical	Response including the ID of the subscription.	• Result Code. • Subscription ID.
Delete VSI monitoring subscription request	Vertical → 5GT-VS	Request to delete a previous subscription.	• Subscription ID.
Delete VSI monitoring subscription response	5GT-VS → Vertical	Response with the result of the operation.	• Result Code.
Notify VSI monitoring parameter	5GT-VS → Vertical	Notification about a new monitoring parameter or threshold crossed event	• VSI ID. • Map<Monitoring Parameter; Value> • Timestamp.

5.10.1.13 Notify VSI failure event

This operation allows the 5GT-VS to inform the vertical about failures related to VSIs owned by the vertical itself. These notifications are generated asynchronously by the 5GT-VS and they are sent to the URL provided by the vertical in the instantiation request.

The Notify VSI failure event message is specified in Table 41.

TABLE 41: NOTIFY VSI FAILURE MESSAGES

Message	Direction	Description	Parameters
Notify VSI failure event	5GT-VS → Vertical	Notification of a failure event.	• VSI ID. • LC notification type (e.g. instantiation result, termination result, modification result, scaling result). • Result Code. • VSD (only for notifications about scaling results, it indicates the new VSD currently associated to the VSI).

5.10.2 Mgt-Vs reference point

5.10.2.1 Create tenant

This operation allows the system administrator to create a new tenant in the 5GT-VS repository. A tenant corresponds to a TSC (i.e. a vertical or an M(V)NO) who has established a business relationship with the TSP owning the 5GT-VS. Each tenant will be able to request VSIs in the limit of the SLAs signed between customer and provider; for this reason, the tenant information stored in the 5GT-VS repository will be used to validate and authorize the requests received on the Ve-Vs reference point for management of VSDs and VSIs. Tenant groups are used to define the sharing of services on a given slice, e.g. there could be a group of ‘car manufacturers’.

The Create tenant messages are specified in Table 42.

TABLE 42: CREATE TENANT MESSAGES

Message	Direction	Description	Parameters
Create tenant request	Mgt → 5GT-VS	Request to create a new tenant.	• Tenant ID. • Tenant name. • Tenant credential. • List<Group ID>.
Create tenant response	5GT-VS → Mgt	Response with the result of the operation.	• Result Code.

5.10.2.2 Query tenant

This operation allows the system administrator to retrieve all the information associated with an existing tenant, including the IDs of the VSDs created by and the VSIs instantiated for the tenant, the IDs of the active SLAs, and the total amount of resources currently allocated for the VSIs belonging to the tenant. Detailed information about each VSD, VSI, and SLA can then be obtained through the related query methods.

The Query tenant messages are specified in Table 43.

TABLE 43: QUERY TENANT MESSAGES

Message	Direction	Description	Parameters
Query tenant request	Mgt → 5GT-VS	Request to retrieve the details of a given tenant.	• Tenant ID.
Query tenant response	5GT-VS → Mgt	Response with the information associated to the given tenant.	• Tenant ID. • Tenant name. • Tenant credential. • List<Group ID>. • List<SLA ID>. • List<VSD ID>. • List<VSI ID>. • Total allocated resources.

5.10.2.3 Delete tenant

This operation allows the system administrator to delete an existing tenant. A tenant can be removed from the system only when all its VSIs have been terminated.

The Delete tenant messages are specified in Table 44.

TABLE 44: DELETE TENANT MESSAGES

Message	Direction	Description	Parameters
Delete tenant request	Mgt → 5GT-VS	Request to delete a tenant with the given ID.	• Tenant ID.
Delete tenant response	5GT-VS → Mgt	Response with the result of the operation.	• Result Code.

5.10.2.4 Create SLA

This operation allows the system administrator to create a new SLA for an existing tenant in the 5GT-VS repository. A non-exhaustive list of information that can be included in an SLA includes the maximum amount of resources (e.g. computing vCPUs or RAM or disk storage space, optionally distinguished between cloud and MEC resources) that can be allocated for a given tenant or the maximum number of VSIs that can be requested by the given tenant. The SLA information is used by the 5GT-VS to validate the requests from the tenants and, in particular, by the Arbitrator to identify contentions among vertical services belonging to the same customer.

The Create SLA messages are specified in Table 45.

TABLE 45: CREATE SLA MESSAGES

Message	Direction	Description	Parameters
Create SLA request	Mgt → 5GT-VS	Request to create a new SLA for the given tenant.	• Tenant ID. • List<SLA constraint>. • SLA Status (enabled, disabled).
Create SLA response	5GT-VS → Mgt	Response with the result of the operation and the SLA ID.	• Result Code. • SLA ID.

5.10.2.5 Query SLA

This operation allows the system administrator to retrieve the information about an SLA, given its ID.

The Query SLA messages are specified in Table 46.

TABLE 46: QUERY SLA MESSAGES

Message	Direction	Description	Parameters
Query SLA request	Mgt → 5GT-VS	Request to retrieve the details of a given SLA.	• SLA ID.
Query SLA response	5GT-VS → Mgt	Response with the information associated to the given SLA.	• SLA ID. • Tenant ID. • List<SLA constraint>. • SLA Status (enabled, disabled).

5.10.2.6 Delete SLA

This operation allows the system administrator to delete an existing SLA.

The Delete SLA messages are specified in Table 47.

TABLE 47: DELETE SLA MESSAGES

Message	Direction	Description	Parameters
Delete SLA request	Mgt → 5GT-VS	Request to delete an SLA with the given ID.	• SLA ID.
Delete SLA response	5GT-VS → Mgt	Response with the result of the operation.	• Result Code.

5.10.2.7 Modify SLA

This operation allows the system administrator to modify an existing SLA, e.g. to change its status enabling or disabling it or to update its content (amount of resources, etc.).

The Modify SLA messages are specified in Table 48.

TABLE 48: MODIFY SLA MESSAGES

Message	Direction	Description	Parameters
Modify SLA request	Mgt → 5GT-VS	Request to modify an SLA with the given ID.	• SLA ID. • List<SLA constraint>. • SLA Status (enabled, disabled).
Modify SLA response	5GT-VS → Mgt	Response with the result of the operation.	• Result Code.

5.10.2.8 Create policy

This operation, added in 5GT-VS R2, allows the system administrator to create a new policy for an existing tenant in the 5GT-VS repository. Policies are defined in an abstract format that can be specialized for each specific policy type. The policies are used by the 5GT-VS to identify restrictions or characteristics for VSIs requested by a given tenant, in order to specify the associated policies at the NFV-NSI level in the NFV-NSI instantiation requests to the 5GT-SO.

The Create policy messages are specified in Table 49.

TABLE 49: CREATE POLICY MESSAGES

Message	Direction	Description	Parameters
Create policy request	Mgt → 5GT-VS	Request to create a new policy for the given tenant.	• Tenant ID. • Policy Type. • List<Policy Key, Policy Value>. • Policy Status (enabled, disabled).
Create policy response	5GT-VS → Mgt	Response with the result of the operation and the policy ID.	• Result Code. • Policy ID.

5.10.2.9 Query policy

This operation, added in 5GT-VS R2, allows the system administrator to retrieve the information about a policy, given its ID.

The Query policy messages are specified in Table 50.

TABLE 50: QUERY POLICY MESSAGES

Message	Direction	Description	Parameters
Query policy request	Mgt → 5GT-VS	Request to retrieve the details of a given policy.	• Policy ID.

Query policy response	5GT-VS → Mgt	Response with the information associated to the given policy.	• Policy ID. • Tenant ID. • Policy Type. • List<Policy Key, Policy Value> • Policy Status (enabled, disabled).

5.10.2.10 Delete policy

This operation, added in 5GT-VS R2, allows the system administrator to delete an existing policy.

The Delete policy messages are specified in Table 51.

TABLE 51: DELETE POLICY MESSAGES

Message	Direction	Description	Parameters
Delete policy request	Mgt → 5GT-VS	Request to delete a policy with the given ID.	• Policy ID.
Delete policy response	5GT-VS → Mgt	Response with the result of the operation.	• Result Code.

5.10.2.11 Modify policy

This operation, added in 5GT-VS R2, allows the system administrator to modify an existing policy, e.g. to change its status enabling or disabling it or to update its content.

The Modify policy messages are specified in Table 52.

TABLE 52: MODIFY POLICY MESSAGES

Message	Direction	Description	Parameters
Modify policy request	Mgt → 5GT-VS	Request to modify a policy with the given ID.	• Policy ID. • List<Policy Key, Policy Value> • Policy Status (enabled, disabled).
Modify policy response	5GT-VS → Mgt	Response with the result of the operation.	• Result Code.

5.10.2.12 Create VS blueprint

This operation allows the system administrator to on-board a new VSB in the 5GT-VS 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) together with the NSDs and the VNF packages or MEC application descriptors (AppDs) that describe the NFV network services able to implement the vertical service defined in the blueprint itself. This design phase includes also the definition of suitable “translation rules” between VSDs that can be derived from the blueprint and the corresponding NSD deployment flavours, i.e. translation rules between QoS-like parameters and resource-like parameters. For this reason, a VSB is typically on-boarded together with the corresponding NSDs, VNF packages, AppDs, and translation rules.

The Create VS blueprint messages are specified in Table 53.

TABLE 53: CREATE VS BLUEPRINT MESSAGES

Message	Direction	Description	Parameters
Create VS blueprint request	Mgt → 5GT-VS	Request to create a new VSB.	• VSB (format specified in Section 5.4.1). • List<NSD> (format specified in Section 5.4.3). • List<VNF Package>. • List<AppD>. • List<Translation Rule>.
Create VS blueprint response	5GT-VS → Mgt	Response with the result of the operation and the VSB ID generated by the 5GT-VS.	• Result Code. • VSB ID.

5.10.2.13 Query VS blueprint

This operation allows the system administrator to retrieve the information about one or more VSBs, based on a filter.

The Query VS blueprint messages are specified in Table 54.

TABLE 54: QUERY VS BLUEPRINT MESSAGES

Message	Direction	Description	Parameters
Query VS blueprint request	Mgt → 5GT-VS	Request to retrieve one or more VSBs matching the given filter.	• Filter (e.g. VSB ID, …).
Query VS blueprint response	5GT-VS → Mgt	Response including the details of the requested VSBs.	• List<VS Blueprint> (format specified in Section 5.4.1).

5.10.2.14 Delete VS blueprint

This operation allows the system administrator to delete an existing VSB.

The Delete VS blueprint messages are specified in Table 55.

TABLE 55: DELETE VS BLUEPRINT MESSAGES

Message	Direction	Description	Parameters
Delete VS blueprint request	Mgt → 5GT-VS	Request to delete a VSB with the given ID.	• VSB ID.
Delete VS blueprint response	5GT-VS → Mgt	Response with the result of the operation.	• Result Code.

5.11 Annex III: Network service descriptors

In this annex we describe the most relevant fields of VNFD and NSD information elements, which have not been modified by 5G-T. For complete definitions, see IFA011 [29], and IFA014 [28], respectively.

5.11.1 Virtual Network Function Descriptor (VNFD)

This descriptor encapsulates all the information necessary to specify the characteristics of a VNF such as the software image that is going to use, the resources it will use, and other properties that help in the deployment of the VNF.

TABLE 56: VNFD INFORMATION ELEMENT

Field	Description
vnfId	Unique identifier for the VNFD.
swImageDesc	Software image that contain the VNF to be virtually deployed.
virtualCompDesc	CPU resources used by the VNF.
virtualStorageDesc	Disk resources for the VNF.
vnfExtCpd	Connection point descriptors used to connect with the VNF.
lifecycleManagementScript	Scripts used for the life cycle management of the VNF. This is a list of pairs <eventType, script> to specify how to deal with different life cycle events.
vnfIndicator	List of monitoring indicators used for this VNF, such as CPU usage, memory pressure, etc.

5.11.2 Physical Network Function Descriptor (PNFD)

An NFV-NS can make use of already existing NFs that are deployed as physical appliances. Information related to those NFs is not specified by the NSD, hence the PNFD must provide information on how that PNF can be connected in the VNFFG and what is its functionality.

TABLE 57: PNFD INFORMATION ELEMENT

Field	Description
pnfdid	Unique identifier for a PNFD descriptor.
pnfExtCp	List of connection points descriptors that can be used to attach to the PNF.
functionDescription	A string describing the PNF functionality.

5.11.3 Connection Point Descriptors (CPDs)

Every PNF and VNF have connection points that are interconnected with Virtual Links (VLs), these connection points are entities of their own that are part of the NSD.

TABLE 58: CPD INFORMATION ELEMENT

Field	Description
cpdId	Unique identifier for a CPD.
layerProtocol	The communication protocols used in the link: Ethernet, IPv4, IPv6, etc.
cpProtocol	Refers to protocol specific data, e.g. L2 or L3 address used to reach the CP.
description	A string describing the use of the CP.

5.11.4 Service Access Point Descriptor (SAPD)

It is a CPD used to provide access to an NFV-NS. It can be considered as an ingress/egress port for the NFV-NS.

TABLE 59: SAPD INFORMATION ELEMENT

Field	Description
cpdId	Unique identifier for a SAPD descriptor.
nsVirtualLinkDescId	Identifier of a VL used by the NS to connect with a VNF/PNF of the NS.
associatedCpdId	Identifier of a CPD attached to the VNF/PNF of the NS. Note that this CP is the one the VLD gives access to.
(Inherited_fields)	Fields inherited from the CPD entity.

5.11.5 Virtual Network Functions Forwarding Graph Descriptor (VNFFGD)

The NSD must define how the service NFs are interconnected (i.e. the NFs graph) and how the traffic flows in the service graph/sequence.

TABLE 60: VNFFGD INFORMATION ELEMENT

Field	Description
vnffgdId	Unique identifier for a VNFFGD descriptor.
vnfProfileId	List of VNFD_id present in the NFV-NS. These are nodes of the VNFFG.
appProfileId	List of profiles of APPs present in the NFV-NS. These are nodes of the VNFFG
pnfProfileId	List of profiles of PNFs present in the NFV-NS. These are nodes of the VNFFG.
virtualLinkProfileId	List of profiles of VLs present in the NFV-NS. These are edges of the VNFFG.
nestedNsProfileIdcpdPoolId	List of profiles of nested network services.References a pool of descriptors of connection points attached to the constituent

	VNFs and PNFs, SAPs or nested network services.
Nfpd	Network forwarding path associated with this forwarding graph.
appProfileId	List of profiles of APPs present in the NFV-NS. These are nodes of the VNFFG

5.11.6 Virtual Link Descriptor (VLD)

NFV-NSs use virtual links to connect the set of NFs present in the network service. Every VL should satisfy parameters to specify QoS, connectivity, security, and deployment flavours.

TABLE 61: VLD INFORMATION ELEMENT

Field	Description
virtualLinkDescId	Unique identifier for a VL descriptor.
connectivityType	Refers to type of link and protocol used.
virtualLinkDescFlavour	Latency, packet delay, packet loss ratio, traffic priority and availability of the VL.
description	A string specifying the purpose of the VL.

5.11.7 Monitoring Parameter

The monitoring parameters are a set of metrics to control the behaviour of resources used by the deployed NFV-NS.

TABLE 62: MONITORING PARAMETER INFORMATION ELEMENT

Field	Description
id	Unique identifier for a monitoring parameter.
name	Explanatory name of the monitoring parameter.
performanceMetric	Identify the virtualized resource performance metric.

5.11.8 Life Cycle Management

This information element specifies which script should be executed to deal with a specific event.

TABLE 63: LCMO INFORMATION ELEMENT

Field	Description
event	Type of event or external stimulus that triggers the LCM: e.g. start instantiation, start scaling, …
script	Program to be executed upon the occurance of the event, written in a domain-specific language.

5.11.9 Network Service Deployment Flavour (NSDF)

This information element details the specific flavour of the NFV-NS to be deployed and references the flavours of the components that are part of the NFV-NS.

TABLE 64: NSDF INFORMATION ELEMENT

Field	Description
nsdfId	Unique identifier for a NSDF entity.
vnfProfile	VNF profile to be used for the network service flavour.
pnfProfile	PNF profile to be used for the network service flavour.
virtualLinkProfile	VL profile to be used for the network service flavour.
NsProfile	Specifies a NS profile supported by this network service.
dependencies	List of IDs specifying the deployment order of the constituent VNFs and nested NSs.

5.11.10 Virtual Network Function Deployment Flavour (VNFDF)

Information element used to specify deployment details of a certain VNF.

TABLE 65: VNFDF INFORMATION ELEMENT

Field	Description
flavourId	Unique identifier for a VNFDF entity.
vduProfile	Describes additional instantiation data for the VDUs used in this flavour.
instantiationLevel	Levels (1…N) of resources that can be used to instantiate the VNF using this flavour.
localAffinityOrAntiAffinityRule	Set of affinity/antiaffinity rules related to the flavour. This is a list of restrictions such as location, resource-specific deployments, etc.
virtualLinkProfile	Defines the internal VLD along with additional data which is used in this DF.
monitoring_params	List or matrix of monitoring parameter IDs for every VNF instance to be deployed.

5.11.11 Virtual Link Deployment Flavour (VLDF)

This information element enhances the specification of behavioural requirements used by the VL to be deployed. It is necessary to meet the requirements that the link will have to hold within the NFV-NS that it belongs to.

TABLE 66: VLDF INFORMATION ELEMENT

Field	Description
flavourId	Unique identifier for a VLDF entity.
qos	Defines quality of service in terms of latency, delay variation, etc.
serviceAvailabilityLevel	Specifies one of three predefined availability levels

5.11.12 Quality of Service (QoS)

This information element specifies the quality of service attributes of a VL or an NFP. The most relevant attribute is the latency, which is used in placement decisions.

TABLE 67: QOS INFORMATION ELEMENT

Field	Description
latency	Specifies the maximum latency in ms.
packetDelayVariation	Specifies the maximum jitter in ms.
packetLossRatio	Specifies the maximum packet loss ratio.
priority	Specifies the priority level in case of congestion on the underlying physical links.

5.12 Annex IV: Vertical Service Examples

5.12.1 Real-time probe

This example of a vertical service is not intended as a vertical service for production use. But it can be used to test the actual deployment mechanisms and to determine the real-time capabilities of a virtualized infrastructure. The vertical service provides one server, performing the Linux utility cyclictest[33] repeatedly and providing the results as monitored Info.

5.12.1.1 Vertical Service Blueprint

No information is left open in the VSB. There is no virtual machine image to be provided by the vertical and the lifetime of vertical service instances are limited to one hour.

TABLE 68: VERTICAL SERVICE BLUEPRINT LATENCY PROBE

Field	Description
Name	Real-time Probe
Description	A Linux server performing the cyclictest utility, reporting the measured average and maximum latency.
Version	0.2
Identity	Xyz4712_bp

[33] http://manpages.ubuntu.com/manpages/cosmic/man8/cyclictest.8.html

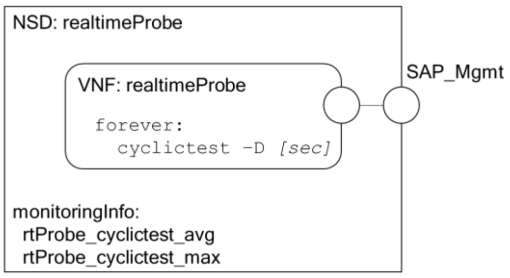
Parameters	none
Atomic functional components involved	stdLinux (see Table 74)
Service sequence	VNFFG in textual notation (ETSI NFV IFA 014) of  NSD: realtimeProbe VNF: realtimeProbe forever: cyclictest -D [sec] SAP_Mgmt monitoringInfo: rtProbe_cyclictest_avg rtProbe_cyclictest_max
Connectivity service	n/a
External interconnection	sapMgmt
Internal interconnection	n/a
SST	n/a (see the field SLA instead)
Service constraints	Geographical area: n/a Security: low Priority: medium Cost: n/a Synchronization: low Etc.
Management and control capabilities for the tenant	Provider managed
SLA	n/a
Monitoring	rtprobe_cyclictest_avg rtprobe_cyclictest_max
Lifetime	On-demand, 1h
Charging	n/a

TABLE 69: ATOMIC FUNCTIONAL COMPONENTS STDLINUX

Field	Description
Number of Application servers.	n/a (provided by Translator)
Images of virtual applications.	rtp: http://abc.com/stdLinux
Virtual application connection end points	cpIn, cpInRe, cpReIn
Lifecycle operations	To be defined
Scaling rules	n/a

5.12.1.2 Vertical Service Descriptor

No additional data is provided in the VSD, it is similar to the VSB.

TABLE 70: VERTICAL SERVICE DESCRIPTOR FOR LATENCY PROBE

Field	Description
Name	Latency 5Tonic
Description	Measure real-time capabilities in 5Tonic.
Version	2.0
Blueprint	Xyz4712_bp
Identity	Abc0816_vsd
SST	n/a
Service constraints	Geographical area: n/a Security: low Priority: medium Cost: n/a Synchronization: low Etc.
Management and control capabilities for the tenant	Provider managed
SLA	n/a
Monitoring	rtprobe_cyclictest_avg rtprobe_cyclictest_max
Lifetime	On-demand, 1h
Charging	To be defined

5.13 Annex V: Blueprints and descriptors

This annex reports the json format for the blueprints of vertical services, defined in WP3 for the different use cases. These blueprints are used during the integration and validation activities in WP5.

5.13.1 Automotive service

The automotive use case includes two services, the Extended Virtual Sensing and the Video Streaming services. In the following we report the VSB of these services, while VSDs, NSDs and VNFDs are available on Github at the following link:

https://github.com/5g-transformer/5gt-vs/tree/master/blueprints/automotive

```
{
    "version": "0.1",
    "name": "EVS",
    "description": "Automotive Extended Virtual Sensing Service",
    "parameters": [{
      "parameterId": "MaximumNumberVehicles",
      "parameterName": "MaximumNumberVehicles",
      "parameterType": "number",
      "parameterDescription": "Maximum number of vehicles that should be supported by
the service",
      "applicabilityField": "automotive"
    }],
    "atomicComponents": [{
        "componentId": "CIM",
        "serversNumber": 1,
        "endPointsIds": ["vCIM_data_ext"]
    }, {
        "componentId": "EVS_algorithm",
        "serversNumber": 2,
        "endPointsIds": ["vEVS_data_ext"]
    }, {
        "componentId": "DENM_transmitter",
        "serversNumber": 1,
        "endPointsIds": ["vDENMgenerator_data_ext"]
    }, {
        "componentId": "EPC",
        "serversNumber": 1,
        "endPointsIds": ["pEPC_data"]
    }],
    "endPoints": [{
        "endPointId": "evs_sap_data",
        "external": true,
        "management": true,
        "ranConnection": true
    }, {
        "endPointId": "vCIM_data_ext",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "vEVS_data_ext",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "vDENMgenerator_data_ext",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "pEPC_data",
        "external": true,
        "management": true,
        "ranConnection": false
    }],
    "connectivityServices": [{
        "endPointIds": ["sap_data","vCIM_data_ext","vEVS_data_ext","DENM_transmitter"],
        "external": true
    }],
    "configurableParameters": ["MinTargetTimeToCollision","MaxTargetTimeToCollision"]
}
```

FIGURE 48: EXTENDED VIRTUAL SENSING VSB

```
{
    "version": "0.1",
    "name": "VideoStreaming",
    "description": "Automotive Video Streaming Service",
    "parameters": [{
      "parameterId": "MaximumNumberUsers",
      "parameterName": "MaximumNumberUsers",
      "parameterType": "number",
      "parameterDescription": "Maximum number of users that should be supported by the
service",
      "applicabilityField": "automotive"
    }],
    "atomicComponents": [{
        "componentId": "Video_Streamer",
        "serversNumber": 1,
        "endPointsIds": ["vVS_data_ext"]
    }, {
        "componentId": "Video_Streamer_Algorithm",
        "serversNumber": 1,
        "endPointsIds": ["vVS_algo_data_ext"]
    }, {
        "componentId": "EPC",
        "serversNumber": 1,
        "endPointsIds": ["pEPC_data"]
    }],
    "endPoints": [{
        "endPointId": "vs_sap_data",
        "external": true,
        "management": true,
        "ranConnection": true
    }, {
        "endPointId": "vVS_data_ext",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "vVS_algo_data_ext",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "pEPC_data",
        "external": true,
        "management": true,
        "ranConnection": false
    }],
    "connectivityServices": [{
        "endPointIds": ["vs_sap_data","vVS_data_ext","vVS_algo_data_ext","pEPC_data"],
        "external": true
    }]
}
```

FIGURE 49: VIDEO STREAMING VSB (AUTOMOTIVE UC)

5.13.2 Entertainment services

The entertainment use case includes a single service, the virtual Content Delivery Network service. In the following we report the VSB of this service, while VSD, NSD and VNFDs are available on Github at the following link:

`https://github.com/5g-transformer/5gt-vs/tree/master/blueprints/entertainment/`

```
{
   "version": "0.1",
    "name": "CDN",
    "description": "Content Delivery Network Service",
    "parameters": [{
      "parameterId": "MaximumNumberUsers",
      "parameterName": "MaximumNumberUsers",
      "parameterType": "number",
      "parameterDescription": "Maximum number of users that should be supported by the
service",
      "applicabilityField": "entertainment"
    }],
    "atomicComponents": [{
        "componentId": "OriginServer",
        "serversNumber": 1,
        "endPointsIds": ["spr1MgtExt", "spr1DataExt"]
    }, {
        "componentId": "VideoCache",
        "serversNumber": 2,
        "endPointsIds": ["spr2DistExt", "spr2DataExt"]
    }, {
        "componentId": "WebServer",
        "serversNumber": 1,
        "endPointsIds": ["webDistExt"]
    }],
    "endPoints": [{
        "endPointId": "mgtSap",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "videoSap",
        "external": true,
        "management": true,
        "ranConnection": true
    }, {
        "endPointId": "spr1MgtExt",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "spr1DataExt",
        "external": false,
        "management": false,
        "ranConnection": false
    }, {
        "endPointId": "spr2DistExt",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "spr2DataExt",
        "external": false,
        "management": false,
        "ranConnection": false
    }, {
        "endPointId": "webDistExt",
        "external": true,
        "management": true,
        "ranConnection": false
    }],
    "connectivityServices": [{
        "endPointIds": ["videoSap","spr2DistExt","webDistExt"],
        "external": true
    }, {
        "endPointIds": ["mgtSap","spr1MgtExt"],
        "external": true
    }, {
        "endPointIds": ["spr1DataExt","spr2DataExt"],
        "external": false
    }]
}
```

FIGURE 50: VIRTUAL CONTENT DELIVERY NETWORK VSB

5.13.3 e-Health services

The eHealth use case includes two services, the eHealth Monitoring and the eHealth Emergency Management services. In the following we report the VSB of these services, while VSDs, NSDs and VNFDs are available on Github at the following link:

`https://github.com/5g-transformer/5gt-vs/tree/master/blueprints/eHealth/`

```
{
    "version": "0.1",
    "name": "eHealthMonitoring",
    "description": "Monitoring Service for eHealth use case",
    "parameters": [{
      "parameterId": "MaximumNumberUsers",
      "parameterName": "MaximumNumberUsers",
      "parameterType": "number",
      "parameterDescription": "Maximum number of users that should be supported by the
service",
      "applicabilityField": "eHealth"
    }],
    "atomicComponents": [{
        "componentId": "eHealth_monitoring_backend",
        "serversNumber": 1,
        "endPointsIds": ["data_ehealth_mon_be_sap", "mgt_ehealth_mon_be_sap"]
    }, {
        "componentId": "vEPC",
        "serversNumber": 1,
        "endPointsIds": ["s1c_s1u_vepc_sap", "sgi_vepc_sap", "s5_vepc_sap",
"mgt_vepc_sap"]
    }],
    "endPoints": [{
        "endPointId": "s1c_s1u_ehealth_mon_sap",
        "external": true,
        "management": false,
        "ranConnection": true
    }, {
        "endPointId": "mgt_eHealth_mon_sap",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "sgi_ehealth_mon_sap",
        "external": true,
        "management": false,
        "ranConnection": false
    }, {
        "endPointId": "s5_ehealth_mon_sap",
        "external": true,
        "management": false,
        "ranConnection": false
    }, {
        "endPointId": "data_ehealth_mon_be_sap",
        "external": false,
        "management": false,
        "ranConnection": true
    }, {
        "endPointId": "mgt_ehealth_mon_be_sap",
        "external": false,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "s1c_s1u_vepc_sap",
        "external": false,
        "management": false,
        "ranConnection": true
    }, {
        "endPointId": "sgi_vepc_sap",
        "external": false,
        "management": false,
        "ranConnection": false
    }, {
        "endPointId": "s5_vepc_sap",
        "external": false,
        "management": false,
        "ranConnection": false
    }, {
        "endPointId": "mgt_vepc_sap",
        "external": false,
        "management": true,
        "ranConnection": false
    }],
    "connectivityServices": [{
        "endPointIds": ["mgt_vepc_sap","mgt_ehealth_mon_be_sap","mgt_eHealth_mon_sap"],
        "external": true
    }, {
        "endPointIds": ["s1c_s1u_ehealth_mon_sap","s1c_s1u_vepc_sap"],
        "external": true
    }, {
        "endPointIds": ["s5_ehealth_mon_sap","s5_vepc_sap"],
        "external": true
    }, {
        "endPointIds": ["sgi_ehealth_mon_sap","sgi_vepc_sap","data_ehealth_mon_be_sap"],
        "external": true
    }]
}
```

FIGURE 51: EHEALTH MONITORING SERVICE VSB

```
{
    "version": "0.1",
    "name": "eHealthEmergency",
    "description": "Emergency Service for eHealth use case",
    "parameters": [{
      "parameterId": "MaximumNumberAmbulances",
      "parameterName": "MaximumNumberAmbulances",
      "parameterType": "number",
      "parameterDescription": "Maximum number of ambulances that should be supported by
the service",
      "applicabilityField": "eHealth"
    }],
    "atomicComponents": [{
        "componentId": "eHealth_monitoring_service",
        "serversNumber": 1,
        "endPointsIds":           ["sgi_ehealth_mon_sap",           "s5_ehealth_mon_sap",
"s1c_s1u_ehealth_mon_sap", "mgt_ehealth_mon_sap"]
    }, {
        "componentId": "eHealth_edge",
        "serversNumber": 1,
        "endPointsIds": ["sgi_ehealth_emergency_edge_sap",
"s5_ehealth_emergency_edge_sap", "mgt_ehealth_emergency_edge_sap"]
    }],
    "endPoints": [{
        "endPointId": "s1c_s1u_ehealth_emergency_sap",
        "external": true,
        "management": false,
        "ranConnection": true
    }, {
        "endPointId": "mgt_ehealth_emergency_sap",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "sgi_ehealth_mon_sap",
        "external": false,
        "management": false,
        "ranConnection": true
    }, {
        "endPointId": "s5_ehealth_mon_sap",
        "external": false,
        "management": false,
        "ranConnection": false
    }, {
        "endPointId": "s1c_s1u_ehealth_mon_sap",
        "external": false,
        "management": false,
        "ranConnection": true
    }, {
        "endPointId": "mgt_ehealth_mon_sap",
        "external": false,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "sgi_ehealth_emergency_edge_sap",
        "external": false,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "s5_ehealth_emergency_edge_sap",
        "external": false,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "mgt_ehealth_emergency_edge_sap",
        "external": false,
        "management": true,
        "ranConnection": false
    }],
    "connectivityServices": [{
        "endPointIds":
["mgt_ehealth_emergency_edge_sap","mgt_ehealth_mon_sap","mgt_ehealth_emergency_sap"],
        "external": true
    }, {
        "endPointIds": ["s1c_s1u_ehealth_mon_sap","s1c_s1u_ehealth_emergency_sap"],
        "external": true
    }, {
        "endPointIds": ["s5_ehealth_emergency_edge_sap","s5_ehealth_mon_sap"],
        "external": false
    }, {
        "endPointIds": ["sgi_ehealth_emergency_edge_sap","sgi_ehealth_mon_sap"],
        "external": false
    }]
}
```

FIGURE 52: EHEALTH EMERGENCY MANAGEMENT SERVICE VSB

5.13.4 e-Industry services

The e-Industry use case includes a single service, the Factory Control service. In the following we report the VSB of this service, while VSD, NSD and VNFDs are available on Github at the following link:

`https://github.com/5g-transformer/5gt-vs/tree/master/blueprints/eIndustry/`

```
{
    "version": "0.1",
    "name": "eIndustry",
    "description": "Factory Control Service",
    "parameters": [{
      "parameterId": "MaximumNumberEndDevices",
      "parameterName": "MaximumNumberEndDevices",
      "parameterType": "number",
      "parameterDescription": "Maximum number of end devices that should be supported by
the service",
      "applicabilityField": "eIndustry"
    }, {
      "parameterId": "EndDevicesDensity",
      "parameterName": "EndDevicesDensity",
      "parameterType": "number",
      "parameterDescription": "Maximum density of end devices that should be supported by
the service",
      "applicabilityField": "eIndustry"
    }],
    "atomicComponents": [{
        "componentId": "FactoryControlServer",
        "serversNumber": 1,
        "endPointsIds": ["fcs_data_ext"]
    }, {
        "componentId": "EPC",
        "serversNumber": 1,
        "endPointsIds": ["pEPC_data"]
    }],
    "endPoints": [{
        "endPointId": "fc_sap_data",
        "external": true,
        "management": true,
        "ranConnection": true
    }, {
        "endPointId": "fcs_data_ext",
        "external": true,
        "management": true,
        "ranConnection": false
    }, {
        "endPointId": "pEPC_data_ext",
        "external": true,
        "management": true,
        "ranConnection": false
    }],
    "connectivityServices": [{
        "endPointIds": ["fc_sap_data","fcs_data_ext","pEPC_data"],
        "external": true
    }]
}
```

FIGURE 53: FACTORY CONTROL SERVICE VSB

5.13.5 MNO/MVNO services

The MVNO use case includes a single service, the Wireless Edge Core Network service. In the following we report the VSB of this service, while NSD and VNFDs are available on Github at the following link:

https://github.com/5g-transformer/5gt-vs/tree/master/blueprints/mvno/

```
{
   "version": "0.1",
       "name": "Wireless_Edge_Core_Network_Service",
       "description": "This service activates a backbone and connects it to the tenant's
access network to allow its end-users connecting to the data network.",
       "parameters": [{
              "parameterId": "users",
              "parameterName": "users",
              "parameterType": "number",
              "parameterDescription": "number of users"
       },
       {
              "parameterId": "active_sessions",
              "parameterName": "active_sessions",
              "parameterType": "number",
              "parameterDescription": "Number of sessions being under processing"
       },
       {
              "parameterId": "session_rate",
              "parameterName": "session_rate",
              "parameterType": "number",
              "parameterDescription": "Number of sessions setup per second"
       }]
}
```

FIGURE 54: MVNO SERVICE VSB

5.13.6 Additional services for performance monitoring

The VSB of the real-time network probe vertical service is shown in Figure 55.

```
{
       "version": "0.2",
       "name": "realtimeProbe_locationAny",
       "description": "One server, performing cyclictest.",
	"atomicComponents": [{
		"componentId": "networkProbe",
		"serversNumber": 1,
		"endPointsIds": ["CP_Mgmt"]
	}],
	"endPoints": [{
		"endPointId": "CP_Mgmt",
		"external": false,
		"management": true,
		"ranConnection": false
	}, {
		"endPointId": "SAP_mgmt",
		"external": true,
		"management": true,
		"ranConnection": false
	}],
	"connectivityServices": [{
		"endPointIds": ["CP_Mgmt","SAP_mgmt"],
		"external": true
	}]
}
```

FIGURE 55: NETWORK PROBE SERVICE VSB