H2020 5Growth Project

Grant No. 856709

D3.3: First version of software implementation for the platform

Abstract

The goal of this deliverable is to contain the documentation of the first release of software developed within the 5Growth project for the integration of 5Growth platform with ICT-17 platforms and with the infrastructure deployed for the project pilots (available in the project GIT repository). This code has been developed based on the initial analysis and design reported in D3.2. Further design decisions not included in D3.2 are summarized in this document as well.

Document properties

Document number	D3.3
Document title	First version of software implementation
Document responsible	Diego San Cristobal (ERC)
Document editor	Diego San Cristobal (ERC)
Editorial team	Diego San Cristobal (ERC), Fabio Ubaldi (TEI), Giada Landi (NXW), Juan Brenes (NXW), Aitor Zabala (TELCA), José Bonnet (ALB), Carlos Marques (ALB), Miguel Mesquita (ALB), Carlos Guimaraes (UC3M) Jorge Baranda (CTTC), Josep Mangues (CTTC), Engin Zeydan (CTTC), Paolo Dini (CTTC), Ricardo Martínez (CTTC), Luca Vettori (CTTC), Carlos Guimaraes (UC3M), Carlos J. Bernardos (UC3M), Luigi Girletti (UC3M).
Target dissemination level	PU
Status of the document	Final
Version	1.0
Delivery date	November 30, 2020
Actual delivery date	November 30, 2020

Production properties

Reviewers	Carlos Guimaraes (UC3M), Aitor Zabala (TELCA), Andres Garcia Saavedra (NEC)

Disclaimer

Contents

List of Figures

List of Tables

List of Acronyms

5Gr-RL – 5Growth Resource Layer

5Gr-SO – 5Growth Service Orchestrator

5Gr-VS – 5Growth Vertical Slicer

BBU – Baseband Unit

CSMF – Communication Service Management Function

ExpB – Experiment Blueprints

ExpD – Experiment Descriptor

HSS – Home Subscriber Server

IWL – Interworking Layer

MSO-LO – Multi-Site NSO to Local Orchestrator

NBI – North Bound Interface

NFs – Network Functions

NFVO – Network Functions Virtualization Orchestrator

NSDs – Network Service Descriptors

NST – Network Slice Template

RBAC – Role-Based Access Control

RU – Remote Unit

SBI – South Bound Interface

vEPC – virtual Evolved Packet Core

VS LCM – Vertical Service Lifecycle Manager

VSB – Vertical Service Blueprints

VSD – Vertical Service Descriptor

VSMF – Vertical Service Management Function

VNFDs – VNF Descriptors

Executive Summary

5Growth project targets the validation of vertical use cases based on innovative technologies such as 5G infrastructure, virtualization and dynamic service orchestration, organized around INNOVALIA, COMAU, EFACEC_S and EFACEC_E pilots. The 5Growth software stack contributes to that purpose providing functionalities to orchestrate vertical services, among others. However, an experiment validation framework is required as well, which is a feature provided by ICT-17 platforms. Therefore, 5Growth stack must work jointly with ICT-17 platforms to benefit from their experiment validation facilities.

In the integration of 5Growth with 5G-EVE, needed for INNOVALIA pilot, the 5Gr-VS interacts with 5G-EVE Portal, requesting the deployment and instantiation of the whole vertical service. Then, it is the 5G-EVE IWL which instructs where the network services are deployed, either under the 5G-EVE domain or under the 5Growth domain (by interacting with the 5Gr-SO). Conversely, in the integration of 5Growh with 5G-VINNI, needed for EFACEC_S and EFACEC_E pilots, the 5Gr-VS interacts with the NSMF at 5G-VINNI. In this case, also the full service is requested by means of 5Gr-VS. The network services deployment is done by the SONATA platform available at 5G-VINNI site, which also requires an adaptor to translate the request received.

Furthermore, in the context of the pilots, adaptors may be needed to translate the standards used by 5Growth to the ones used by the infrastructure deployed. For COMAU pilot, a translation is needed to adapt the 5Growth stack network slicing functionalities to the standards used by the 5G infrastructure.

The required software adaptors for integrating 5Growth with ICT-17 platforms and the project pilots are listed in the introductory part of this deliverable. Later in the document, the first software release of each adaptor is fully documented, covering the general description of the software architecture, detailing the services offered, illustrating the workflow diagrams and explaining the implementation in terms of methods, data models and other resources.

This first software release covers in general the most common lifecycle management operations. Towards the final software release, further testing and debugging will be done in a fully integrated environment and new functionalities will be developed in the areas of: *(i)* integration within the pilots; *(ii)* support of required 5Growth WP2 innovations; and *(iii)* integrated monitoring support.

1. Introduction

This deliverable D3.3 *First version of software implementation* contains the documentation of the first release of integration of 5Growth with ICT-17 platforms software components. It has been developed under the scope of tasks T3.2 to T3.6. The deliverable includes a low-level description of the software components and references to the repositories for the software components.

1.1. Initial context

This software is based on the analysis, architecture, and interface definitions exposed in D3.2 [1]. In that previous deliverable, some different alternatives were being considered. In the following subsections, a summary of the final design decisions is described.

1.1.1. 5Growth integration with 5G-EVE

For the integration of 5Growth with 5G-EVE, option 2 described in D3.2 Section 3.3 [1] has been selected. The choice is motivated by the ability to fully exploit the inter-site capabilities offered by the 5G-EVE platform for provisioning and monitoring services deployed across multiple sites, relying on the default model defined in the 5G-EVE framework to interact with external platforms from ICT-19 projects. It can be summarized as follows. The Communication Service Management Function (CSMF) at 5Growth Vertical Slicer (5Gr-VS) interacts with the 5G-EVE Portal through a programmable REST API to request the deployment and instantiation of the whole vertical service by the 5G-EVE platform (i.e., vertical service request). When this request is received, the 5G-EVE platform, more specifically, the 5G-EVE Interworking Layer (IWL), instructs what network services are deployed and instantiated in the vertical premises' resources, by interacting with the 5Growth Service Orchestrator (5Gr-SO) in the vertical premises. This interaction is visually depicted in Figure 1.

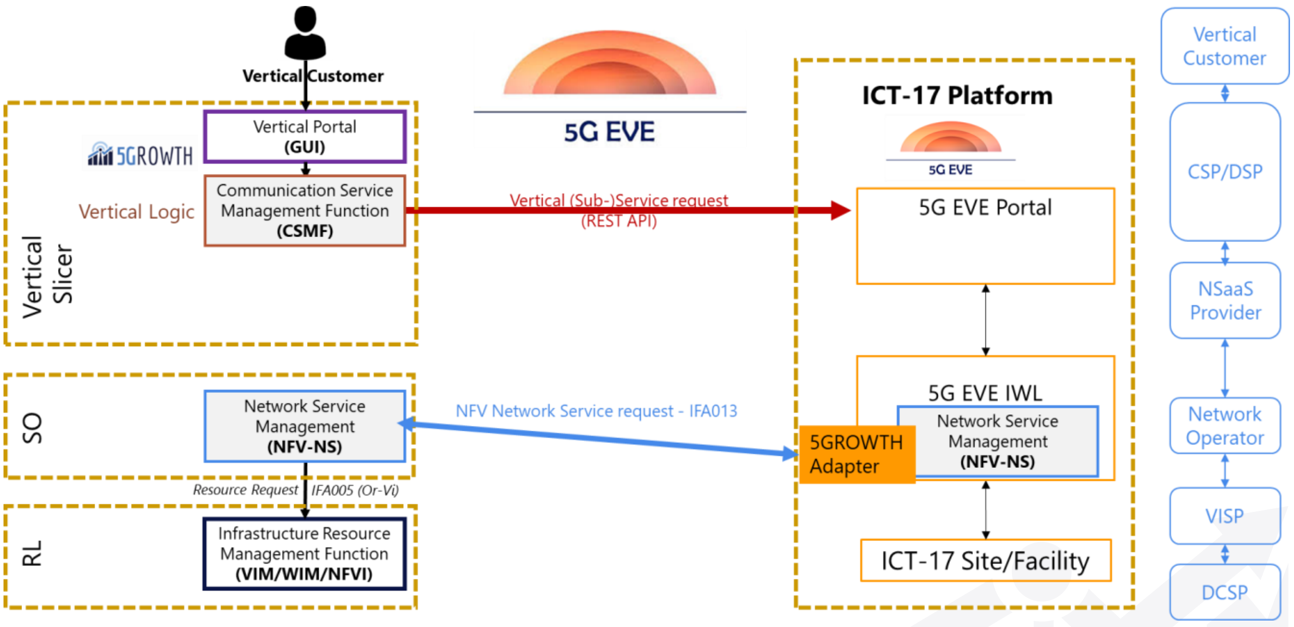

FIGURE 1: 5GROWTH INTEGRATION WITH 5G-EVE

For further details, please refer to D3.2 Section 3.3 [1].

The required software components are the following:

- 5Gr-VS driver towards 5G-EVE, in 5Growth platform, see Section 2.1;
- 5G-EVE IWL catalogue driver, in 5G-EVE platform, see Section 2.2.1;
- 5G-EVE IWL Lifecycle Manager driver, in 5G-EVE platform, see Section 2.2.2.

1.1.2. 5Growth integration with 5G-VINNI

For the integration of 5Growth with 5G-VINNI, the CSMF at 5Gr-VS interacts with the Network Slice Management Function (NSMF) at 5G-VINNI. In particular, the integration between 5Growth and 5G-VINNI is done with the SONATA platform, a result of the 5GTANGO project, which is available at the 5G-VINNI Aveiro site. This integration comprises the adaptation of the 5Gr-VS requests to the SONATA Slice Manager, which has been designed and implemented prior to the 3GPP technical specification 28.531 [2].

The flow between 5Gr-VS and 5G-VINNI's Network Function Virtualization Orchestrator (NFVO) follows the lines shown in Figure 2.

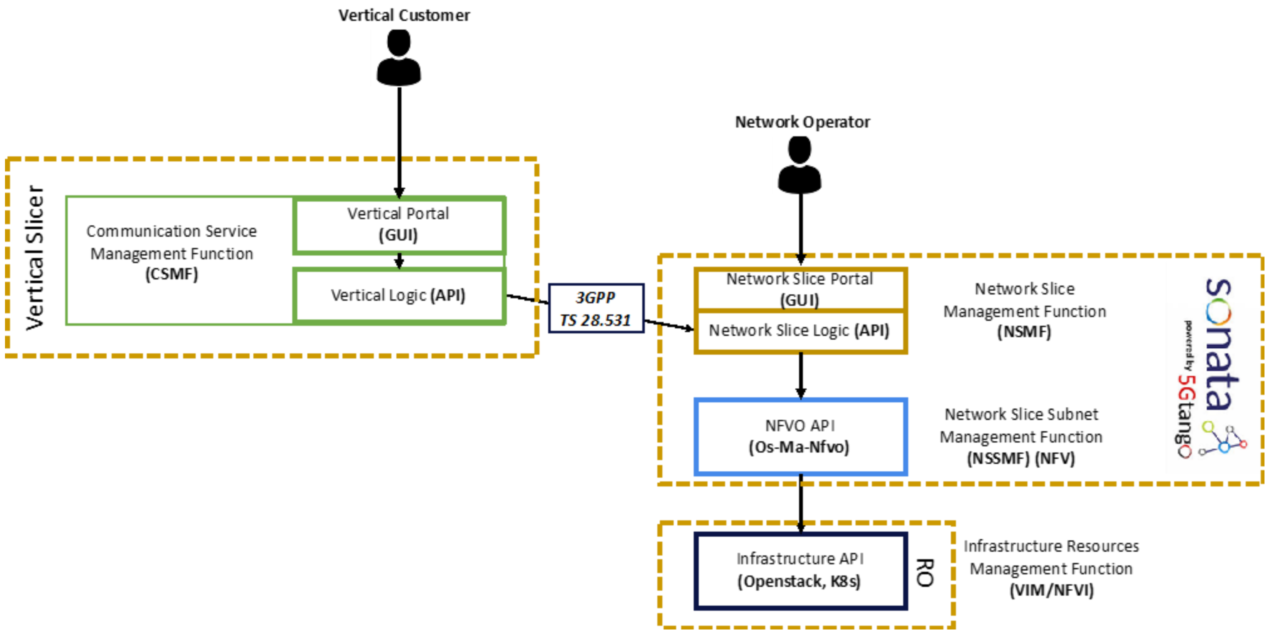
Vertical Customer
Vertical Slicer
Communication Service Management Function (CSMF)
Vertical Portal (GUI)
Vertical Logic (API)
3GPP TS 28.531
Network Operator
Network Slice Portal (GUI)
Network Slice Logic (API)
Network Slice Management Function (NSMF)
NFVO API (Os-Ma-Nfvo)
Network Slice Subnet Management Function (NSSMF) (NFV)
sonata powered by 5Gtango
Infrastructure API (Openstack, K8s)
RO
Infrastructure Resources Management Function (VIM/NFVI)

FIGURE 2: 5GROWTH INTEGRATION WITH 5G-VINNI

For further details, please refer to D3.2 Section 3.4 [1].

The required software components are the following:

- 5Gr-VS driver towards 5G-VINNI, in 5Growth platform, see Section 2.3;
- SONATA adaptor, in 5G-VINNI platform, see Section 2.4.

1.1.3. COMAU pilot

COMAU pilot is using the network slicing feature offered by 5Growth platform. For that reason, specific code was developed for this pilot in particular to adapt the format handled by 5Growth to the format handled by the Radio Access Network (RAN) controller equipment used in the pilot. This is further detailed in Section 2.5.

1.2. Structure of the document

Section 2 contains the documentation of the software artifacts. For each of the sections, there are three subsections:

- **general description**: includes the overview and the architecture of the software;
- **services and workflows**: describes the services or operations and depicts the workflow diagrams;
- **implementation**: lists the resources, methods and/or data models.

Section 3 covers a high-level summary of the improvements and functionalities envisioned for the second release of software.

2. Software artifacts

This section contains the description of the software contained in this delivery. It includes the code developed to integrate 5Growth with ICT-17 platforms and the code required by COMAU pilot to use 5Growth Resource Layer (5Gr-RL) functionalities. The list of components is collected in Table 1:

TABLE 1: LIST OF SOFTWARE COMPONENTS

Name	Section	Repository
5Gr-VS driver towards 5G-EVE	2.1	https://github.com/5growth/5gr-vs
5G-EVE IWL catalogue driver	2.2.1	https://github.com/nextworks-it/5g-catalogue Note: in "5Growth" branch
5G-EVE IWL Lifecycle Manager driver	2.2.2	https://github.com/5growth/mso-lo
5Gr-VS driver towards 5G-VINNI	2.3	https://5growth.eu/redmine/projects/5growth/repository/5gr-vs
SONATA adaptor	2.4	https://github.com/5growth/sonata-drivers
Code developed for COMAU integration	2.5	https://5growth.eu/redmine/projects/5growth/repository/pilots/COMAU

2.1. 5Gr-VS driver towards 5G-EVE

2.1.1. General description

The interaction between the 5Gr-VS and the 5G-EVE platform exploits the 5Gr-VS multi-domain functionalities at the CSMF level, where a vertical service can be decomposed into multiple "sub-services" each of them deployed in a specific domain. Following this approach, the 5Gr-VS can delegate the provisioning and management of the requested vertical services (or part of them) to external domains. In the particular case of the interaction with 5G-EVE, the vertical service requested at the North Bound Interface (NBI) of the 5Gr-VS is mapped into an equivalent concept at the 5G-EVE level (i.e. a 5G-EVE experiment), which is then managed through the 5G-EVE portal.

As already discussed in Deliverable D3.2 [1], the automated interaction between the 5Gr-VS and the 5G-EVE portal covers the phases of the vertical service customization, provisioning, and runtime, up to the termination. At the design phase, i.e., during the definition of the blueprints, the service designer needs to define the service components that must be deployed in each domain and build the Vertical Service Blueprints (VSB) and Experiment Blueprints (ExpB) accordingly. These blueprints must be on-boarded in the 5G-EVE portal and in the 5Gr-VS catalogue, so that they will be available as models to request a customized instantiation of a particular vertical service spanning across 5Growth and 5G-EVE domains.

An example of VSB for the 5Gr-VS catalogue is reported in Figure 3.

```
{
   "vsBlueprint": {
       "interSite": true,
       "version": "1.0",
       "name": "vs service name",
       "description": "end_to_end service",
       "parameters": [
           {
               "parameterId": "5GEVE_VSS.<experiment_param_id>",
               "parameterName": "experiment parameter name",
               "parameterType": "number",
               "parameterDescription": "description",
               "applicabilityField": "applicabilityField"
           }
...
       ],
       "configurableParameters": [
           "5GEVE_VSS.expd.kpi.<expb_kpiId>.lowerbound",
           "5GEVE_VSS.expd.kpi.<expb_kpiId>.upperbound",
           "5GEVE_VSS.exp.targetSites",
           "5GEVE_VSS.exp.experimentStopTime",
           "5GEVE_VSS.exp.useCase",
           "5GEVE_VSS.tc.tcBlueprintIds",
           "5GEVE_VSS.tc.<tcBlueprintId>.params.<param_id>"

       ],
       "atomicComponents": [
           {
               "type": "SERVICE",
               "componentId": "5GEVE_VSS",
               "compatibleSite": "5GEVE",
               "associatedVsbId": "<Experiment blueprint id>"
           }
       ]
   }
}
```

FIGURE 3: EXAMPLE OF VSB FOR 5GR-VS CATALOGUE

The vertical service is declared as an inter-site one and the list of its atomic components includes at least one subservice (i.e. with type declared as "SERVICE"). In our case, the site where this subservice needs to be deployed is "5GEVE" and the VSB associated with the subservice is defined in the "associatedVsbId" field. It should be noted that, in the case of composite vertical services where all the subservices are managed directly in the local domain controlled by the 5Gr-VS, this ID will correspond to the ID of a VSB available in the 5Gr-VS catalogue. In our case, where the subservice is controlled by the 5G-EVE platform, the ID corresponds to a blueprint available in the 5G-EVE Portal catalogue, and in particular to an Experiment Blueprint (ExpB).

In terms of QoS parameters, defined in the "parameters" field and driving the translation between vertical services and network slices/services of specific size, it should be noted that the parameters associated with the composite service should map the ones of the subservices. This approach allows determining the dimension of each subservice based on the custom parameters defined by the customer when defining the Vertical Service Descriptor (VSD) of the composite service from the 5Gr-VS GUI. In our case, we have a 1:1 mapping between the parameters of the composite VSB and the ones of the 5G-EVE ExpB associated with the subservice ("experiment_param_id"). More complex mappings may be investigated in the future.

Finally, in order to enable the full usage of all the 5G-EVE functionalities from the 5Growth platform, we exploit the concept of the "configurableParameters" that in the 5Gr-VS are adopted to allow the user to request a particular configuration of the vertical service. Declaring specific configurable parameters in the VSB, the user can request the execution of particular test cases in the 5G-EVE platform, configure each of them, set target thresholds for application and infrastructure KPIs, etc. going beyond the original lifecycle management functionalities offered by the 5Growth platform alone.

2.1.2. Services and workflows

The following figures provide a detailed view of the workflows between 5Gr-VS and 5G-EVE Portal during the different phases of a vertical service lifecycle.

Figure 4 shows the design of the vertical service. Here the vertical interacts initially with the 5G-EVE Portal GUI to onboard the blueprints of vertical service (VSB-A in this example), test cases (optional) and experiment blueprint associated with the part of the service that needs to run in the 5G-EVE platform. In a second step, the vertical onboards the blueprint of the composite service (VSB in this example) into the 5Gr-VS catalogue, where it is stored. The composite VSB information model must follow the format previously discussed (Figure 3).

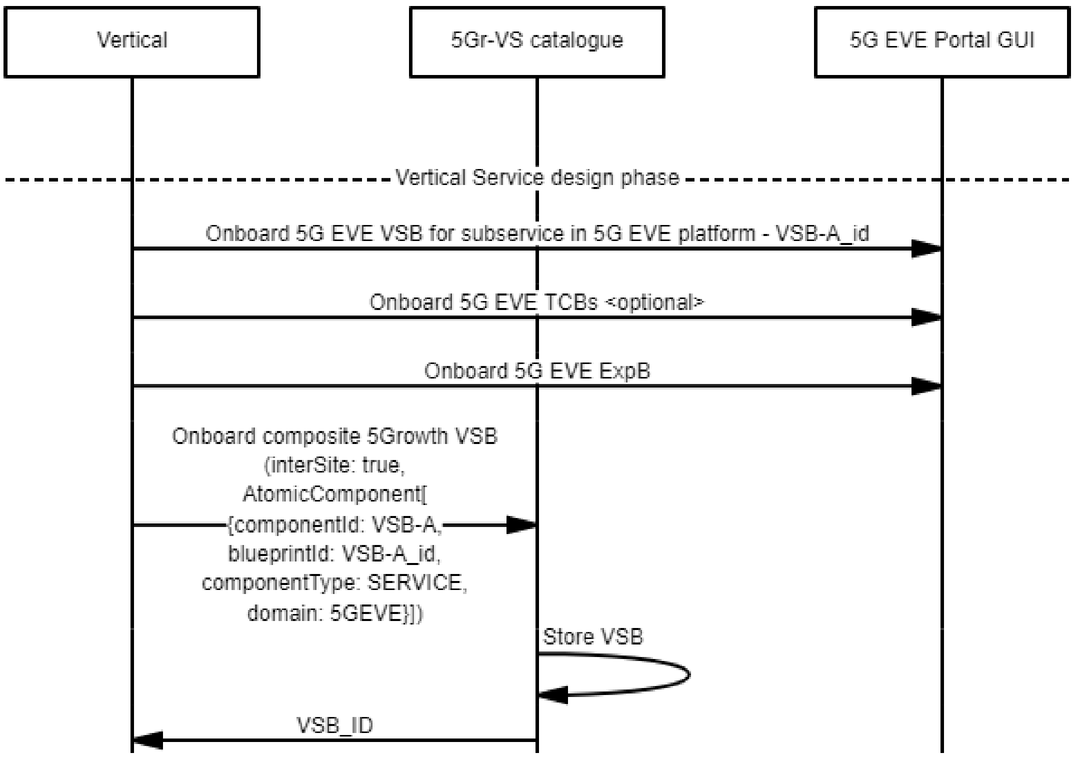
Vertical
5Gr-VS catalogue
5G EVE Portal GUI
Vertical Service design phase
Onboard 5G EVE VSB for subservice in 5G EVE platform - VSB-A_id
Onboard 5G EVE TCBs <optional>
Onboard 5G EVE ExpB
Onboard composite 5Growth VSB (interSite: true, AtomicComponent[{componentId: VSB-A, blueprintId: VSB-A_id, componentType: SERVICE, domain: 5GEVE}])
Store VSB
VSB_ID

FIGURE 4: WORKFLOW FOR THE DESIGN OF A VERTICAL SERVICE DEPLOYED IN 5G-EVE PLATFORM

Once the blueprints are onboarded in both 5G-EVE portal and 5Gr-VS catalogues, the vertical can request the instantiation of the desired vertical service. As usual in the 5Growth platform, the first step is to define the VSD describing the requirements of the service (see Figure 5). The composite VSD must specify the values for all the parameters declared in the composite VSB, some of which, as explained before, map the parameters of the subservice. Following this mapping, the 5Gr-VS catalogue builds also the descriptor of the subservice (VSD-A in this example) and stores it in the internal repository.

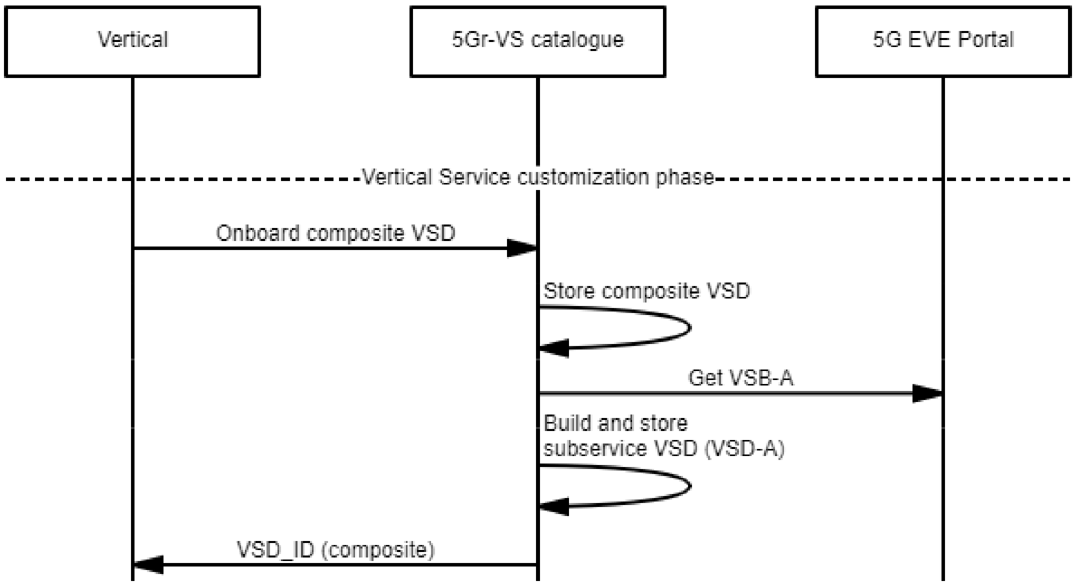

FIGURE 5: WORKFLOW FOR THE CUSTOMIZATION OF A VERTICAL SERVICE DEPLOYED IN 5G-EVE PLATFORM

The instantiation phase, depicted in Figure 6, begins with the vertical service instantiation request issued by the vertical to the 5Gr-VS. The request, as previously discussed, can include parameters related to the execution of the experiment, encoded in the configuration parameters to be provided by the vertical. Following the definition of the composite service available in the blueprint, the VS Lifecycle Manager (VS LCM) identifies the need to interact with the external 5G-EVE domain to delegate the instantiation of the subservice. Therefore, the VS LCM communicates with the *VsmfInteractionHandler* requesting a service modeled according to the VSD "VSD-A", to be instantiated in the 5G-EVE domain and to be configured with the set of configuration parameters provided by the vertical in the initial request. Based on the target domain, the *VsmfInteractionHandler* forwards the request to the proper driver. At the 5G-EVE Vertical Service Management Function (VSMF) driver, the parameters of the incoming request are translated into an Experiment Descriptor (ExpD) including the VSD-A itself, which is then onboarded into the 5G-EVE Portal catalogue.

The onboard ExpD request is mapped into the corresponding HTTP request defined in the 5G-EVE Portal OpenAPI[1]. As for all the HTTP messages exchanged with the 5G-EVE Portal, the request is authorized using a token. The token is retrieved and periodically updated interacting with the 5G-EVE Role-Based Access Control (RBAC) component. In the workflow represented in the picture this interaction is omitted for simplicity, but it is required whenever the REST clients need to acquire a new token or refresh an existing one.

[1] https://github.com/5GEVE/OpenAPI/tree/master/Portal

After the ExpD onboarding step, the 5G-EVE VSMF Driver uses the ExpD identifier returned by the 5G-EVE Portal to request the scheduling of the experiment. By default, the experiment is scheduled starting from the current time. However, the vertical can declare a different scheduling time using the configuration parameters of the instantiation request. The 5G-EVE Portal returns the unique identifier assigned to the experiment (called experiment_ID in the picture). This ID is in turn assigned by the 5G-EVE VSMF Driver to the subservice instance (VS-A_ID) and returned back to the *VsmfInteractionHandler* and, from here, to the VS LCM where it is stored in the record of the vertical service instance.

After the scheduling, the 5G-EVE VSMF Driver follows automatically all the phases of the 5G-EVE experiment, triggering the subsequent requests according to the evolution of the experiment status which is retrieved through periodical polling messages. In particular, when the experiment moves to the “READY” status, the 5G-EVE VSMF Driver requests its deployment in the environment prepared by the 5G-EVE site manager(s). This command triggers the instantiation of all the Network Service(s) associated to the vertical service in the 5G-EVE platform, with the proper deployment flavor selected based on the ExpD parameters.

At the end of the experiment instantiation, signaled by the “INSTANTIATED” status, the 5G-EVE VSMF Driver triggers the execution of the experiment. This request may include optional parameters related to the selection and configuration of the test cases, if provided by the Vertical in the configuration parameters. When the experiment moves in the “RUNNING_EXECUTION” status, the 5G-EVE VSMF Driver notifies the *VsmfInteractionHandler* about the change in the status of the subservice, which is reported as instantiated to the VS LCM component and updated accordingly in the vertical service record. It should be noted that the execution of the experiment on the 5G-EVE platform also starts the collection of the metrics and KPIs declared in the blueprint, which can be used to evaluate the performance of the vertical service. These metrics and KPIs can be retrieved from the 5G-EVE Portal GUI, visualizing their graphs or downloading the corresponding values in CSV files.

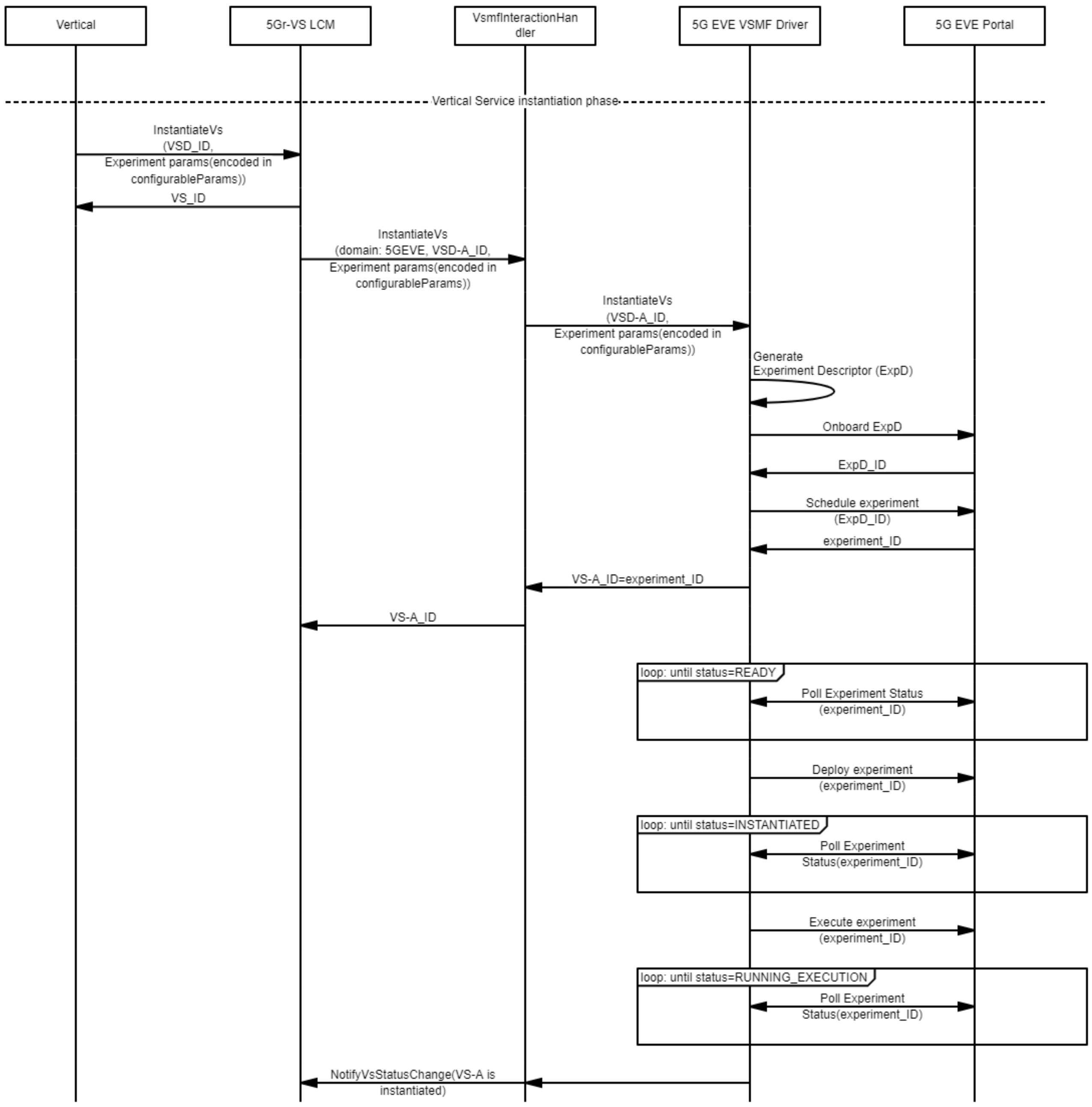
Vertical
5Gr-VS LCM
VsmfInteractionHandler
5G EVE VSMF Driver
5G EVE Portal
Vertical Service instantiation phase
InstantiateVs (VSD_ID, Experiment params(encoded in configurableParams))
VS_ID
InstantiateVs (domain: 5GEVE, VSD-A_ID, Experiment params(encoded in configurableParams))
InstantiateVs (VSD-A_ID, Experiment params(encoded in configurableParams))
Generate Experiment Descriptor (ExpD)
Onboard ExpD
ExpD_ID
Schedule experiment (ExpD_ID)
experiment_ID
VS-A_ID=experiment_ID
VS-A_ID
loop: until status=READY
Poll Experiment Status (experiment_ID)
Deploy experiment (experiment_ID)
loop: until status=INSTANTIATED
Poll Experiment Status(experiment_ID)
Execute experiment (experiment_ID)
loop: until status=RUNNING_EXECUTION
Poll Experiment Status(experiment_ID)
NotifyVsStatusChange(VS-A is instantiated)

FIGURE 6: WORKFLOW FOR THE INSTANTIATION OF A VERTICAL SERVICE IN 5G-EVE PLATFORM

Figure 7 shows the termination phase, which is initiated by the vertical requesting the termination of the composite vertical service from the 5Gr-VS NBI. The VS LCM identifies the subservice corresponding to the composite service instance and invokes the *VsmfInteractionHandler* to terminate it. The request is forwarded to the 5G-EVE VSMF Driver, which in turn sends the HTTP request to the 5G-EVE Portal and starts the polling queries until the experiment status becomes "TERMINATED". The change of the status is reported back to the *VsmfInteractionHandler* and, from here, to the VS LCM that updates the corresponding entry in the vertical service records.

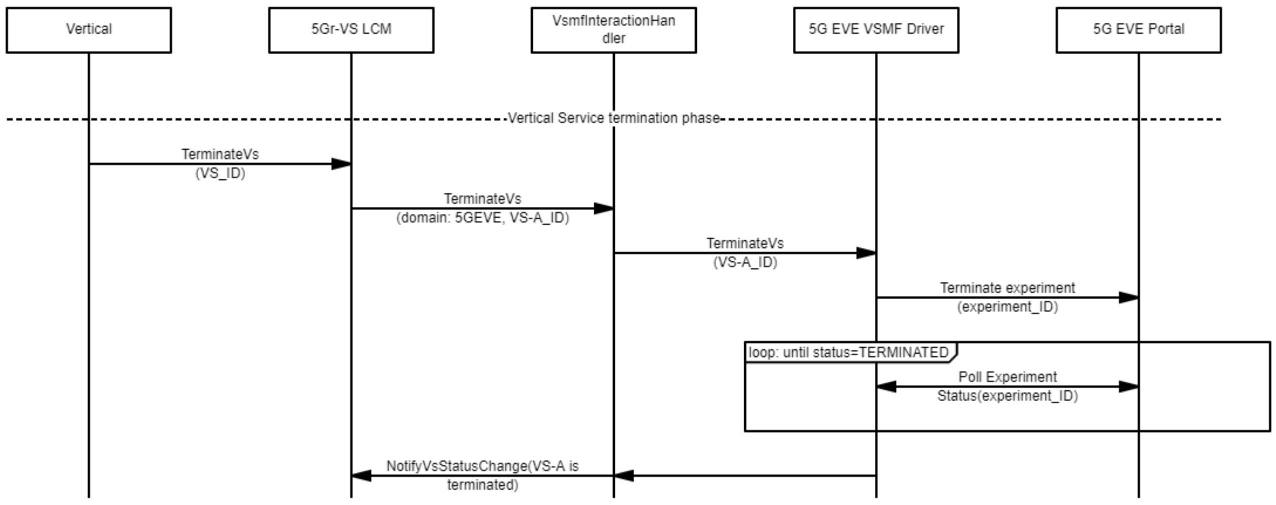
Vertical
5Gr-VS LCM
VsmfInteractionHandler
5G EVE VSMF Driver
5G EVE Portal
Vertical Service termination phase
TerminateVs (VS_ID)
TerminateVs (domain: 5GEVE, VS-A_ID)
TerminateVs (VS-A_ID)
Terminate experiment (experiment_ID)
loop: until status=TERMINATED
Poll Experiment Status(experiment_ID)
NotifyVsStatusChange(VS-A is terminated)

FIGURE 7: WORKFLOW FOR THE TERMINATION OF A VERTICAL SERVICE IN 5G-EVE PLATFORM

2.1.3. Implementation

In terms of implementation, the interaction between the 5Gr-VS and the 5G-EVE platform is enabled through a dedicated VSMF (Vertical Service Management Function) driver, the 5G-EVE VSMF Driver, integrated with the 5Gr-VS software, as represented in Figure 8. In particular, the picture highlights in yellow the components updated from the previous 5Gr-VS version and in green the new components introduced to handle the interaction with 5G EVE Portal.

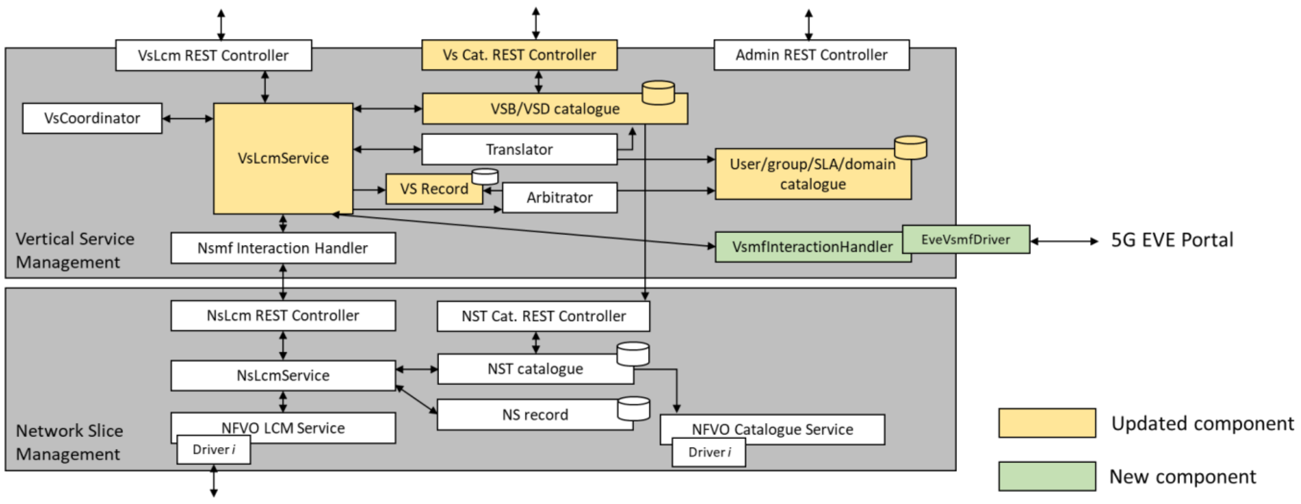

FIGURE 8: 5GR-VS SOFTWARE ARCHITECTURE

.

The 5G EVE VSMF driver has been already integrated in the 5Gr-VS and it is available in the 5Gr-VS code[2], in the `it.nextworks.nfvmano.sebastian.vsfm.sbi.vsmf.drivers` package. In detail, it is

[2] https://github.com/5growth/5gr-vs

implemented through the *EveVsmfDriver* class, extending the *AbstractVsmfDriver* abstract class, which provides a generalized model for the communication with external domains providing VSMF capabilities. The *AbstractVsmfDriver* class implements the *VsLcmProviderInterface* java interface, which defines the methods of the internal NBI exposed by the 5Gr-VS CSMF component.

The 5Gr-VS instantiates a new *EveVsmfDriver* through the *VsmfInteractionHandler* (available in the `it.nextworks.nfvmano.sebastian.vsfm.sbi.vsmf` package), which is the entity that manages all the drivers to interact with external VSMF-like domains. The *VsmfInteractionHandler* offers a single access point for the other internal modules of the 5Gr-VS (e.g. the Vertical Service Lifecycle Manager) for requesting actions into external VSMF-like domain, wrapping their particular interfaces. The configuration of these external domains is kept centralized in the Domain Catalogue component, which is queried by the *VsmfInteractionHandler* at the bootstrap of the 5Gr-VS to determine the drivers to be instantiated and their parameters (e.g. type, target URL, capabilities, etc.). The *VsmfInteractionHandler* activates also a subscription with the 5Gr-VS Domain Catalogue, so that new drivers can be dynamically instantiated whenever a new domain is added in the catalogue from the 5Gr-VS management interface.

Internally, the 5G-EVE VSMF Driver acts as REST client of the 5G-EVE Portal in order to perform the following actions:

- Request the on-boarding of an experiment descriptor (ExpD) corresponding to the vertical service to be deployed in the 5G-EVE platform;
- Request the instantiation of an experiment based on the given ExpD;
- Request the execution of the experiment, providing as optional parameters the details of the test cases to be executed;
- Query the status and the information of the experiment;
- Request the termination of the experiment.

These actions are performed through REST clients that have been autogenerated from the OpenAPI of the 5G-EVE Portal. The REST clients are available in the 5Gr-VS software repository under the 5GEVE_REST_CLIENT folder, together with their original OpenAPI specifications. In detail, the following REST client are used:

- 5GEVE_RBAC_LOGIN_REST_CLIENT, used to interact with the Role Based Access Control (RBAC) component of the 5G-EVE Portal for authentication and authorization issues. The driver initially communicates with the 5G-EVE RBAC providing the credentials, in order to retrieve a token that is used in all the following interactions.
- 5GEVE_PORTAL_CATALOGUE_REST_CLIENT, used to retrieve VSBs and ExpBs available in the 5G-EVE platform and to onboard the ExpD corresponding to the user's vertical service request on the 5Gr-VS NBI.
- 5GEVE_EXPERIMENT_LCM_REST_CLIENT, used to request lifecycle-related commands on the experiment, i.e. instantiation, queries, execution, termination.

2.2. 5G-EVE IWL driver

The 5G-EVE Interworking Layer (IWL) 5Growth Service Orchestrator (5Gr-SO) driver is part of the South Bound Interface (SBI) of 5G-EVE IWL component named the adaptation layer, this component in charge of abstracting the on-boarding and lifecycle management capabilities from multiple NFV Orchestrators (NFVOs). The adaptation layer driver exposes to 5G-EVE upper layers an ETSI SOL005 [3] REST interface, and the 5Gr-SO exposes an ETSI IFA013 [4] REST interface. Rising the necessity of integration between these two interfaces, there should be a coherent translation between both interfaces. The integration point between both domains is the 5G-EVE Multi-Site NSO to Local Orchestrator (MSO-LO) REST API application, which will be the component in charge of translating ETSI IFA013 and ETSI SOL005 data models. Figure 9 depicts where the 5G-EVE IWL driver component is going to be located in the architecture.

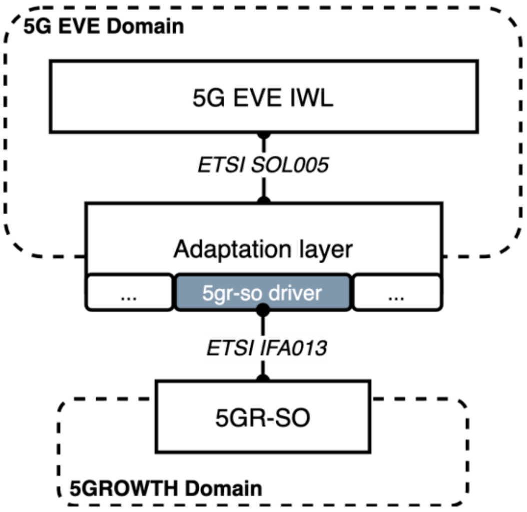

FIGURE 9: 5G-EVE IWL DRIVER COMPONENTS LOCATION

2.2.1. IWL catalogue

2.2.1.1. General description

The 5G-EVE IWL catalogue is the functional element of the 5G-EVE platform responsible for the management of Network Service Descriptors (NSD) and VNF packages and their synchronization with the catalogues of the specific NFVOs deployed in each 5G-EVE site facility. The 5G-EVE IWL catalogue exposes at its NBI a REST API compliant with the ETSI NFV SOL 005 interface, adopting the TOSCA based data models for NSD and VNF packages, as defined in the ETSI NFV SOL 001 [5] specification. These data models are based on the OASIS TOSCA Simple Profile in YAML and they are used to store internally in the IWL catalogue all the NSDs and VNF packages available in the NFVOs at the various 5G-EVE facilities using a unified, cross-site model. Since the NFVOs may adopt different kinds of information models, the IWL catalogue needs to translate between the TOSCA-based model used internally and the site-specific information models used in the target NFVOs (and vice versa).

This translation is performed through a number of NFVO drivers, e.g. the OSM and the ONAP drivers already available in the 5G-EVE IWL catalogue. In the case of 5Growth sites, a new IWL catalogue driver is required to perform the translation between the TOSCA-based models and ETSI NFV IFA 014 [6] and IFA 011 [7] information models adopted by the 5Gr-SO for Network Service Descriptors (NSDs) and VNF Descriptors (VNFDs) respectively.

2.2.1.2. Services and workflows

The IWL catalogue driver developed in 5Growth supports the following functionalities:

- **On-boarding of NSDs**, received at the IWL catalogue by the 5G-EVE Portal in TOSCA format, into the 5Gr-SO catalogue. This on-boarding is performed whenever an experiment developer uploads a new service or context blueprint into the 5G-EVE Portal, together with the corresponding NSD. While the blueprint is stored in the 5G-EVE Portal catalogue, the NSD is forwarded to the IWL catalogue and from here to NFVO at the target site (see Figure 10). In this case, the IWL catalogue driver needs to perform a translation from the ETSI NFV SOL 001 format into the ETSI NFV IFA 014 format.
- **Retrieval of all the NSDs available at the 5Gr-SO catalogue.** This action is performed at the start-up of the IWL catalogue and it is repeated periodically during its operational time in order to synchronize the list of NSDs with the remote 5Gr-SO catalogue and to update the local repository accordingly (see Figure 11). The required information model translation is from the ETSI NFV IFA 014 format into the ETSI NFV SOL 001 format.
- **Retrieval of all the VNFDs available at the 5Gr-SO catalogue**. This action is performed at the start-up of the IWL catalogue and it is repeated periodically during its operational time in order to synchronize the list of VNFDs with the remote 5Gr-SO catalogue (see Figure 12). It should be noted that in the 5G-EVE model VNF packages can be onboarded only from the site catalogues, since this procedure requires a manual verification of the VNF images to be compliant with the site administration policies. For this reason, in the management of VNF packages the NFVOs at each site act as "master", while the IWL catalogue operates just in read mode. Therefore, it needs to poll periodically the NFVOs to become aware of new VNF packages made available in the remote sites. In this case, the IWL driver needs to translate from the ETSI NFV IFA 011 format to the ETSI NFV SOL 001 format.

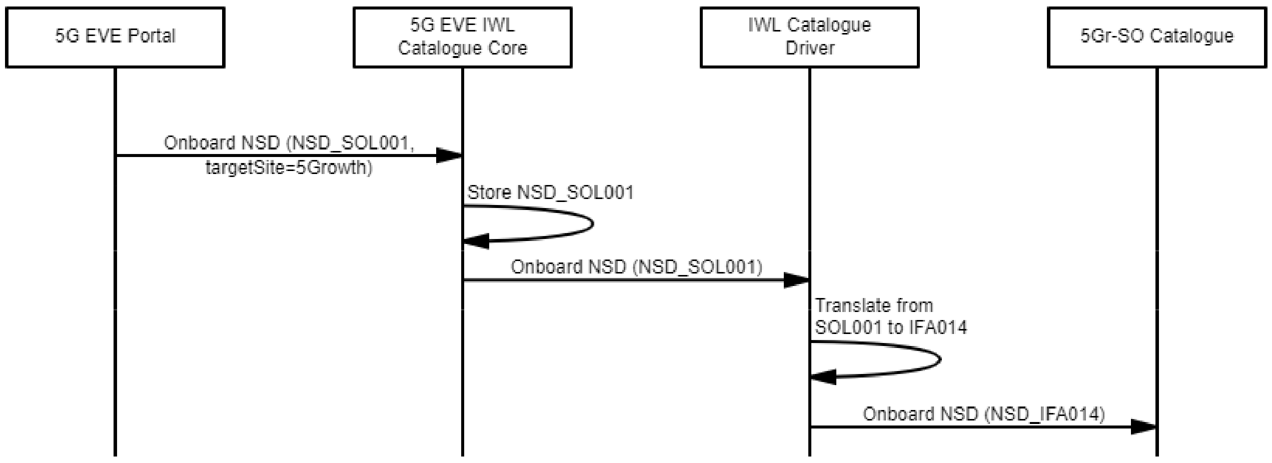

FIGURE 10: WORKFLOW FOR THE ONBOARDING OF A NSD IN 5G-EVE PLATFORM AND A 5GROWTH SITE

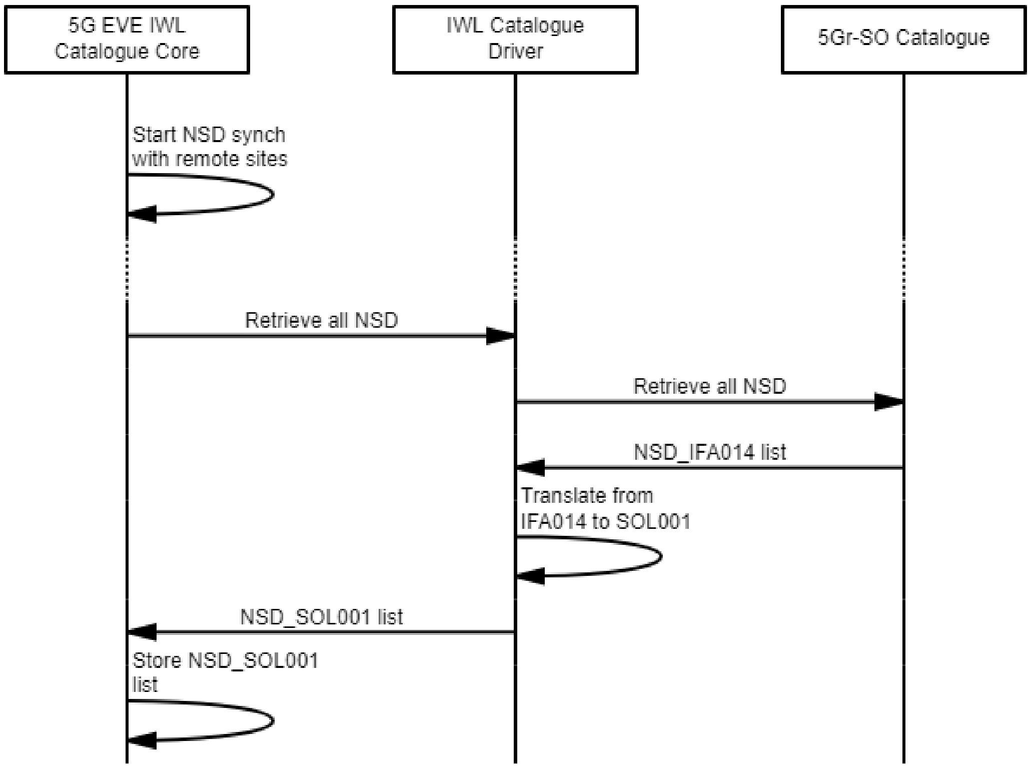

FIGURE 11: WORKFLOW FOR NSD SYNCHRONIZATION BETWEEN 5G-EVE PLATFORM AND 5GROWTH SITE

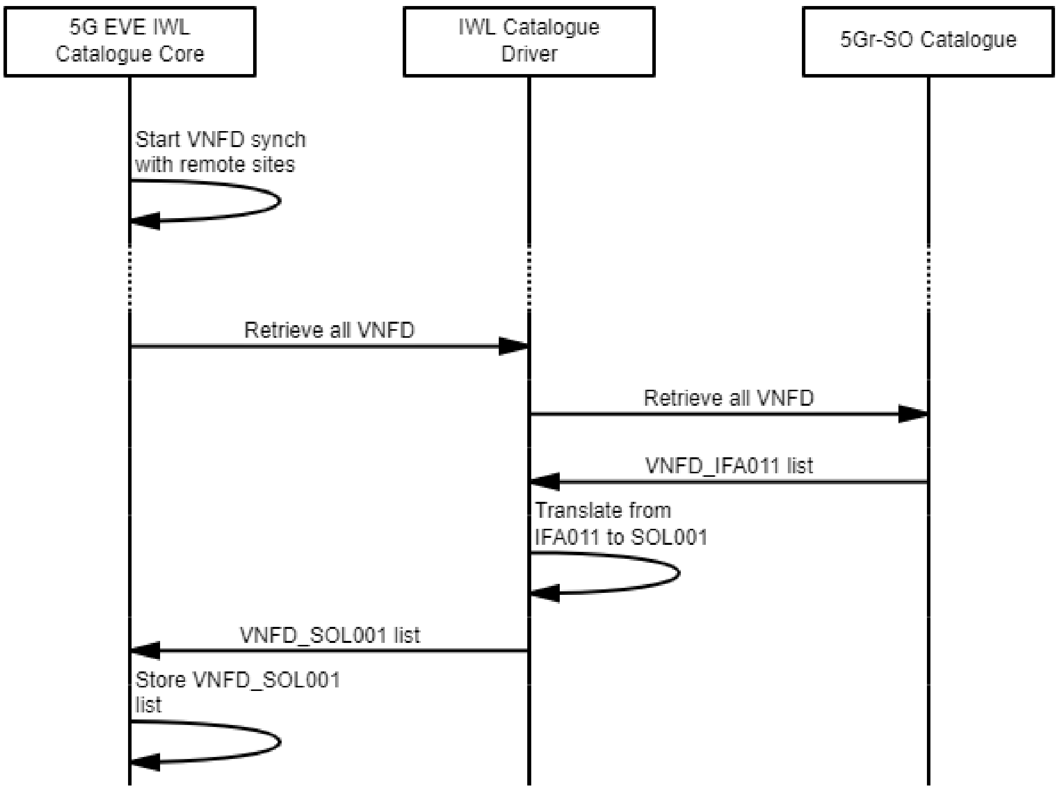

FIGURE 12: WORKFLOW FOR VNFD SYNCHRONIZATION BETWEEN 5G-EVE PLATFORM AND 5GROWTH SITE

It is worth to mention that only the first functionality is aligned with the mechanisms originally supported in the interaction between the 5Gr-VS and the 5Gr-SO, where the 5Gr-VS acts as originator of all the NSD/VNFD management actions. The support of the synchronization mechanism, based on the retrieval of the entire list of NSDs and VNFDs available in the 5Gr-SO catalogue, has required an extension in the functionalities and in the NBI REST API of the 5Gr-SO itself.

2.2.1.3. Implementation

The implementation of the IWL catalogue driver is based on the development of a dedicated plugin developed as a *Maven* artifact and integrated into the IWL catalogue. The java classes are implemented in the *it.nextworks.nfvmano.catalogue.plugins.mano.fivegrowthCataloguePlugin* package. The plugin has been integrated in the 5G-EVE IWL Catalogue through a dependency in its pom file and minor modifications in its core software to instantiate the plugin and interconnect it to the rest of the system. The plugin is made available in the folder *FivegrowthCataloguePlugin* of the public IWL catalogue repository[3] in the 5Growth branch.

In terms of software structure, the plugin is implemented through the *SOPlugin* class. It extends the *MANOPlugin* abstract class offered by the catalogue, which provides all the internal mechanisms to

[3] https://github.com/nextworks-it/5g-catalogue Note: in "5Growth branch"

interact with the core of the IWL catalogue through its internal Kafka bus. In detail, the 5GrowthCataloguePlugin implements the following methods:

- *acceptNsdOnboardingNotification*, used to onboard new NSD received from the IWL catalogue core into the 5Gr-SO catalogue;
- *getAllNsd*, used to retrieve all the NSDs from the 5Gr-SO catalogue;
- *getAllVnfd*, used to retrieve all the VNFDs from the 5Gr-SO catalogue.

The plugin interacts with the 5Gr-SO catalogue through a REST client, implemented in the class *SODriver*, while the translation between TOSCA and ETSI NFV IFA information models is handled in the classes *IfaToSolTranslator* (from IFA to TOSCA) and *SolToIfaTranslator* (from TOSCA to IFA). The new endpoints of the NBI REST are presented in the following table.

TABLE 2: NEW 5GR-SO NBI ENDPOINTS

Name	Method	Endpoint	Description
Get NSDs	GET	/ns/nsd	Returns associated information of all NSDs onboarded in the 5Gr-SO.
Get VNFDs	GET	/ns/vnfd	Returns associated information of all VNFDs onboarded in the 5Gr-SO.

2.2.2. IWL Lifecycle Manager

2.2.2.1. General description

The main lifecycle management capabilities that are exposed by the MSO-LO driver are listed in Table 3. These capabilities are necessary to manage the lifecycle of network services. Currently, this interface abstracts some of the most common functionalities that most NFVO LCMs expose, such as the creation of an NS from an NSD id, the instantiation, the scaling and the termination of an NS.

TABLE 3: 5G-EVE MSO-LO DRIVER CAPABILITIES

Capability	Supported	Integration Tests	Description
get_ns_list	NO	N/A	Retrieves a list of network services from the underlying NFVO.
create_ns	YES	YES	Creates a network service identifier from the underlying NFVO.
get_ns	YES	YES	Retrieves a single network service by its id from the underlying NFVO.
delete_ns	NO	N/A	Deletes the network service by its id from the underlying NFVO.
instantiate_ns	YES	YES	Instantiates an already created network service in the underlying NFVO.
terminate_ns	YES	YES	Terminates an instantiated network service in the underlying NFVO.

scale_ns	YES	NO	Scales an instantiated network service in the underlying NFVO.
get_op_list	NO	N/A	Retrieves from the underlying NFVO a list of the current operations.
get_op	YES	YES	Retrieve an operation status from the underlying NFVO by id.

Table 3, describes most of the capabilities exposed by the MSO-LO 5Gr-SO driver, if it is supported, and if it has been tested against the 5Gr-SO. Some capabilities such as the *get_ns_list*, *delete_ns* are supported by ETSI IFA013 but are not supported by the 5Gr-SO due to some limitations in its NBI. Moreover, the capability *get_op_list* is not supported by ETSI IFA013 standard and consequently, the 5Gr-SO does not support this capability. Overall, the unsupported capabilities are not critical to managing the lifecycle of the network services because they are mostly non-functional requirements. Finally, the capability of scaling is supported by IFA013 and the 5Gr-SO, so it has been integrated into the driver, however, this capability has not been tested.

2.2.2.2. Services and Workflows

Figure 13, represents the workflow to create a new network service (by assigning it a new identifier), to instantiate it, and to verify using the operation identifier if the network service operation has been completed. Additionally, Figure 13 also represents the workflow to terminate an already instantiated network service. Figure 14, represents the workflow of scaling a previously instantiated network service. Notice that the adaptation layer and the 5Gr-SO do not notify the 5G-EVE IWL and the MSO-LO respectively. When an operation has been completed, it is the 5G-EVE IWL the component in charge of polling the adaptation layer and retrieving the operation status.

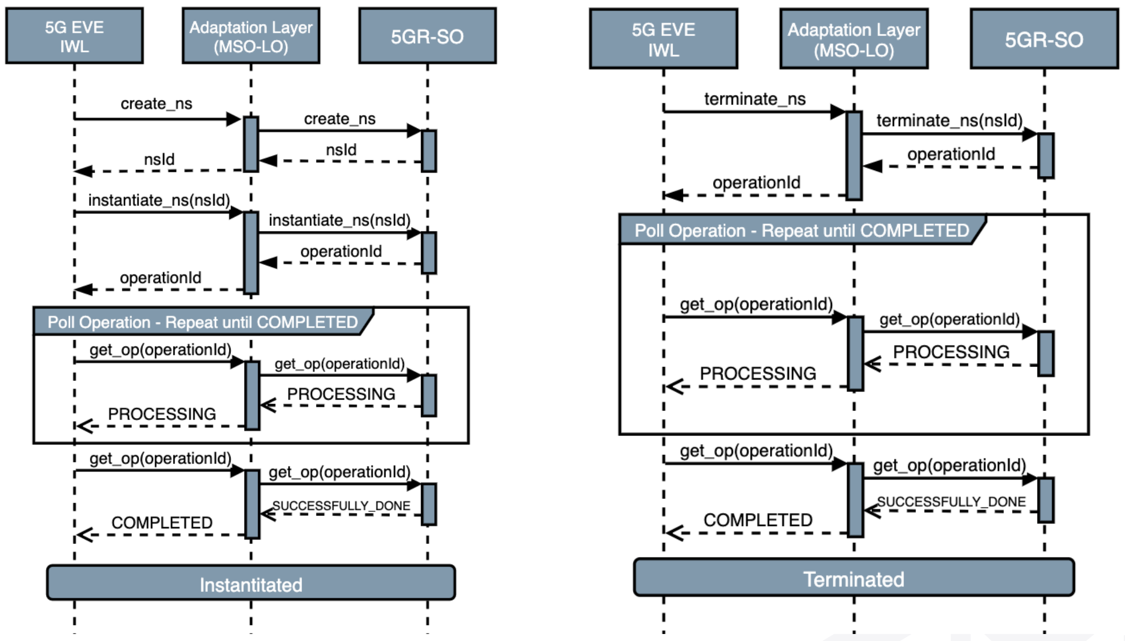

FIGURE 13: 5G-EVE IWL DRIVER WORKFLOWS (CREATE, INSTANTIATE AND TERMINATE)

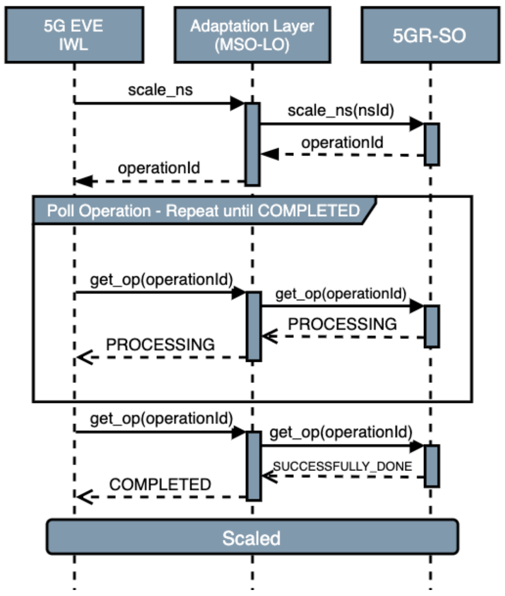

FIGURE 14: 5G-EVE IWL DRIVER WORKFLOWS (SCALE)

Network Service Lifecycle high-level workflows and data model mapping between ETSI SOL005 and ETSI IFA013 can be found in Table 4, which shows a detailed description of the process and data model translation for each type of action.

TABLE 4: 5G-EVE IWL DRIVER DATA MODEL TRANSLATION

Operation	Workflow and Data Model Translation
Create	5G-EVE IWL creates using an ETSI SOL005 *CreateNsRequest* a new network service from an already onboarded NSD, the adaptation layer translates such request into an IFA013 *CreateNsIdentifierRequest*, its response *CreateNsIdentifierResponse* will be mapped to ETSI SOL005 *NsInstance* data model.
Instantiate	5G-EVE IWL instantiates using ETSI SOL005 *InstantiateNsRequest* the network service, the adaptation layer translates such request into IFA013 *InstantiateNsRequest*, its response *InstantiateNsResponse* will be mapped to ETSI SOL005 response which has to include an HTTP Location header containing the identifier of the Lifecycle operation (operation id).
Status Polling	With the operation id, the 5G-EVE IWL will poll the adaptation layer, the adaptation layer will translate ETSI SOL005 poll request into an IFA013 *GetOperationStatusRequest* which its response *GetOperationStatusResponse* will be mapped to ETSI SOL005 *NsLcmOpOcc*. The 5G-EVE IWL will stop polling the adaptation layer once an end state (COMPLETED or FAILED) is reached.
Terminate	With a network service instantiated, the 5G-EVE IWL can terminate it by using an ETSI SOL005 *TerminateNsRequest*, the adaptation layer will then translate such request into an IFA013 *TerminateNsRequest*, its response *TerminateNsResponse* will be mapped to ETSI SOL005 response which has to include an HTTP Location header containing the identifier of the Lifecycle operation (operation id).
Scale	In case the network service wants to be scaled, the 5G-EVE IWL can scale it by using an ETSI SOL005 *ScaleNsRequest*, the adaptation layer will translate such request into ETSI IFA013 *ScaleNsRequest*, its response *ScaleNsResponse* will be mapped to

	ETSI SOL005 response which has to include an HTTP Location header containing the identifier of the Lifecycle operation (operation id).

2.2.2.3. Implementation

The implementation of the 5G-EVE IWL driver requires the API translation between two standard interfaces, ETSI IFA013 and ETSI SOL005. The implemented driver is capable of translating requests from the NBI of the 5G-EVE IWL MSO-LO which follows ETSI SOL005 (Table 5), to the NBI of the 5Gr-SO which follows ETSI IFA013 (Table 6).

TABLE 5: NBI OF 5G-EVE IWL MSO-LO (ETSI SOL005)

Name	Method	Endpoint	Description
Create NS Id	POST	/nfvo/{nfvoId}/ns_instances	Creates a new NS instance Identifier.
Get NS	GET	/nfvo/{nfvoId}/ns_instances/{nsInstanceId}	Returns the information of the network service referenced by *nsId*.
Instantiate NS	POST	/nfvo/{nfvoId}/ns_instances/{nsInstanceId}/instantiate	Instantiate the Network Service identified by *nsId*.
Scale NS	POST	/nfvo/{nfvoId}/ns_instances/{nsInstanceId}/scale	Scale the Network Service identified by *nsId*.
Terminate NS	POST	/nfvo/{nfvoId}/ns_instances/{nsInstanceId}/terminate	Terminated the Network Service identified by *nsId*.
Get Operation Status	GET	/nfvo/{nfvoId}/ns_lcm_op_occs/{nsLcmOpOccId}	Returns the status of an operation by its operation identifier.

TABLE 6: NBI OF 5GROWTH 5GR-SO (ETSI IFA013)

Name	Method	Endpoint	Description
Create NS Id	POST	/ns	Creates and returns a Network Service Identifier.
Get NS	GET	/ns/{nsId}	Returns the information of the network service referenced by *nsId*.
Instantiate NS	PUT	/ns/{nsId}/instantiate	Instantiate the Network Service referenced by *nsId*.
Scale NS	PUT	/ns/{nsId}/scale	Scales the Network Service referenced by *nsId*.
Terminate NS	PUT	/ns/{nsId}/terminate	Terminated the Network Service identified by *nsId*.
Get Operation Status	GET	/operation/{operationId}	Returns the status of an operation by its operation identifier.

The code developed for the 5G-EVE adaptation layer can be found on GitHub[4] under the MSO-LO official code repository. All changes to add the capability of managing the lifecycle of network services in 5G-SO are located at the master branch, and included in the latest release of the component. Additionally, the 5G-EVE MSO-LO has been deployed as a docker image at docker hub[5]. Table 7, summarizes and maps the source code, the docker image and the interfaces documentation to their corresponding remote locations.

TABLE 7: 5G-EVE IWL DRIVER RESOURCES

Resource	**Location**
Source Code	https://github.com/5growth/mso-lo
Latest Release	https://github.com/5growth/mso-lo/releases/tag/v1.4.0
Docker Image	https://hub.docker.com/r/5geve/mso-lo
Documentation	https://github.com/5growth/mso-lo/blob/master/README.md
ETSI SOL 005 (NBI Specification)	https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/02.06.01_60/gs_NFV-SOL005v020601p.pdf
ETSI IFA 013 (SBI Specification)	https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/013/03.04.01_60/gs_NFV-IFA013v030401p.pdf

2.3. 5Gr-VS driver towards 5G-VINNI

2.3.1. General description

This driver allows the deployment of a slice of a complex network service on SONATA ordered by 5Gr-VS. For that, 5Gr-VS needs to fulfill a data model compliant with VS descriptors, in such a way that they may be translatable into a SONATA data model compliant with SONATA descriptors. The driver implements HTTP requests to manage the lifecycle of network services, to retrieve information from the service, and to change their configuration. The driver implements the interface *NsmfLcmProviderInterface* furnished by the 5Gr-VS suite. This interface defines which operations to override by each driver, providing a common set of operations each time the 5Gr-VS needs to interact with other external systems in order to deploy and manage slices of complex network services.

The VS Blueprint shown in Figure 15 provides information related to high-level definitions needed by the 5Gr-VS to manage the target service.

[4] https://github.com/5growth/mso-lo
[5] https://hub.docker.com/r/5geve/mso-lo

```
{
  "vsBlueprint": {
    "version": "1.1",
    "name": "test_sonata",
    "description": "test_sonata"
  },
  "nsts": [
    {
      "nstId": "end_to_end_slice",
      "nstName": "end_to_end_slice",
      "nstVersion": "0.2",
      "nstProvider": "ALB",
      "nsdId": "e2e_nsd",
      "nsdVersion": "0.1",
      "nsstIds": [
        "536fa477-119e-4fea-add5-05afc5b53e33"
      ]
    },
    {
      "nstId": "536fa477-119e-4fea-add5-05afc5b53e33",
      "nstName": "536fa477-119e-4fea-add5-05afc5b53e33",
      "nstVersion": "0.1",
      "nstProvider": "ALB",
      "nsstIds": []
    }
  ]
}
```

FIGURE 15: EXAMPLE OF VSB FOR 5GR-VS CATALOGUE WITH SONATA NSTIDS

2.3.2. Services and workflows

These operations mainly forward information relevant to the management of services in 5Gr-VS format, as seen in Figure 16. It is expected that this information will be transformed inside the SONATA adaptor to a format compliant for effective management of SONATA services before the operations are performed.

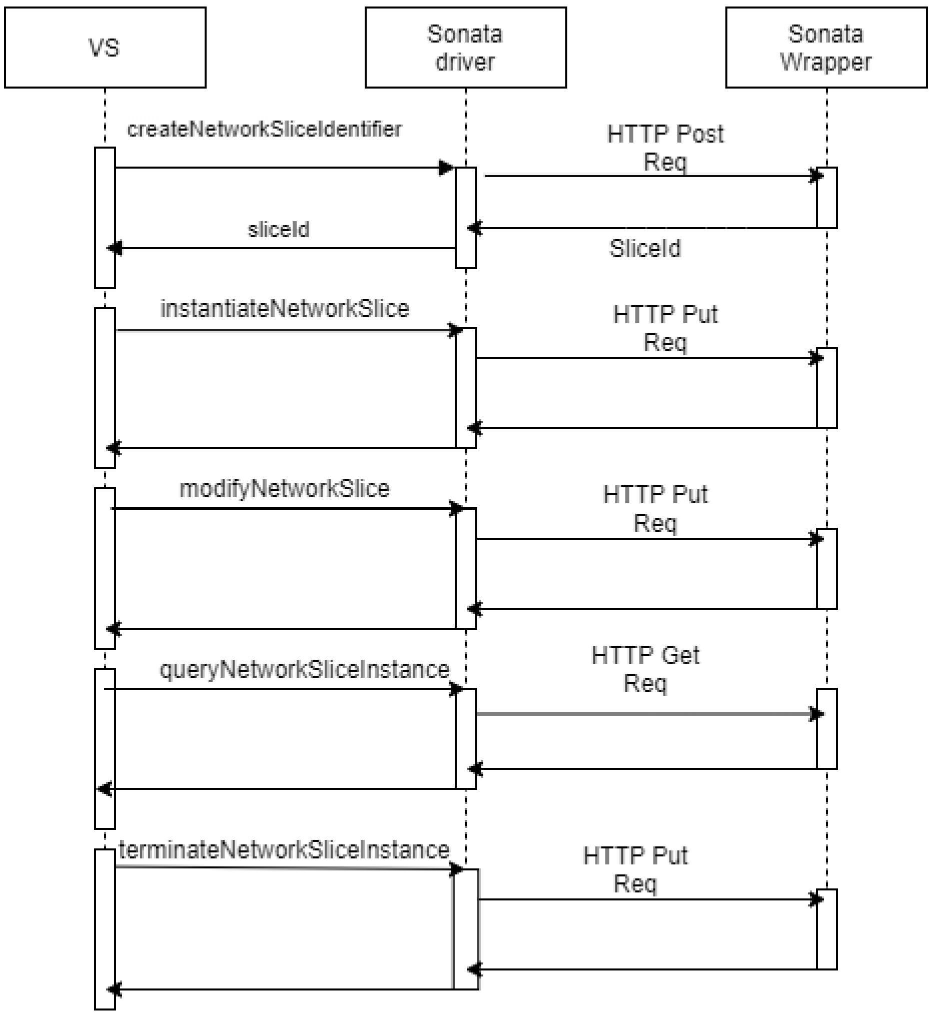

FIGURE 16: OPERATIONS SUPPORTED BY 5GR-VS DRIVER FOR SONATA

The operations implemented on this driver are:

- *createNetworkSliceIdentifier* – operation to obtain a unique slice identifier to be used by the VS representing the slice;
- *instantiateNetworkSlice* – operation used to deploy the network service slice on SONATA;
- *modifyNetworkSlice* – operation to modify the topology of the network service slice on SONATA;
- *terminateNetworkSliceInstance* – operation to un-deploy the service slice on SONATA;
- *queryNetworkSliceInstance* – operation to obtain selected data about the running slice network service on SONATA.

As we can see in the following Figure 17, we have depicted a vertical service instantiation workflow. It includes the initial request to obtain a valid identifier to the external system, in this case, SONATA, linked to the internal designation identifier. After successfully obtaining the identifier, this will be used to effectively initiate the desired network service or obtain a failure. The definitive prognosis will be attained by polling the external system, obtaining a code representing the final state of the service initiation.

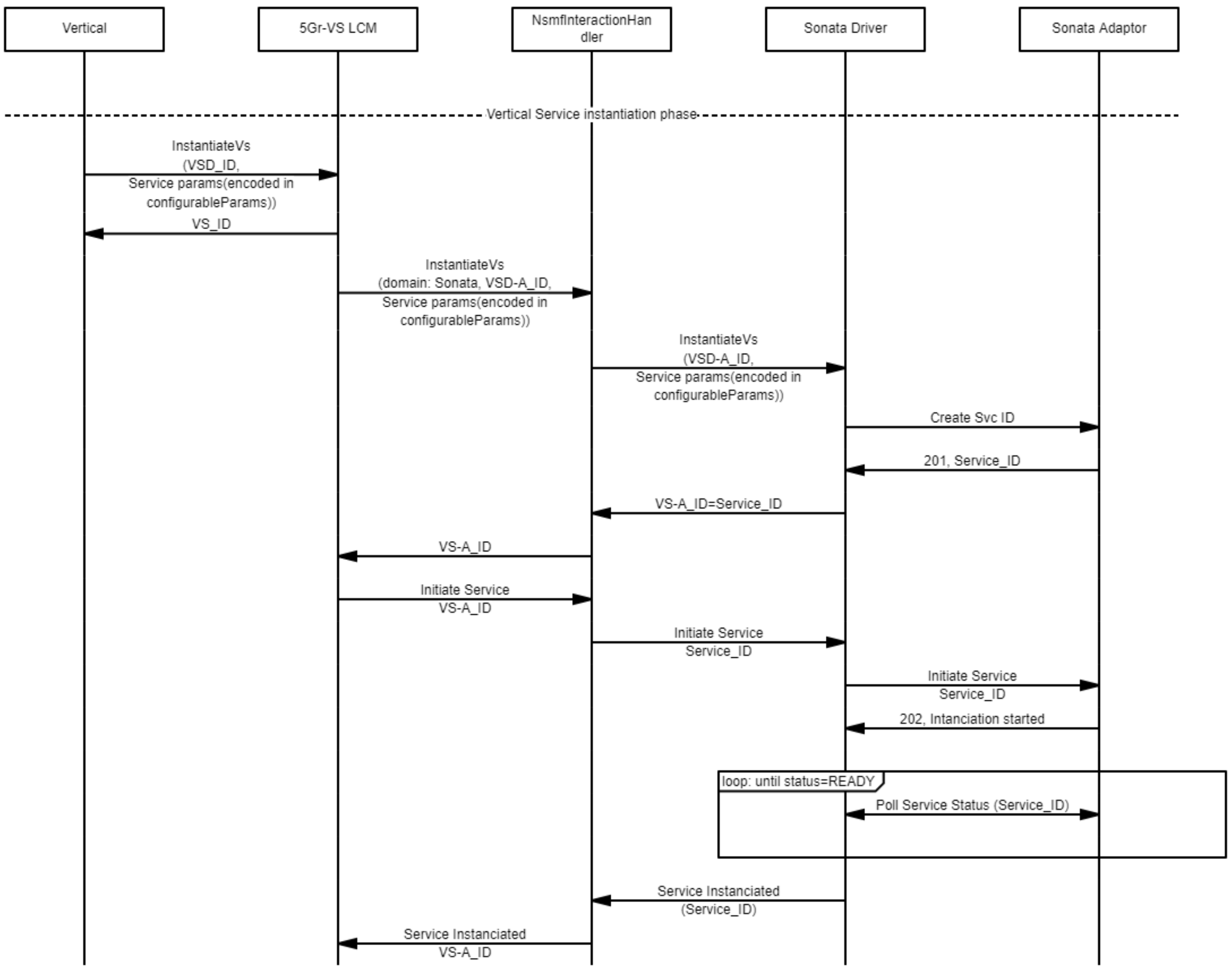
Vertical
5Gr-VS LCM
NsmfInteractionHandler
Sonata Driver
Sonata Adaptor
Vertical Service instantiation phase
InstantiateVs (VSD_ID, Service params(encoded in configurableParams))
VS_ID
InstantiateVs (domain: Sonata, VSD-A_ID, Service params(encoded in configurableParams))
InstantiateVs (VSD-A_ID, Service params(encoded in configurableParams))
Create Svc ID
201, Service_ID
VS-A_ID=Service_ID
VS-A_ID
Initiate Service VS-A_ID
Initiate Service Service_ID
Initiate Service Service_ID
202, Intanciation started
loop: until status=READY
Poll Service Status (Service_ID)
Service Instanciated (Service_ID)
Service Instanciated VS-A_ID

FIGURE 17: FLOWS FOR ID CREATION AND INSTANTIATION OF A SONATA SERVICE INSIDE 5GR-VS

The termination workflow follows the same logic, the identifier obtained in the initiation process is used to identify the service to terminate. The termination is achieved once again by polling the final status on the external system. This is illustrated in Figure 18.

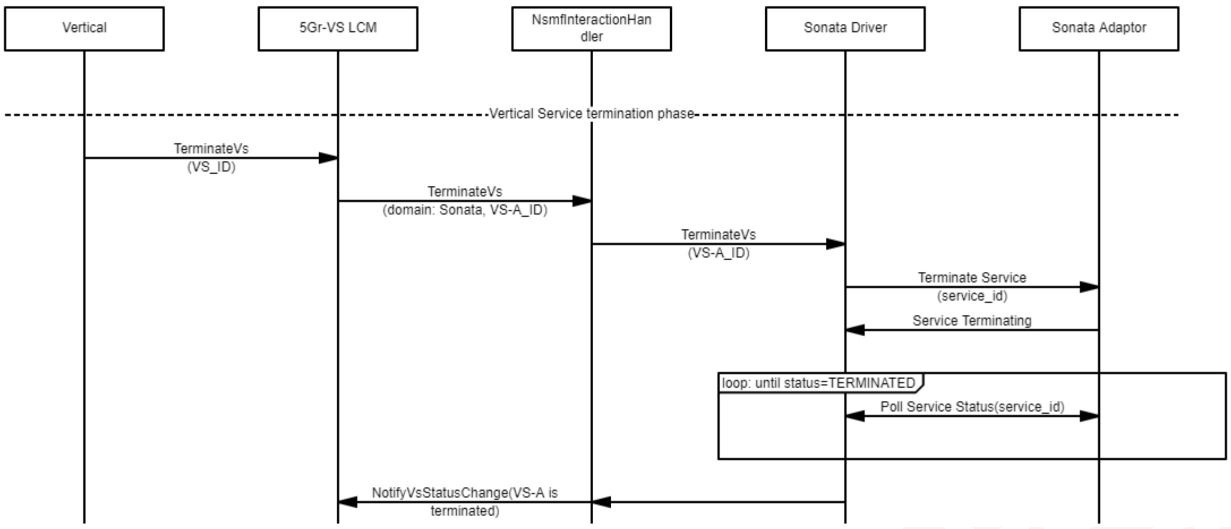
Vertical
5Gr-VS LCM
NsmfInteractionHandler
Sonata Driver
Sonata Adaptor
Vertical Service termination phase
TerminateVs (VS_ID)
TerminateVs (domain: Sonata, VS-A_ID)
TerminateVs (VS-A_ID)
Terminate Service (service_id)
Service Terminating
loop: until status=TERMINATED
Poll Service Status(service_id)
NotifyVsStatusChange(VS-A is terminated)

FIGURE 18: FLOWS FOR TERMINATION OF A SONATA SERVICE INSIDE 5GR-VS

2.3.3. Implementation

The implementation of the SONATA driver relies on the structure built inside the 5Gr-VS, namely the NSMF Interaction Handler. This handler has the responsibility to adapt drivers for external systems within the 5Gr-VS. It forwards information depicted on the descriptors related to the external system, which must be adapted in its destination to achieve the desired objectives. This is illustrated in Figure 19.

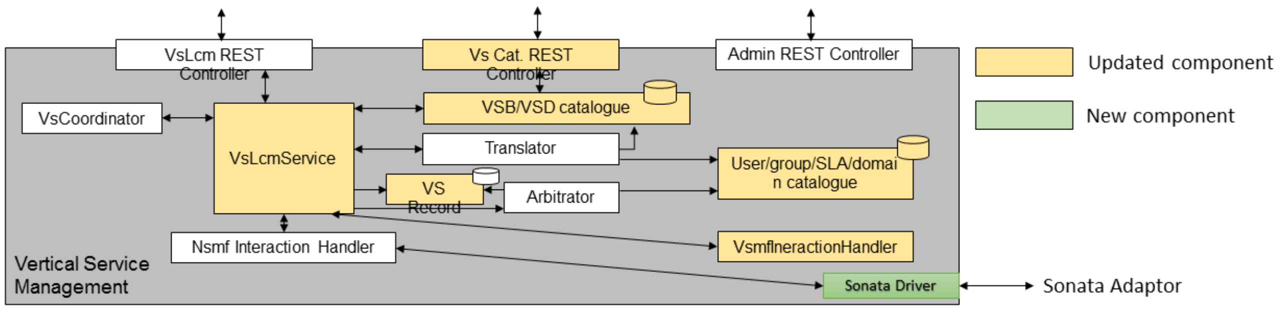

FIGURE 19: 5GR-VS SOFTWARE ARCHITECTURE FOR SONATA INTEGRATION

The data model used by the driver uses the descriptors defined within the 5Gr-VS suite:

- the VS Blueprint – high-level descriptor;
- the VSD descriptors – descriptor related with the slice;
- the VS *vnfd* descriptors – descriptor related with each virtual network function;
- the VS *nsd* descriptors – descriptor related to the network service;
- the VS *pnfd* descriptors – descriptors related to each physical deployment unit.

Although there is not a direct mapping between these descriptors and those belonging to SONATA, it is possible to achieve a transformation from one set to the other by following the similar functions each parameter has in each set of descriptors. Most of the time, the VS parameters within the set of descriptors will have no direct mapping on the SONATA descriptors, as can be seen in Annex 1: Descriptors mapping. The code resides on a git repository[6] inside the *NSMF_DRIVERS\5growthSonata* folder.

2.4. SONATA adaptor

2.4.1. General description

The purpose of this component is to make the connection between the Vertical Slicer in 5Growth Platform (5Gr-VS) and SONATA Platform.

This component has the following requirements:

[6] https://5growth.eu/redmine/projects/5growth/repository/5gr-vs

- To handle messages in format TS 28.531 [2] with Vertical Slicer;
- To handle messages with SONATA in the specific SONATA API format;
- To translate the format of the messages from/to TS 28.531 and SONATA API;
- To support the messages type: Instantiate, Terminate and Query Slice Instantiations;

This component was implemented in *python* language and the code can be found in github[7].

Figure 20 shows the high-level architecture.

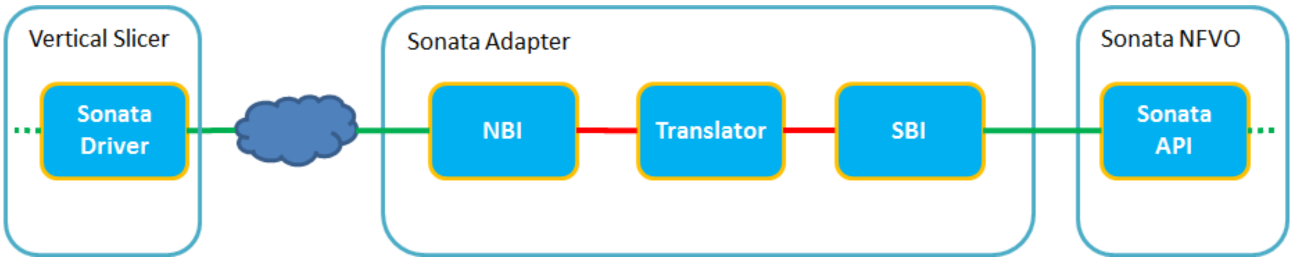

FIGURE 20: SONATA ADAPTER HIGH-LEVEL ARCHITECTURE

This architecture is mainly composed of three principal software blocks: the NBI (North bound API), the Translator and the SBI (South Bound API). The NBI is responsible to exchange REST messages with the 5Gr-VS in the format TS 28.531 and provide/receive them to/from the Translator. The Translator is responsible to change the format of the messages between format TS 28.531 and SONATA API, receive these messages from NBI and provide them to SBI (and vice versa). Finally, the SBI is responsible to exchange REST messages with SONATA API and provide/receive them to/from the Translator.

2.4.2. Services and workflows

Figure 21: High-Level Sequence DiagramFigure 21 shows the flow diagram of the Instantiation and Termination process.

[7] https://github.com/5growth/sonata-drivers

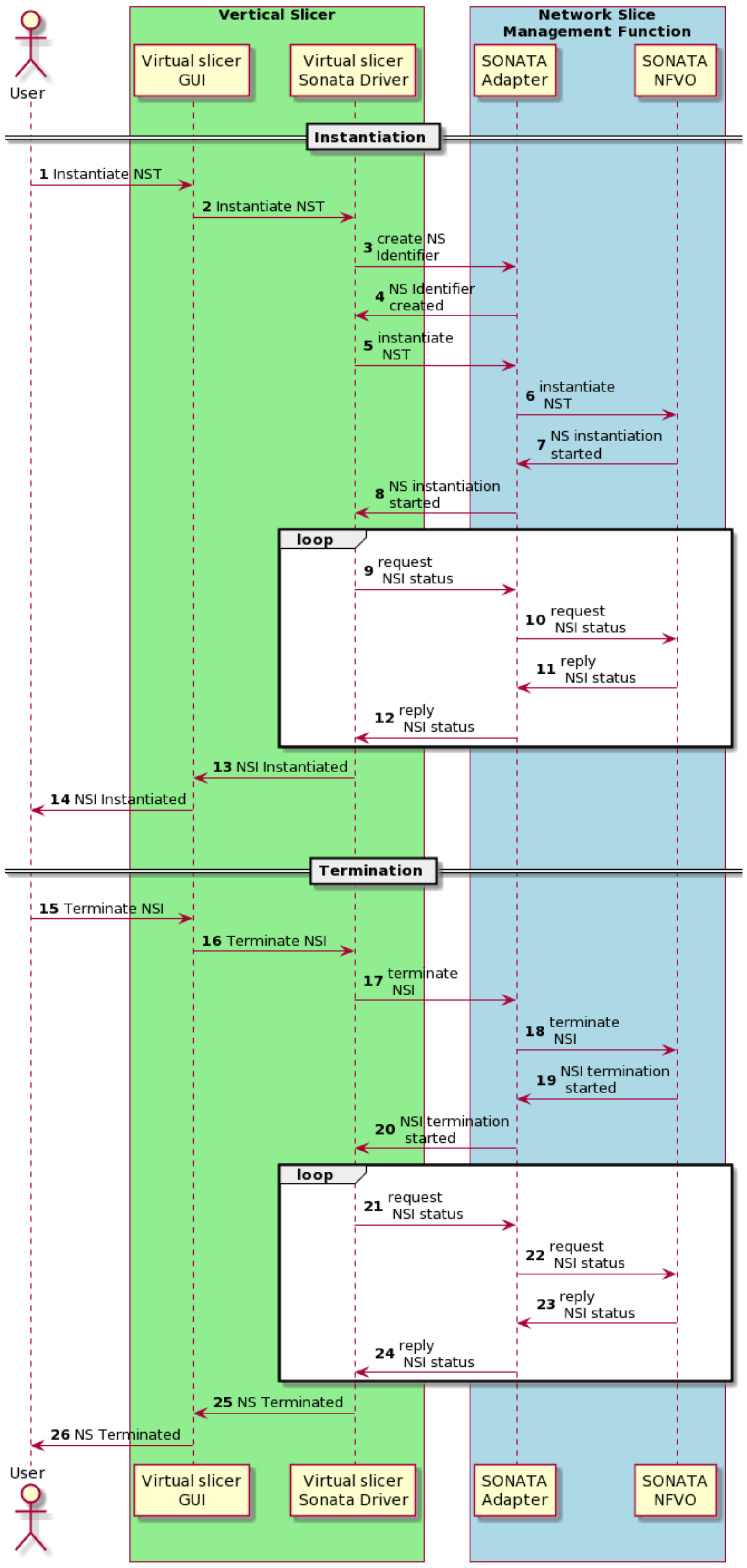

FIGURE 21: HIGH-LEVEL SEQUENCE DIAGRAM

The sequence for instantiation and termination process can be detailed as follows:

1. The Vertical User accesses the Vertical Slicer GUI and gives the instruction to instantiate a specific Network Slice Template (NST);
2. The 5Gr-VS knows that this *NstId* is for SONATA Platform and forwards the instruction to the VS SONATA Driver;
3. The VS SONATA Driver starts the process of instantiation by sending the message to create a Network Slice Instantiation Identifier (*NsiId*) to the SONATA Adapter;
4. The SONATA Adapter replies with the *NsiId* to the VS SONATA Driver;
5. The VS SONATA Driver sends the message to instantiate the Network Slice providing the *NsiId* and *NstId*;
6. The SONATA Adaptor translates the message and sends that to SONATA NFVO to instantiate the slice;
7. The SONATA NFVO replies that the instantiation has started;
8. The SONATA Adapter forwards that the instantiation has started to the VS SONATA Driver;
9. The VS SONATA Driver starts a loop to get the status of NSI until it is completed. For that, the VS SONATA Driver, periodically polls the status of *NsiId* to SONATA Adapter;
10. The SONATA Adapter forwards the request to the SONATA NFVO;
11. The SONATA NFVO replies with the status of *NsiId* and some installation information;
12. The SONATA Adapter translates the message and replies to the VS SONATA Driver;
13. The VS SONATA Driver, when receives the status for the finalized instantiation, replies to 5Gr-VS with the information needed;
14. The 5Gr-VS GUI sends the information to Vertical User that the instantiation was finalized.
15. The Vertical User accesses the Vertical Slicer GUI and instructs the termination of a specific *NsiId*;
16. The 5Gr-VS knows that this *NsiId* is for SONATA Platform and forwards the instruction to the VS SONATA Driver;
17. The VS SONATA Driver starts the process of termination by sending the message to terminate a *NsiId* to the SONATA Adapter;
18. The SONATA Adaptor translates the message and sends it to the SONATA NFVO to terminate the slice;
19. The SONATA NFVO replies that the termination has started;
20. The SONATA Adapter forwards that the termination has started to the VS SONATA Driver;
21. The VS SONATA Driver starts a loop to get the status of NSI until it is completed. For that, the VS SONATA Driver periodically sends the message to request the status of *NsiId* to SONATA Adapter;
22. The SONATA Adapter forwards the request to SONATA NFVO;
23. The SONATA NFVO replies with the status of *NsiId* and some slice information;
24. The SONATA Adapter translates the message and replies to VS SONATA Driver;
25. The VS SONATA Driver, when receives the status for the finalized termination, replies to VS with the information needed;
26. The 5Gr-VS GUI sends the information to Vertical User that the termination was finalized.

2.4.3. Implementation

In this version, the SONATA packages (descriptors) are previously onboarded in the SONATA platform, and the VS Blueprint (VSB) has the NST Id of the SONATA platform. The translator changes the messages between format TS 28.531, described in Table 9, and SONATA API format, described inside Table 10.

The API of the NBI has the following HTTP Methods, gathered in Table 8.

TABLE 8: NBI API

Action	HTTP Method	Endpoint	Body	Output Status	Output Body
Create a Network Slice Identifier	POST	http://<sonata-ip>:<sonata-adapter-port>/api/v1/ns	<CreateNsiIdRequest>	201	"nsiId"
Instantiate a Network Slice	PUT	http://<sonata-ip>:<sonata-adapter-port>/api/v1/ns/<nsiId>/action/instantiate	<InstantiateNsiRequest>	202	Null
Request the information of a Network Slice Instantiation	GET	http://<sonata-ip>:<sonata-adapter-port>/api/v1/ns/<nsiId>		200	NetworkSliceInstance
Request the information of all Network Slice Instantiations	GET	http://<sonata-ip>:<sonata-adapter-port>/api/v1/ns		200	[NetworkSliceInstance]
Terminate a Network Slice Instantiation	PUT	http://<sonata-ip>:<sonata-adapter-port>/api/v1/ns/<nsiId>/action/terminate	<TerminateNsiRequest>	202	Null

The body content that goes in the messages from Table 8 is described in Table 9.

TABLE 9: MESSAGE BODY DEFINITION

CreateNsiIdRequest
{
  "description": "string",
  "name": "string",
  "nstId": "string"
}
 	
TerminateNsiRequest
{
  "nsiId": "string"
}
	InstantiateNsiRequest
	{
	  "dfId": "string",
	  "ilId": "string",
	  "locationConstraints": {
	    "altitude": 0,
	    "latitude": 0,
	    "longitude": 0,
	    "range": 0
	  },
	  "nsSubnetIds": [
	    "string"
	  ],
	  "nsiId": "string",
	  "nstId": "string",
	  "ranEndPointId": "string",
	  "userData": {
	    "additionalProp1": "string",
	    "additionalProp2": "string",
	    "additionalProp3": "string"
	  }
	}
		NetworkSliceInstance
		{
		  "name": "string",
		  "description": "string",
		  "nsiId": "string",
		  "nstId": "string",
		  "nsdId": "string",
		  "nsdVersion": "string",
		  "dfId": "string",
		  "instantiationLevelId": "string",
		  "nfvNsId": "string",
		  "soManaged": "bool",
		  "networkSliceSubnetInstances":
		"list<string>",
		  "tenantId": "string",
		  "status": "string",
		  "errorMessage": "string",
		  "nfvNsUrl": "string"
		}

The SONATA API has the below HTTP Methods supported by SBI.

TABLE 10: SONATA API SUPPORTED BY SBI

Action	HTTP Method	Endpoint	Body	Output Status

Instantiate a Network Slice	POST	http://<sonata-ip>:32002/api/v3/requests	{
			  "name":"<nsi-name>",
			  "nst_id":"<uuid_of_existing_NST>",
			  "request_type":"CREATE_SLICE"
			}
				201

Instantiate a Network Slice specifying the subnet per VIM	POST	http://<sonata-ip>:32002/api/v3/requests	{
			  "name":"<nsi-name>",
			  "nst_id":"<uuid_of_existing_NST>",
			  "request_type":"CREATE_SLICE",
			  "instantiation_params":"[
			  {
			    \"vim_id\":\"<uuid_of_existing_VIM>\",
			    \"subnet_id\":\"<id_of_subnet1_from_NST>\"
				201

			},
			{
			\"vim_id\":\"<uuid_of_existing_VIM>\",
			\"subnet_id\":\"<id_of_subnet2_from_NST>\"
			}]"
			}	

Request the information of all Network Slice instantiations	GET	http://<sonata-ip>:32002/api/v3/requests		200
Request the information of a Network Slice Instantiation	GET	http://<sonata-ip>:32002/api/v3/requests/<nsi_uuid>		200

Terminate a Network Slice Instantiation	POST	http://<sonata-ip>:32002/api/v3/requests	{
			"instance_uuid":"<nsi_uuid>",
			"request_type":"TERMINATE_SLICE"
			}
				201

2.5. Code developed for COMAU integration

2.5.1. General description

The code developed for COMAU uses the Radio Access Network (RAN) slice feature developed in 5Growth stack (i.e. 5Gr-VS, 5Gr-SO and 5Gr-RL modules) and described in D2.2 [8]. Figure 22 (taken from D2.2) reports the architecture of this feature highlighting in red the blocks that are involved.

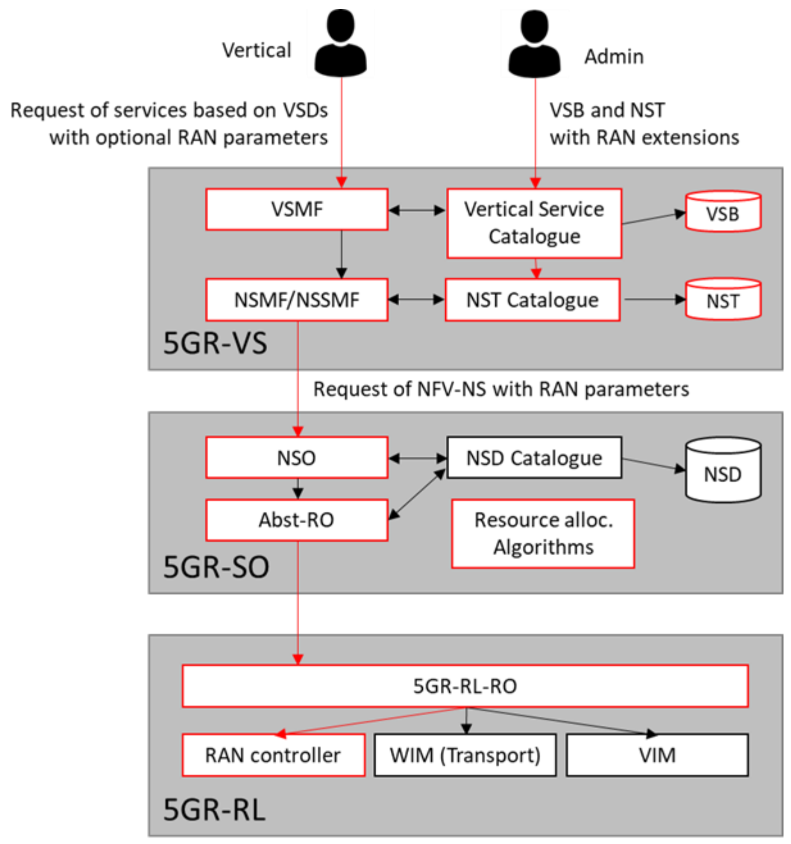
Vertical
Admin
Request of services based on VSDs with optional RAN parameters
VSB and NST with RAN extensions
VSMF
Vertical Service Catalogue
VSB
NSMF/NSSMF
NST Catalogue
NST
5GR-VS
Request of NFV-NS with RAN parameters
NSO
NSD Catalogue
NSD
Abst-RO
Resource alloc. Algorithms
5GR-SO
5GR-RL-RO
RAN controller
WIM (Transport)
VIM
5GR-RL

FIGURE 22: RAN SLICE SUPPORT IN 5GROWTH STACK

Two operations are done for COMAU integrations:

- Development of 5Gr-RL plugin that interacts with Ericsson RAN Controller used in the Pilot
- Design of Service Descriptors (i.e. NSD, VSD, VSB) for COMAU use cases

The 5Gr-RL plugin is used to map the 5Gr-RL command in Ericsson RAN controller specific command to configure the Radio part in COMAU pilot. The NSD/VSD/VSB descriptions are defined according to the 3GPP specifications TS28.530 for RAN slice support [9] whose architecture is shown in Figure 23.

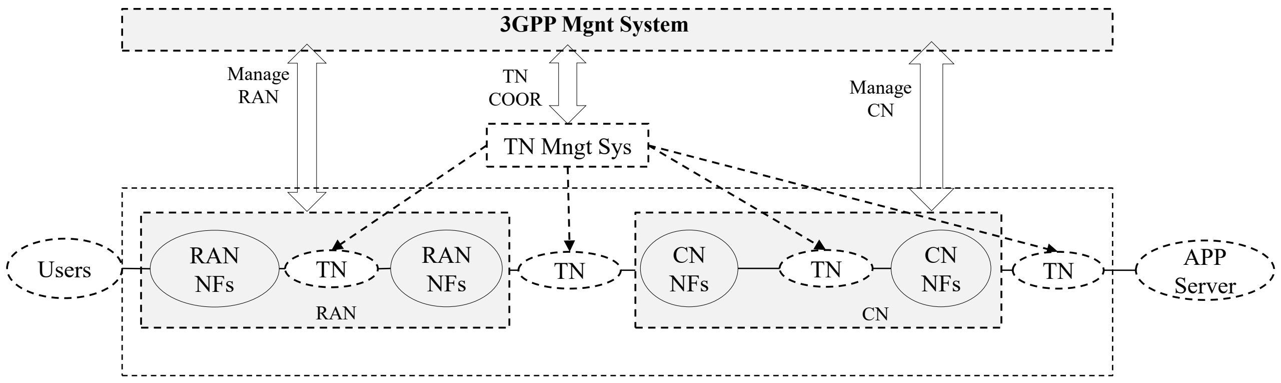

FIGURE 23: RAN SLICE NETWORK SERVICE ARCHITECTURE

2.5.2. Services and workflows

The supported operations are:

- lifecycle operation of the VNF instance (i.e. allocation, modification, termination API) of a Vertical Service instance
- lifecycle operation of VNF virtual network (i.e. allocation, modification termination API) that handles the VNF connectivity inside the radio domain (i.e. the connection of VNF up to the Radio domain gateway) of a Vertical Service instance.

Figure 24 and Figure 25 show respectively the allocation and termination workflow for the VNF and virtual networks of Vertical Service. The handled API is the same used to configure VNFs in a datacenter VIM and follows the ETSI IFA005 [10] specification.

The sequence for the instantiation process can be detailed as follows:

1. Vertical requires the instantiation of a Vertical service instance to 5Gr-VS
2. 5Gr-VS stores it and map it to the Network Service Descriptor that gives the set of VNF and VL to be allocated
3. 5Gr-VS requires the instantiation of an instance of NSD to 5Gr-SO
4. 5Gr-SO places the VNF and VL on NFVI-POP and Logical Link (LL) according to the abstract view provided by 5Gr-RL
5. 5Gr-SO requests the allocation of the resources on NFVI-POP for VNF
6. 5Gr-SO requests the allocation of the resources on LL for VL
7. 5Gr-RL allocates the required resources and provides a reference to them
8. 5Gr-SO stores the references
9. 5Gr-SO sends an instantiation reply to 5Gr-VS

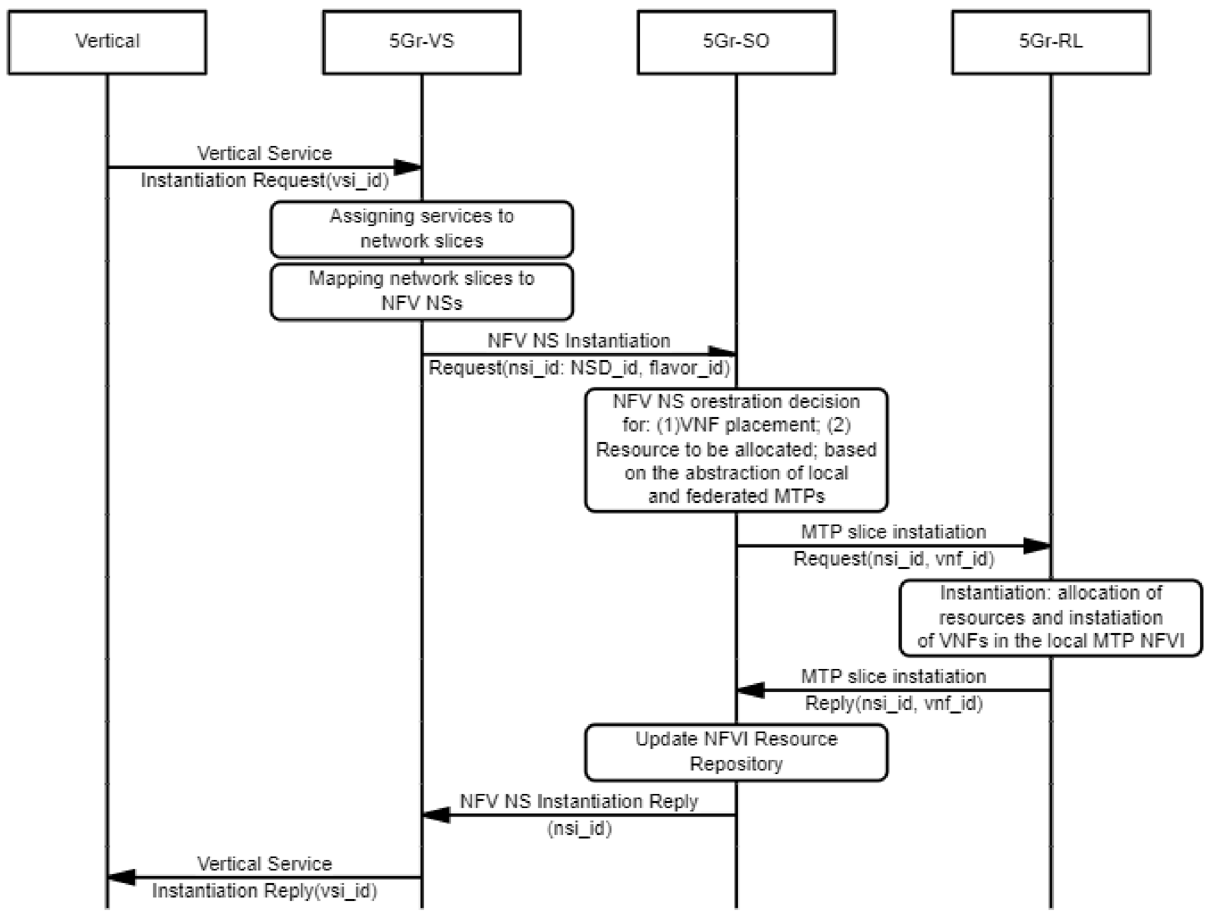
Vertical
5Gr-VS
5Gr-SO
5Gr-RL
Vertical Service Instantiation Request(vsi_id)
Assigning services to network slices
Mapping network slices to NFV NSs
NFV NS Instantiation Request(nsi_id: NSD_id, flavor_id)
NFV NS orestration decision for: (1)VNF placement; (2) Resource to be allocated; based on the abstraction of local and federated MTPs
MTP slice instatiation Request(nsi_id, vnf_id)
Instantiation: allocation of resources and instatiation of VNFs in the local MTP NFVI
MTP slice instatiation Reply(nsi_id, vnf_id)
Update NFVI Resource Repository
NFV NS Instantiation Reply (nsi_id)
Vertical Service Instantiation Reply(vsi_id)

FIGURE 24: VERTICAL SERVICE ALLOCATION WORKFLOW IN 5GROWTH STACK

The sequence for the termination process can be detailed as follows:

1. Vertical requires the termination of an instantiated Vertical service to 5Gr-VS
2. 5Gr-VS sets the instance to “terminating”
3. 5Gr-VS requires the termination of the instantiated NS to 5Gr-SO
4. 5Gr-SO retrieves the resource references for the instantiated NS
5. 5Gr-SO requests the termination of the resources on NFVI-POP for VNF
6. 5Gr-SO requests the termination of the resources on LL for VL
7. 5Gr-RL terminates the required resources
8. 5Gr-RL notifies 5Gr-SO of the resource termination
9. 5Gr-SO removes the references
10. 5Gr-SO sends a termination reply to 5Gr-VS

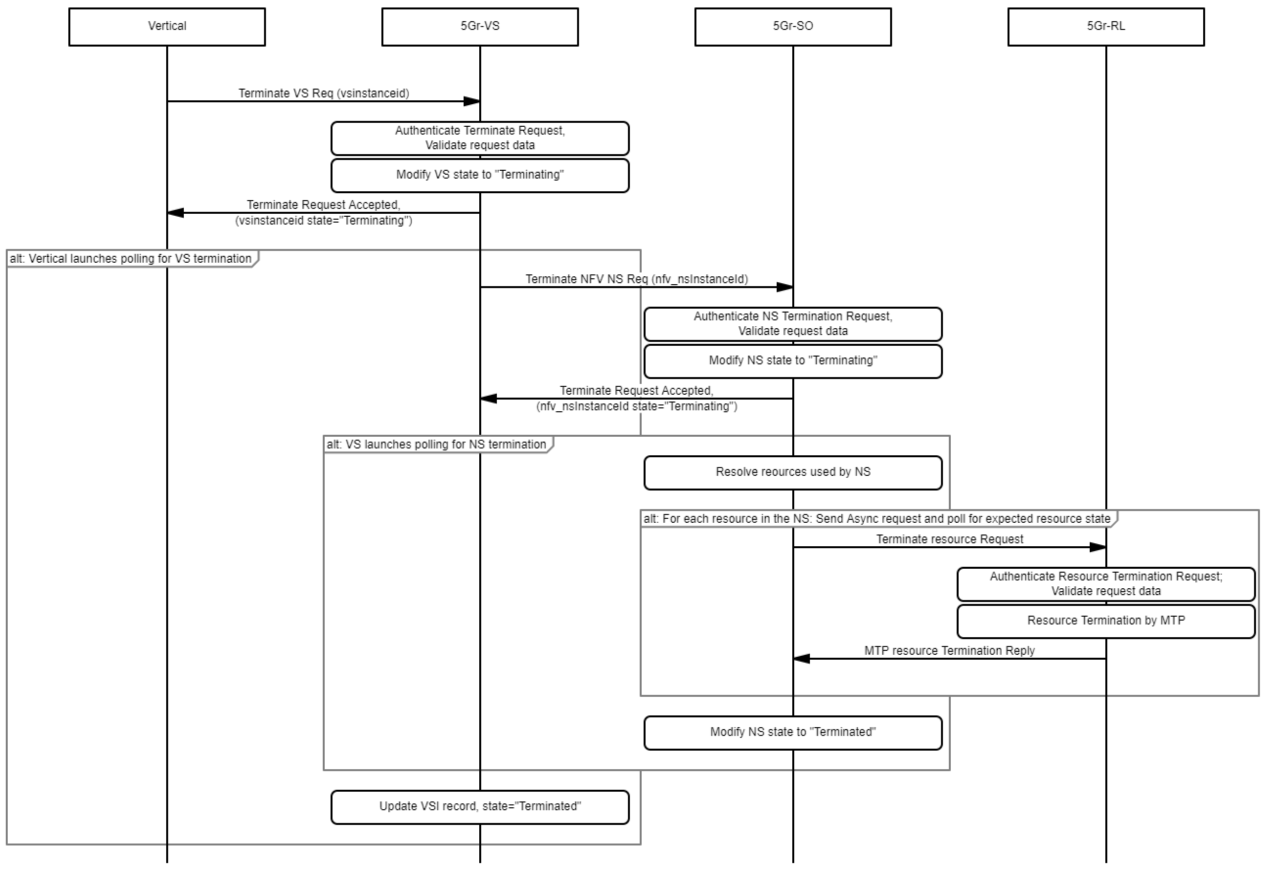
Vertical
5Gr-VS
5Gr-SO
5Gr-RL
Terminate VS Req (vsinstanceid)
Authenticate Terminate Request, Validate request data
Modify VS state to "Terminating"
Terminate Request Accepted, (vsinstanceid state="Terminating")
alt: Vertical launches polling for VS termination
Terminate NFV NS Req (nfv_nsInstanceId)
Authenticate NS Termination Request, Validate request data
Modify NS state to "Terminating"
Terminate Request Accepted, (nfv_nsInstanceId state="Terminating")
alt: VS launches polling for NS termination
Resolve reources used by NS
alt: For each resource in the NS: Send Async request and poll for expected resource state
Terminate resource Request
Authenticate Resource Termination Request; Validate request data
Resource Termination by MTP
MTP resource Termination Reply
Modify NS state to "Terminated"
Update VSI record, state="Terminated"

FIGURE 25: VERTICAL SERVICE TERMINATION WORKFLOW IN 5GROWTH STACK

2.5.3. Implementation

As the 5Gr-RL plugin development follows the implementation time-plan of the RAN feature defined in D2.2, the implementation is an ongoing process. The current plugin can handle functions developed in the first release of the 5Growth platform like Radio VNF. Table 11 reports the API that is managed by the Radio Plugin. As additional functions will be released later in the platform (like the PNF and 3GPP MF support), the API for handling such functions will be described in a future project deliverable.

The Network Service Descriptor (NSD) of the radio connection between the device (such as an arm, robot, smartphone, tablet…) and the vertical application server is divided into RAN functions and Core functions that are connected via a generic transport network. The NSDs for COMAU use cases define two Network Functions (NFs) for RAN part and two for Core part. The RAN NFs are the Remote Unit (RU) and Baseband Unit (BBU), while the core NFs are the virtual Evolved Packet Core (vEPC) and the Home Subscriber Server (HSS), that is separated from vEPC. The VSB and VSD define the vertical requirements (like maximum number of devices, scaling rules) with the addition of the parameters to configure the 5Gr-VS slicer. Additional details of the NSD, VSD, and VSB can be found in the project git repository[8].

TABLE 11: REST API USED BY RADIO PLUGIN

Action	HTTP Method	Endpoint	Output Status
Request Radio Coverage Area	GET	curl -X GET http://<plugin-ip>:<plugin-port>/abstract-radio-coveragearea	200
Request NFVI POP Information	GET	curl -X GET http://<plugin-ip>:<plugin-port>/network_resources/nfvi-pop-network-information	
Request NFVI POP Compute Capabilities	GET	curl -X GET http://<plugin-ip>:<plugin-port>/compute_resources/information	
Instantiate a VNF	POST	curl -X POST http://<plugin-ip>:<plugin-port>/compute_resources -d ' { "accelerationCapability": ["string"], "computeId": "string", "computeName": "string", "flavourId": "string","hostId": "string","operationalState": "string","vcImageId": "string","virtualCpu": {"cpuArchitecture": "string","cpuClock": 0,"numVirtualCpu": 0, ”virtualCpuOversubscriptionPolicy": "string", ”virtualCpuPinning": {"cpuMap": "string","cpuPinningPolicy": "string","cpuPinningRules": "string" } },"virtualDisks": "string", ”virtualMemory": {"numaEnabled": true,"virtualMemOversubscriptionPolicy": "string","virtualMemSize": 0},"virtualNetworkInterface": [{ "accelerationCapability": "string","bandwidth": "string","ipAddress": ["string"],"macAddress": "string","metadata": [{"key": "string","value": "string"}], "networkName": "string","networkId": "string","networkPortId": "string","operationalState": "string","ownerId": "string","resourceId": "string","typeConfiguration":	201

[8] https://5growth.eu/redmine/projects/5growth/repository/pilots/COMAU

		"string","typeVirtualNic": "string"}],"zoneId": "string"}' -vvv -H 'content-type:application/json'	
Terminate a VNF	DELETE	curl -X POST http://<plugin-ip>:<plugin-port>/compute_resources/{computeId}	201
Instantiate a VN	POST	curl -X POST http://<plugin-ip>:<plugin-port>//network_resources -d '{"affinityOrAntiAffinityConstraints": "string","locationConstraints": "string","metadata": [{"key": "string","value": "string"}],"networkResourceName": "string","networkResourceType": "string","reservationId": "string","resourceGroupId": "string","typeNetworkData": "string","typeNetworkPortData": "string","typeSubnetData": {"resourceId": "string","networkId": "string","ipVersion": "string","gatewayIp": "string",”cidr": "string","isDhcpEnabled": true,"addressPool": [0],"metadata": [{"key": "string","value": "string"}]}}' -vvv -H 'content-type:application/json'	201
Terminate a VN	DELETE	curl -X POST http://<plugin-ip>:<plugin-port>/compute_resources/{networkId}	201

3. Considerations for Future Work

This first code release of the platform software provides the initial implementation of the software components required for the integration of 5Growth and the ICT-17 platforms 5G-EVE and 5G-VINNI / SONATA. It aims to provide the baseline implementation that will allow the vertical pilots to start testing the deployment of the use cases across different platforms. As such, the implementation focus was on the support of the most common service lifecycle management operations, like creation, instantiation, status, termination, and scale. The implemented software components have been initially validated in integration tests against mock-ups / dummy components of each ICT-17 platform.

For future perspectives, two lines of work are going to be followed in a parallel approach:

1. **Full workflow integration tests with the real ICT-17 platforms:** upon the successful connection between 5Growth and ICT-17 platform sites, the required environment to test the full interaction workflows will become available. However, as individual integrations were already tested and validated, we envision these future tests to go seamless.
2. **Development and improvements on both interactions with the ICT-17 platforms** are going to be considered according to: *(i)* integration within the pilots; *(ii)* support of required WP2 5Growth innovations; and *(iii)* integrated monitoring support. However, the feasibility of novel developments and improvements on the interactions will require an initial assessment since, it not only depends on the needs of the 5Growth vertical pilots, but also on existing support by the ICT-17 platforms.

The aforementioned lines of work will allow us to identify potential issues that will need further debugging as well as improvements in the code to face a wide range of conditions. Overall, all the proposed tasks are going to contribute to a more complete, mature and robust code (to be released with future D3.5) which can be leveraged to support many other vertical industries and their use cases.

References

[1] 5Growth, “D3.2, "Specification of ICT-17 in-house deployment",” [Online]. Available: https://5growth.eu/wp-content/uploads/2020/04/D3.2-Specification_of_ICT17_in-house_deployment.pdf.

[2] 3GPP, “3GPP TS 28.531, "5G; Management and orchestration; Provisioning",” [Online]. Available: https://www.etsi.org/deliver/etsi_ts/128500_128599/128531/15.00.00_60/ts_128531v150000p.pdf.

[3] ETSI, “ETSI GS NFV-SOL 005, "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point",” [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/02.07.01_60/gs_NFV-SOL005v020701p.pdf.

[4] ETSI, “ETSI GS NFV-IFA 013, "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Os-Ma-nfvo reference point - Interface and Information Model Specification",” [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/013/03.04.01_60/gs_NFV-IFA013v030401p.pdf.

[5] ETSI, “ETSI GS NFV-SOL 001, "Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on TOSCA specification",” [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/001/03.03.01_60/gs_NFV-SOL001v030301p.pdf.

[6] ETSI, “ETSI GS NFV-IFA 014, "Network Functions Virtualisation (NFV); Management and Orchestration; Network Service Templates Specification",” [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/014/02.01.01_60/gs_NFV-IFA014v020101p.pdf.

[7] ETSI, “ETSI GS NFV-IFA 011, "Network Functions Virtualisation (NFV); Management and Orchestration; VNF Packaging Specification",” [Online]. Available: https://www.etsi.org/deliver/etsi_gs/nfv-ifa/001_099/011/02.01.01_60/gs_nfv-ifa011v020101p.pdf.

[8] 5Growth, “D2.2, "Initial implementation of 5G End-to-End Service Platform",” [Online]. Available: https://5growth.eu/wp-content/uploads/2020/05/D2.2-Initial_implementation_of_5G_End-to-End_Service_Platform.pdf.

[9] 3GPP, “3GPP TS 28.530, "5G; Management and orchestration; Concepts, use cases and requirements",” [Online]. Available: https://www.etsi.org/deliver/etsi_ts/128500_128599/128530/15.00.00_60/ts_128530v150000p.pdf.

[10] ETSI, “ETSI GS NFV-IFA 005, "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification",” [Online].

Available: https://www.etsi.org/deliver/etsi_gs/nfv-ifa/001_099/005/03.01.01_60/gs_nfv-ifa005v030101p.pdf.

[11] 5Growth, " Pilots repository," [Online]. Available: https://5growth.eu/redmine/projects/5growth/repository/pilots .

Annex 1: Descriptors mapping

SONATA descriptors - examples

NSD1.yaml

```
descriptor_schema: "https://raw.githubusercontent.com/sonata-nfv/tng-
schema/master/service-descriptor/nsd-schema.yml"

vendor: "eu.5gvinni"
name: "ns-open5gcorer5-access"
version: "0.1"
author: "5GVINNI ALB Team"
description: "This is a service to deploy Open5gCore Release 5"

network_functions:
  - vnf_id: "vnf_open5gcorer5_upfe"
    vnf_vendor: "eu.5gvinni"
    vnf_name: "vnf-open5gcorer5-upfe"
    vnf_version: "0.1"
  - vnf_id: "vnf_open5gcorer5_ue"
    vnf_vendor: "eu.5gvinni"
    vnf_name: "vnf-open5gcorer5-ue"
    vnf_version: "0.1"
  - vnf_id: "vnf_open5gcorer5_gnb"
    vnf_vendor: "eu.5gvinni"
    vnf_name: "vnf-open5gcorer5-gnb"
    vnf_version: "0.1"
  - vnf_id: "vnf_open5gcorer5_mgmt"
    vnf_vendor: "eu.5gvinni"
    vnf_name: "vnf-open5gcorer5-mgmt"
    vnf_version: "0.1"

connection_points:
  - id: "mgmt"
    interface: "ipv4"
    type: "management"
  - id: "out1"
    interface: "ipv4"
    type: "external"
  - id: "out2"
    interface: "ipv4"
    type: "external"
  - id: "out3"
    interface: "ipv4"
    type: "external"

virtual_links:
  - id: "mgmt"
```

```
    connectivity_type: "E-LAN"
    connection_points_reference:
      - "vnf_open5gcorer5_ue:mgmt"
      - "vnf_open5gcorer5_gnb:mgmt"
      - "vnf_open5gcorer5_mgmt:mgmt"
      - "vnf_open5gcorer5_upfe:mgmt"
      - "mgmt"
  - id: "air"
    connectivity_type: "E-Line"
    connection_points_reference:
      - "vnf_open5gcorer5_gnb:inout1"
      - "vnf_open5gcorer5_ue:inout1"
  - id: "ngu1"
    connectivity_type: "E-Line"
    connection_points_reference:
      - "vnf_open5gcorer5_upfe:out1"
      - "out1"
  - id: "cpn4"
    connectivity_type: "E-LAN"
    connection_points_reference:
      - "vnf_open5gcorer5_gnb:out2"
      - "vnf_open5gcorer5_upfe:out2"
      - "out2"
  - id: "ngu"
    connectivity_type: "E-Line"
    connection_points_reference:
      - "vnf_open5gcorer5_gnb:inout2"
      - "vnf_open5gcorer5_upfe:inout1"
  - id: "upt12"
    connectivity_type: "E-Line"
    connection_points_reference:
      - "out3"
      - "vnf_open5gcorer5_upfe:out3"

service_specific_managers:
 - id: "tngssm5gservicetask-config-monitor"
   description: "An SSM functioning as task, config and monitor SSM."
   image: "mesquitasonata/5gservicer5-ssm-access-taskconfigmonitor"
   options:
     - key: "type"
       value: "task"
     - key: "type"
       value: "configure"
     - key: "type"
       value: "monitor"
```

Gnb1.yml

```
descriptor_schema: "https://raw.githubusercontent.com/sonata-nfv/tng-
schema/master/function-descriptor/vnfd-schema.yml"

vendor: "eu.5gvinni"
```

```
name: "vnf-open5gcorer5-gnb"
version: "0.1"
author: "5GVINNI ALB Team"
description: "This is a service to deploy Open5gCore Release 5, namelly its GNB
component"

virtual_deployment_units:
  - id: "gnb1"
    vm_image: "http://www.google.com"
    vm_image_format: "qcow2"
    vm_image_md5: a43594a34839b88c9b6630caf947ebfe
    resource_requirements:
      cpu:
        vcpus: 2
      memory:
        size: 2
        size_unit: "GB"
      storage:
        size: 10
        size_unit: "GB"
    connection_points:
      - id: "eth0"
        interface: "ipv4"
        type: "internal"
      - id: "eth1"
        interface: "ipv4"
        type: "internal"
      - id: "eth2"
        interface: "ipv4"
        type: "external"
        security_groups: []
      - id: "eth3"
        interface: "ipv4"
        type: "external"
      - id: "eth4"
        interface: "ipv4"
        type: "external"
    user_data: |
      password: ubuntu
      chpasswd: { expire: False }
      ssh_pwauth: True
      network: {config: disabled}

virtual_links:
  - id: "mgmt"
    connectivity_type: "E-LAN"
    connection_points_reference:
      - "gnb1:eth0"
      - "mgmt"
    dhcp: True
  - id: "ngc-2-out"
    connectivity_type: "E-LAN"
    connection_points_reference:
      - "gnb1:eth1"
      - "out1"
    dhcp: True
  - id: "air-2-air"
    connectivity_type: "E-LAN"
    connection_points_reference:
      - "inout1"
      - "gnb1:eth2"
    dhcp: True
  - id: "cpn4-2-out2"
    connectivity_type: "E-LAN"
    connection_points_reference:
      - "gnb1:eth3"
      - "out2"
    dhcp: True
  - id: "ngu-2-out3"
    connectivity_type: "E-LAN"
    connection_points_reference:
      - "gnb1:eth4"
      - "inout2"
    dhcp: True

connection_points:
  - id: "mgmt"
    interface: "ipv4"
    type: "management"
  - id: "out1"
    interface: "ipv4"
    type: "external"
  - id: "inout1"
    interface: "ipv4"
    type: "external"
  - id: "out2"
    interface: "ipv4"
    type: "external"
  - id: "inout2"
    interface: "ipv4"
    type: "external"

function_specific_managers:
  - id: "sonfsm5gservicegnbgnb-config"
    description: "FSM for the configuration of the GNB vnf"
    image: "mesquitasonata/5gservicer5-gnb-fsm-css"
    options:
      - key: "type"
        value: "start"
      - key: "type"
        value: "stop"
      - key: "type"
        value: "configure"
```