Research & Innovation Actions

5G PPP Research and Validation of critical technologies and systems: Enabling Smart Energy as a Service via 5G Mobile Network advances.

Project: H2020-ICT-07-2017

Enabling Smart Energy as a Service via 5G Mobile Network advances

Deliverable 3.1

Semi-automatic NS/VNF deployment

Author(s): Filippo Rebecchi, Bruno Vidalenc, Dallal Belabed (TCS), Konstantinos Kalaboukas (SiLO), Antonello Corsi, Giampaolo Fiorentino (ENG), Mihai Mladin (CRE), Alistair Duke (BT), Artemis Voulkidis (POPs).

Status-Version: V1.0

Delivery Date (DOW): 31 January 2018

Actual Delivery Date: 2 February 2018

Distribution - Confidentiality: Public

Abstract:

This deliverable provides the initial design of the automatic NS/VNF deployment as result of the activities carried out in Task 3.1 of the NRG-5 project. The deliverable builds on a thorough analysis of the available open-source software for NFV deployments associated with the specific requirements from NRG-5 use cases. It provides the initial extensions of the information model along with the interfaces to allow for proper MANO integration from a smart-grid management perspective.

Disclaimer

The NRG-5 Consortium is the following:

Participant number	Participant organisation name	Short name	Country
01	Engineering-Ingegneria Informatica SPA	ENG	Italy
02	THALES Communications & Security	TCS	France
03	SingularLogic S.A.	SiLO	Greece
04	Ineo Energy & Systems	ENGIE	France
05	Romgaz S.A	RGAZ	Romania
06	ASM Terni SpA	ASM	Italy
07	British Telecom	BT	UK
08	Wind Telecomunicazioni S.P.A.	WIND	Italy
09	Hispasat S.A.	HIS	Spain
10	Power Operations Limited	POPs	UK
11	Visiona Ingenieria De Proyectos Sl	VIS	Spain
12	Optimum S.A	OPT	Greece
13	Emotion s.r.l	EMOT	Italy
14	Rheinisch-Westfälische Technische Hochschule Aachen	RWTH	Germany
15	Jožef Stefan Institute	JSI	Slovenia
16	TEI of Sterea Ellada/Electrical Engineering Dept.	TEISTE	Greece
17	University Pierre et Marie Curie	UPMC	France
18	Centro Romania Energy	CRE	Romania
19	Rutgers State University of New Jersey	Rutgers	USA

Document Revision History

Date	Issue	Author/Editor/Contributor	Summary of main changes
17/10/17	0.1	F. Rebecchi (TCS)	Initial ToC
18/12/17	0.2	F. Rebecchi (TCS), K. Kalaboukas (SiLO)	Updated ToC, contributions from on the information model
21/12/17	0.3	A. Corsi (ENG)	Added Interfaces and API initial contribution
22/12/17	0.4	F. Rebecchi (TCS), K. Kalaboukas (SiLO)	Extracted requirements from use cases, updated data models
05/01/18	0.5	F. Rebecchi, B. Vidalenc (TCS)	Updated executive summary, added Sonata description, improved ETSI NFV description, added introduction to information model, added conclusion
12/01/18	0.6	M. Mladin (CRE)	Contributions on use cases and requirements.
22/01/18	0.7	A. Duke (BT), A. Voulkidis (POPs), F. Rebecchi, B. Vidalenc, D. Belabed (TCS), K. Kalaboukas (SiLO), A. Corsi (ENG)	1st round of peer reviews, comments and suggestions
29/01/18	0.8	F. Rebecchi, B. Vidalenc, D. Belabed (TCS), K. Kalaboukas (SiLO), A. Corsi (ENG)	Integration of comments from reviewers, overhauled executive summary, added comparison table, acronyms and fixed references.
01/02/18	1.0	A.Corsi, G. Fiorentino (ENG), F. Rebecchi (TCS)	Final review and contributions

Table of Contents

Executive Summary ___ ***6***

1 Introduction ___ ***7***

1.1 Purpose of the Document ___ 7
1.2 Scope of the Document ___ 7
1.3 Structure of the Document ___ 8

2 State of the Art ___ ***9***

2.1 ETSI NFV Architecture Primer ___ 9
2.2 MANO Open Source Projects ___ 12

3 Requirements Extracted from the Use Cases ___ ***21***

3.1 NRG-5 Use Cases Overview ___ 21
3.2 Requirements for the NRG-5 Platform ___ 23

4 Information Model ___ ***26***

4.1 Introduction ___ 26
4.2 Virtual Network Function and Service Descriptors ___ 26
4.3 Modelling Languages ___ 30
4.4 Information Model from Open Source projects ___ 30
4.5 Comparison of the NFV Platforms ___ 46
4.6 NRG5 Extensions to the Information Model ___ 48

5 Interfaces and APIs ___ ***50***

5.1 The Role of Catalogues ___ 50
5.2 Interactions between the Catalogue and the NRG-5 Platform ___ 50
5.3 Interfaces to the Developers ___ 51
5.4 Interfaces to the MANO ___ 52
5.5 Orchestrator to the OSS layer ___ 52

6 Conclusion ___ ***53***

7 Acronyms ___ ***54***

8 Reference ___ ***56***

Table of Figures

Figure 1: ETSI NFV Reference Architecture [3]. 9
Figure 2: NFV entities and their relationship. 11
Figure 3: The OSM Components are aligned with the ETSI Architecture. 13
Figure 4: ONAP Platform Components (Amsterdam Release). 15
Figure 5: High-level Tacker Architecture. 16
Figure 6: SONATA Service Platform Architecture. 18
Figure 7: Upstream projects used in OPNFV (Euphrates Release). 19
Figure 8: Dynamics of process control at highest rate, with latency. 24
Figure 9: Requirements and scope of the NRG-5 data/information model. 26
Figure 10: Graphical representation of the NS to deploy. 27
Figure 11: Example NSD composed of two VNFs chained together. 28
Figure 12: Example VNFD composed of a single VDU. 29
Figure 13: VNFD high-level object model. 33
Figure 14: General Mapping between TOSCA and NFV. 40

Executive Summary

The objective of this deliverable is to present the initial version of the NRG-5 semi-automatic Virtual Network Functions (VNFs) / Network Service (NSs) deployment, as a result of the activities carried out in Task 3.1. As key contributions, it defines the guidelines for extending the programming interfaces and the information model to allow proper Network Function Virtualization Management & Network Orchestration (NFV MANO) integration from a smart-grid management perspective.

The performed analysis has a project-wide resonance and motivates the decision to adapt and extend the Open Source MANO (OSM) platform in the light of the specific NRG-5 requirements. The provided guidelines will direct the development work on the NRG-5 platform performed in WP1 (architecture), WP2 (network functions), and WP3 (deployment). Further details about the actual implementation will be described in future deliverable releases. Also, the knowledge that is likely to be learned during the development and integration phases will be taken into account and retransferred in future deliverables.

More in detail, the deliverable provides a thorough overview of the existing open source alternatives of the NFV MANO platform and the information models to be used in NRG-5. Then, by analyzing the requirements of the smart energy use cases and performing a gap analysis with current functionalities, the document provides a set of recommendations on additions to be integrated in the information model and the catalogues used in NRG-5. Finally, the deliverable presents the relevant applicative programming interfaces (APIs) to model the interactions between the NRG-5 catalogues and the MANO functional blocks, and between the catalogues and the service developers.

1 Introduction

1.1 Purpose of the Document

The purpose of this deliverable is to introduce the basic concepts required for realizing the semi-automatic deployment on end-to-end telecommunication services to realize energy-related advanced functionalities envisioned in NRG-5. This deliverable represents the first technical outcome for the NRG-5 project, and the work has been developed alongside the guidelines provided by the requirements [1], and the reference architecture [2] of the project. The NRG-5 project follows closely the normative work done at the European Telecom Standards Institute (ETSI), aligning to and extending its architectural and modelling specifications [3], [4].

The deliverable provides a broad overview of the currently available platforms for NFV, with a special focus on open source solutions. The general concepts behind NFV and the information models to define the deployment of Network Services (NSs) and Virtual Network Functions (VNFs) are introduced and the specific architectures covered. The analysis of the requirements for the energy-related use cases exposes the weaknesses of the existing platforms, calling for extensions to their information models and architectures. In this deliverable, the focus is on the following Key Performance Indicators (KPIs) to guarantee the performance of the end-to-end network service: latency, reliability, availability and bandwidth.

The development of a catalogue represents one of the first building blocks of a service platform targeting a NFV deployment, being also the entry point for a developer that wants to build and onboard a NS. A catalogue holds all of the usable NSs and VNFs. During the deployment of a NS, the catalogue interacts with the MANO platform in order to locate and configure the basic elements required to setup the demanded NS. Several catalogues are required to store the deployment templates for a NS, usually composed of Network Service Descriptors (NSDs), Virtual Network Function Descriptors (VNFDs), configuration code, data, executables and specific management requirements and preferences. Additional catalogues are also required to retrieve the running NFV instances (both the NSs and their related VNFs), and to track the resources utilized for the established NFV services. The modifications in the data and information models required to cover the NRG-5 use cases are also reflected into the implementation of the catalogue.

The technical content of this document represents a baseline for the early trial program development, and the work related to the actual MANO implementation and the automated VNF sizing and deployment. As the project is still in its initial phases, the content is subject to a rapid evolution. Some details described here (e.g., the APIs or the internal components) might change or be complemented throughout the project based on the experience gained during the development of the pilots. All the changes will be reflected in the final architecture and in the related deliverables

Finally, the intended audience of this deliverable is service developers from Telecom Operators, Utilities, and Energy Service Companies (ESCOs) who have an interest into the NFV specification. Readers are not required to know all the details of ETSI MANO. However, some knowledge of MANO basic concepts would be advantageous for reading this document.

1.2 Scope of the Document

This deliverable is produced with the aim to define the automated deployment strategies to support the NRG-5 smart energy vertical use cases. The document addresses the basic requirements for the integration of the NRG-5 vision within the existing MANO platforms, especially from the standpoint of the adopted information model, and the open application programming interfaces (APIs) to provide for both internal and external access to instantiate service components. It takes as inputs the requirements expressed in the document D1.1 [1], integrating the overall NRG-5 architecture from D1.2 [2]. Moreover, this document is heavily influenced by the standardization work on NFV carried out at ETSI, and loosely by the 5G PPP phase one project SONATA [5] and the subsequent phase two project 5G-TANGO [6]. In the context of the NRG-5 project, this deliverable has a relevant impact on both

1.3 Structure of the Document

The remainder of the document is organized as follows. Section 2 proposes an overview of the currently available open source MANO platforms. This analysis allows the role of the catalogue to be situated precisely within the much more complex architecture proposed by MANO. The NRG-5 use cases are examined in Section 3, and their requirements regarding the platform are extracted. The information model adopted in NRG-5 is described in detail in Section 4. Particular attention is given here to innovations produced by NRG-5 in terms of extra requirements related to the VNFDs and NSDs schematic definitions, to support the specific requirements posed by the smart-energy use cases. The interfaces to interact with the different catalogue implementations are then exposed and abstracted in Section 5, providing an easy interaction with the platform. The conclusions of the deliverable are given in Section 6.

2 State of the Art

Cloud computing, Software Defined Networking (SDN) and Network Function Virtualization (NFV) have emerged as key enablers for the transformation and evolution of traditional telecommunications networks. These technologies add elasticity, flexibility and agility to service provisioning by separating the control and forwarding planes and migrating the network functions from hardware to software [7].

In this section, a global overview of the ETSI NFV MANO concepts and the related open source projects is provided. Also, this section will serve as an introduction to the essential terminology that will be used liberally in this deliverable. First, the ETSI NFV MANO architecture, its relevant functional modules, interfaces, and data models are briefly overviewed. Next, the most promising open-source platform solutions to implement an NFV-based network are presented and compared.

2.1 ETSI NFV Architecture Primer

Starting from 2012, the ETSI Industry Specification Group (ISG) for NFV has specified a high-level functional architectural framework for VNFs [3]. The NFV architectural Framework includes computing, networking, storage, and the respective virtualized resources, bundled in the form of Virtual Machines (VMs). The reason behind the introduction of virtualization lies in the fact that software-based network services allow flexible and efficient setup of network services.

The ETSI NFV architecture is defined by three main high level components as depicted in Figure 1:

- Network Functions Virtualization Infrastructure (NFVI)
- Virtualized Network Functions (VNF)
- Management and Orchestration (MANO)

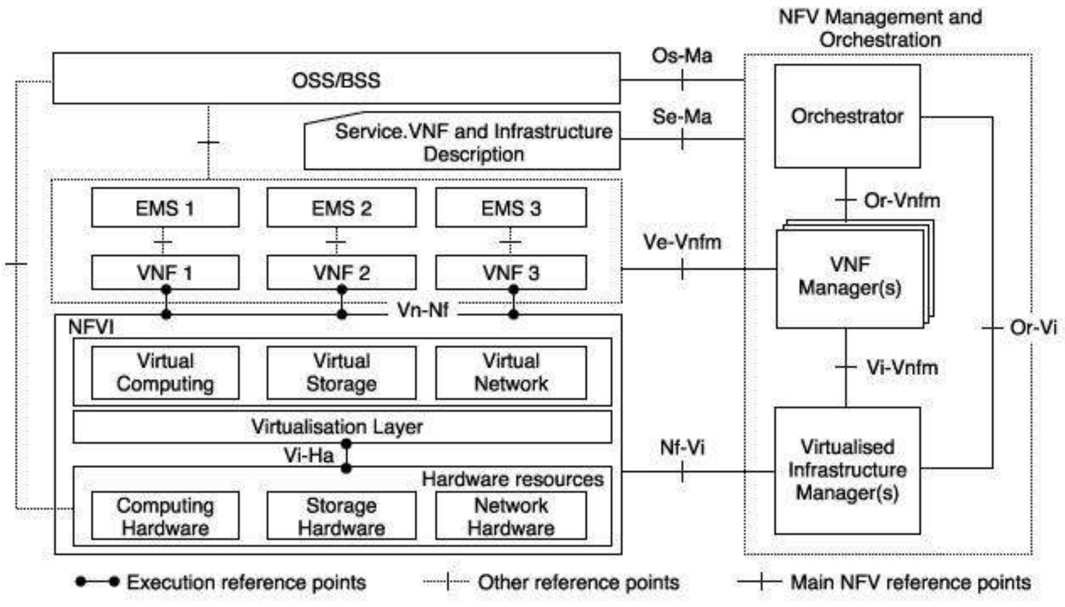


Figure 1: ETSI NFV Reference Architecture [3].

Network Functions Virtualization Infrastructure (NFVI): The NFVI represents the key component of the NFV architecture, hosting the actual hardware and software resources for computation, storage and networking. NFVI creates the virtualization layer that abstracts hardware resources and includes the management of the physical/hardware infrastructure such as servers, storage, and network hardware. Through the employment of proper virtualization technologies and management platforms (such as KVM [8] and OpenStack [9], respectively) the NFVI is able to expose the physical infrastructure as bundles of virtual ones such as virtual compute (virtual machines or containerized virtual operating systems), virtual storage, and virtual networks.

Virtual Network Function (VNF): The VNFs run on top of the NFVI to deploy the software-based version of network functions (NFs). VNFs are specialized networking workloads running on top of virtualized resources. Individual VNFs can be chained together through a properly defined VNF Forwarding Graph (VNFFG) to create a NS. This software functionality should be hardware independent, but often requires hardware optimizations for performance reasons.

NFV Management and Orchestration (MANO): The NFVI and VNFs are managed by the MANO, which provisions and manages the lifecycle of both software and hardware resources. In addition, this layer creates and deletes resources and manages their allocation of the VNFs. The MANO allows flexible creation and deletion of VNFs. It is comprised of the following components:

- **Virtualized Infrastructure Manager** (VIM): The VIM manages the virtualization layer and controls how the NFVI compute, storage and network resources. VIM is therefore responsible for the control of NFVI resources including the creation, maintenance and management of virtual machines (VMs). It also operates with other management functional blocks to determine the service requirements and then manage the infrastructure resources to fulfil them.
- **VNF Manager** (VNFM): The VNFM manages the VNFs lifecycle, including the creation, configuration, maintenance, performance, and security management of VNF instances.
- **NFV Orchestrator** (NFVO): the NFV Orchestrator is responsible for the creation of network service (NS). The NFVO has a central role in the framework by covering both resource and service orchestration. It is the responsibility of NFVO to interact with the VNFM in order to establish the end-to-end service between the VNFs. NFVO also works with the VIM and has the full view of the resources that they are managing within a single point of presence (PoP) or across multiple PoPs. The NFVO in turn is composed of the following components:
    - the Resource Orchestrator (RO), orchestrating the NFVI resources across multiple VIMs;
    - the Network Service Orchestrator (NSO), managing the lifecycle of NSs;
    - the Repository, containing the NS/VNF Catalog, NFV Instances and NFVI Resources.

Reference points: apart from the three main components, the ETSI architecture depicted in Figure 1 describes also several reference points to define the information exchange between its functional blocks. The definition of these reference points ensures the flow of information to remain consistent across the different existing implementation of the functional blocks. It also guarantees the openness of the information exchange. We overview here the reference points defined in the ETSI framework:

- **Os-Ma**: This reference point defines the communication between the Operator Support System / Business Support System (OSS/BSS) and the NFVO. This is the only reference point between OSS/BSS and the management block of NFV (MANO).
- **Se-Ma**: This reference point defines the communication between the MANO and the catalogues to retrieve the information regarding the deployment of NSs and VNFs to perform the management and orchestration functions.
- **Ve-Vnfm**: This reference point is split into two sub-components (hidden in the figure):
    - *Ve-Vnfm-vnf*: it defines the communication between the VNFM and the VNFs. It is used by the VNFM for the management of the VNF lifecycle and for exchanging the configuration and state information with the VNF.
    - *Ve-Vnfm-em*: it defines the communication between the VNFM and the EM functional blocks. It supports VNF lifecycle management, fault and configuration management, and other functions, and it is only used when the EM is aware of virtualization.
- **Nf-Vi**: This reference point defines the information exchange between the VIM and the NFVI. The VIM uses the Nf-Vi to allocate, manage, and control the resources of NFVI.
- **Or-Vnfm**: This reference point defines the communication between the NFVO and the VNFM. It is used for both VNF instantiation and exchanging of VNF lifecycle-related information.
- **Or-Vi**: This reference point defines the communication between the NFVO and the VIM in order to reserve resources in the infrastructure.
- **Vi-Vnfm**: This reference point defines the information exchange between the VIM and the VNFM, such as resource update request for VM running a VNF.

- **Vn-Nf**: This is the only reference point without any management functional block as endpoint. This reference point is used to exchange performance and portability requirements from the VNF to the NFVI.

Components abstraction and their relationship: the entities deployed and managed by the NFV platform to form a NS are depicted in

Figure **2** together with their relationship. It follows an explanation of the principal components of the programming model:

- **Network Service** (NS): a service providing at least one interface to process network packets, usually formed by a composition of VNFs, linked together by Virtual VNF Links (VVLs), with a forwarding graph defined as VNFFG. The NS is orchestrated by the NFVO and described by its descriptor file (NSD), in general referencing one or more VNFDs, VVLDs, and a VNFFGD,
- **VNF Forwarding Graph** (VNFFG): provides information about the forwarding and chaining behaviour of its components VNFs and the associated VVLs. The VNFFG is described by its descriptor file (VNFFGD), and orchestrated by the NFVO.
- **Virtual VNF Link** (VVL): a virtual link connecting two VNF endpoints. The VVL is described by its descriptor file (VVLD), and orchestrated by the VNFM.
- **Virtual Network Function** (VNF): an individual, isolated component that performs a specific *Network Function* (NF) by processing network packets. It is described by a descriptor file (VNFD), and instantiated by the VNFM. It can be further decomposed into several VNF Components (VNFC) each one mapped to a VM described with a Virtual Deployment Unit (VDU) Descriptor.

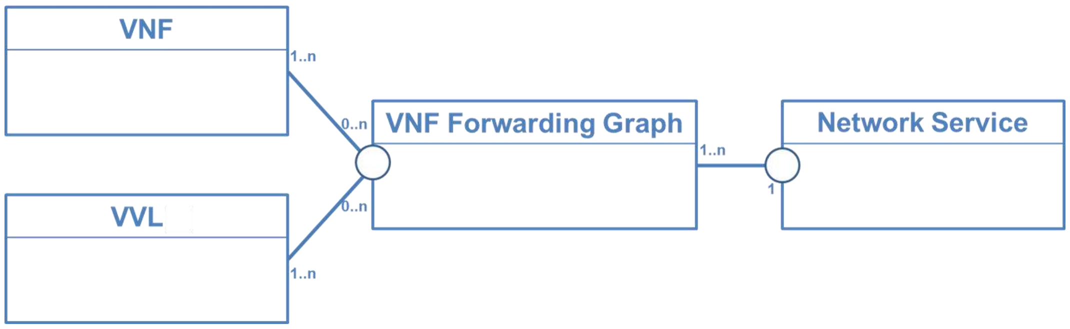


Figure 2: NFV entities and their relationship.

Information Models: appropriate models are provided for specifying and describing the relevant aspects of VNF and NS in an abstracted yet structured and unambiguous way (additional details on the information model will be provided in Section 4). Such *descriptors* map to the entities defined in

Figure 2, assuring that all the required information is included and permitting its easy parsing by the MANO.

- **Network Service Descriptor** (NSD): the deployment template for a NS referencing all the other descriptors and characterizing the end-to-end NS to be deployed. The NFVO manages the NS lifecycle based on the corresponding descriptors.
- **VNF Forwarding Graph descriptor** (VNFFGD): the deployment template for the forwarding graph, providing information about the chaining of the VNFs that compose the NS. It includes the VNFDs needed for orchestration, reference to link information, and description of Physical/Logical interfaces.
- **Virtual VNF Link Descriptor** (VVLD): the deployment template for a virtual link. It provides the information on the link type and its VNF endpoints.
- **VNF Descriptor** (VNFD): the deployment template which describes a VNF in terms of its deployment and operational behaviour requirements. It is primarily used by the VNFM for the procedures of VNF instantiation and lifecycle management of a VNF instance. The information provided in the VNFD is also used by the NFVO to manage and orchestrate Network Services and virtualized resources on the NFVI.
- **Virtual Deployment Unit Descriptor** (VDUD): the deployment template for the components of a VNF, describing the VM specification, required storage and computation resources, initiation and termination scripts, high availability redundancy model, scale out/in limits.

The role of the catalogue(s): these descriptors are collected in several specific catalogues and repositories to support the lifecycle, the management and instantiation of NS. The catalogues are generically identified in Figure 1 by the block "*Service, VNF, and Infrastructure Description*", and communicate with the NFVO by means of the reference point *Se-Ma*. The catalogues store the descriptor templates, the execution scripts, and the monitored data. The stored information is then required to offer the possibility to select, develop, test, and instantiate added-value NS. The NRG-5 catalogue will be covered in greater detail in Section 5.

2.2 MANO Open Source Projects

We present below an analysis of the existing open source NFV-based solutions. For each platform, we highlight its main features, the relevant architecture, and the implementation tools. The analysis will also provide some initial guidelines to design the NRG-5 platform, in particular to identify the features of the main modules and their relevant interfaces for NFV management. To this aim, an alignment to ETSI NFV standardization activities is strongly recommended to boost the wide interoperability with NRG-5.

The following platforms / solutions are considered for evaluation:

- **Open Source MANO** (OSM) [10];
- **Open Network Automation Platform** (ONAP) [11];
- **OpenStack Tacker** [9];
- **SONATA** [5];
- **OPNFV** [12].

2.2.1 Open Source MANO (OSM)

ETSI Open Source MANO (OSM) is an operator-led ETSI community (it is heavily backed by BT, Telefonica, and Telenor among others) that is delivering a production-quality open source Management and Orchestration (MANO) stack broadly aligned with ETSI NFV Information Models. Key reference points, such as Or-Vnfm and Or-Vi might be identified within OSM components. The VNF and Network Service (NS) catalogues are explicitly present in an OSM service orchestrator (SO) component. Its goal is to meet the requirements of production NFV networks.

The OSM platform is already suitable for evaluation purposes, and has a relatively simple and straightforward architecture. Several sample NSDs and VNFDs are also available for evaluation. As a result, the platform can be easily installed and integrated with an appropriate VIM (e.g., OpenStack) to evaluate basic NFV capabilities, trial use cases and proof-of-concepts. The project is relatively young, however, and a number of features still require development and will be available in the upcoming releases. Considering the maturity of the OSM community and the close interaction with the ETSI NFV group this project might emerge as a viable option for production-grade NFV Orchestration.

Core functionalities

- Installation, deployment and configuration of network services following the ETSI MANO specification.
- Multi-site and Multi-vendor support.
- Automating end-to-end Service Orchestration environment to simplify the various lifecycle phases involved in running a complex NFV-based service.
- Delivering a plug-in model for integrating multiple SDN controllers.
- Delivering a plug-in model for integrating multiple VIMs.
- An integrated "Generic" VNFM with support for integrating "Specific" VNFMs.
- Support to integrate Physical Network Functions into an automated Network Service deployment.
- GUI, CLI, and REST interfaces to enable access to all management features.

Specifications

- Delivery of a capability for Create/Read/Update/Delete (CRUD) operations on the definition of Network Service.

- Resource Orchestrator (RO) managing the lifecycle of VNFs.
- Supporting a Model-Driven environment with Data Models aligned with ETSI NFV MANO.
- Simplified VNF Package Generation.
- Graphical User Interface (GUI) to improve the network service design time phase.
- Multi-VIM support expanding OSM so that VMware, multiple versions of OpenStack, OpenVIM and Amazon Web Services Elastic Compute Cloud are enabled.
- Multi-Site support enabling automated service delivery across multiple sites, where a site is represented as a grouping of infrastructure managed by VIM.

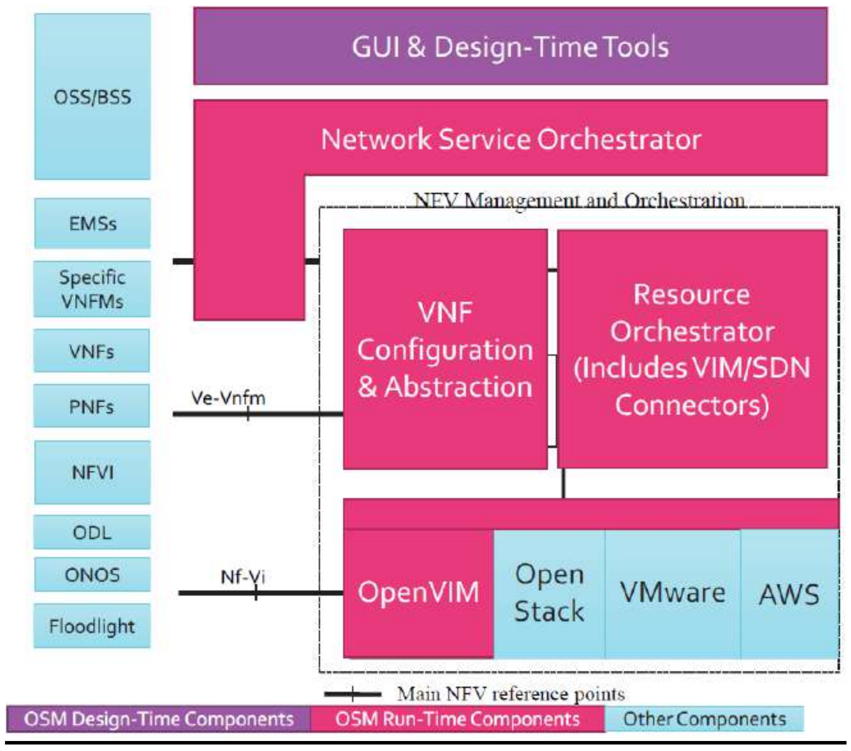

GUI & Design-Time Tools
OSS/BSS
Network Service Orchestrator
NFV Management and Orchestration
EMSs
Specific VNFMs
VNFs
PNFs
NFVI
ODL
ONOS
Floodlight
VNF Configuration & Abstraction
Resource Orchestrator (Includes VIM/SDN Connectors)
Ve-Vnfm
Nf-Vi
OpenVIM
Open Stack
VMware
AWS
Main NFV reference points
OSM Design-Time Components
OSM Run-Time Components
Other Components

Figure 3: The OSM Components are aligned with the ETSI Architecture.

The Open Source MANO (OSM) platform consists of 3 basic components:

- The **Service Orchestrator (SO)**, responsible for end-to-end service orchestration and provisioning. The SO stores the VNF definitions and NS catalogues, manages workflow of the service deployment and can query the status of already deployed services.

- The **Resource Orchestrator (RO)** is used to provision services over a particular IaaS provider in a given location. It stores information about all VIMs available for deployment. The RO component is capable of deploying networking services over different existing NFVI, such as OpenStack, VMware, OpenVIM and AWS. The SO and RO components in Figure 3 can be jointly mapped to the NFVO entity in the ETSI MANO architecture presented in Figure 1.

- The **VNF Configuration and Abstraction (VCA)** module performs the VNF configuration and management. The VCA layer is aligned with the VNFM in the ETSI-NFV MANO specification.

Additionally, OSM hosts the OpenVIM project, which is a lightweight VIM layer implementation suitable for small NFV deployments as an alternative to heavyweight OpenStack or VMware VIMs. The GUI for OSM, provides a simple and intuitive way to interact with the SO, tools for easy on-boarding of VNFs, and a drag-and-drop environment for creating NSs from the VNF catalog.

2.2.2 Open Network Automation Platform (ONAP)

ONAP is an open source project backed by the Linux Foundation that enables telecommunications and cable operators to effectively deliver end-to-end services across Network Functions Virtualization (NFV) Infrastructure, as well as Software Defined Network (SDN) and legacy network services. It is the result of a merger of two projects: **OPEN-O** (supported by Huawei, Brocade, Ericsson, Intel, Red Hat) and Open Source **ECOMP** (the open source version of AT&T's ECOMP project).

ECOMP automates the design and delivery of network services that run on a cloud. In addition to service delivery and automation of SDN tasks, ECOMP also automates many service assurance, performance, and fault management tasks. ECOMP adds several capabilities over and above the common ETSI MANO architecture. It addresses the entire service delivery lifecycle using model driven architectures. For example, it has a rich service design environment for service creators to build services using a collaborative, catalogue driven self-service design studio. The design studio also creates NS, VNF, and infrastructure descriptions that are richer in scope than what's required by the ETSI architecture by adding numerous pieces of metadata. NFVO and VNFM functionalities are enhanced for greater control over NFVI, VIM, and SDN controllers, and for faster on-boarding of new VNF types. ECOMP supports YANG, TOSCA, OpenStack Heat and other modelling languages. Finally, the project includes FCAPS (Fault, Configuration, Accounting, Performance, and Security) functionality to provide greater control over closed loop automation (in the ETSI architecture this functionality resides in the EMS – Element Management System).

OPEN-O is a Linux Foundation open orchestration project that combines NFV MANO and connectivity services orchestration over both SDN and legacy networks. OPEN-O employs a model-driven automation approach, and has adopted standard modelling languages YANG (for networking devices) and TOSCA (for services). The OPEN-O architecture is partitioned into three main orchestration functions: global services orchestrator (GSO), SDN orchestrator (SDNO), and NFV Orchestrator (NFVO), along with a set of common services. OPEN-O is built upon micro services architecture for extensibility, and features a modular design that supports multiple VIMs, VNFMs, SDN Controllers, and legacy network and element management systems.

Core Functionalities

- Enable operators to capitalize on NFV and SDN architecture.
- Multi-domain, multi-location, and end-to-end service composition.
- Adopt industry-wide common information model and TOSCA/YANG data models to ensure extensibility, implementation efficiency, and interoperability.
- Provide a common platform for VNF vendors, simplifying and accelerating VNF onboarding, development, and deployment.
- Enhance the overall service lifecycle, through model-driven automation to enable reuse and incremental upgrades.
- Provide abstraction to operate over diverse SDN, NFV, and legacy network infrastructures.

Specifications

- Multi-VIM and multi-vendor, multi-site support.
- Multi-SDN controllers support (e.g., Open Daylight, ONOS).
- Aligned with ETSI NFV (architecture and data models).
- Compatible with most of the existing Open Source projects (e.g., OPNFV, Open Stack, Open Daylight, ONOS).
- A generic VNFM, which can be easily extended for supporting different type of VNFs.
- It extends ETSI-NFV MANO modules and includes analytics module and policy module to provide more intelligent management of the VNFs.
    - Machine learning methods to monitor and control the system.
    - The analytics module can predict the future situation and interact with the policy module to control / manage the overall system behaviour.

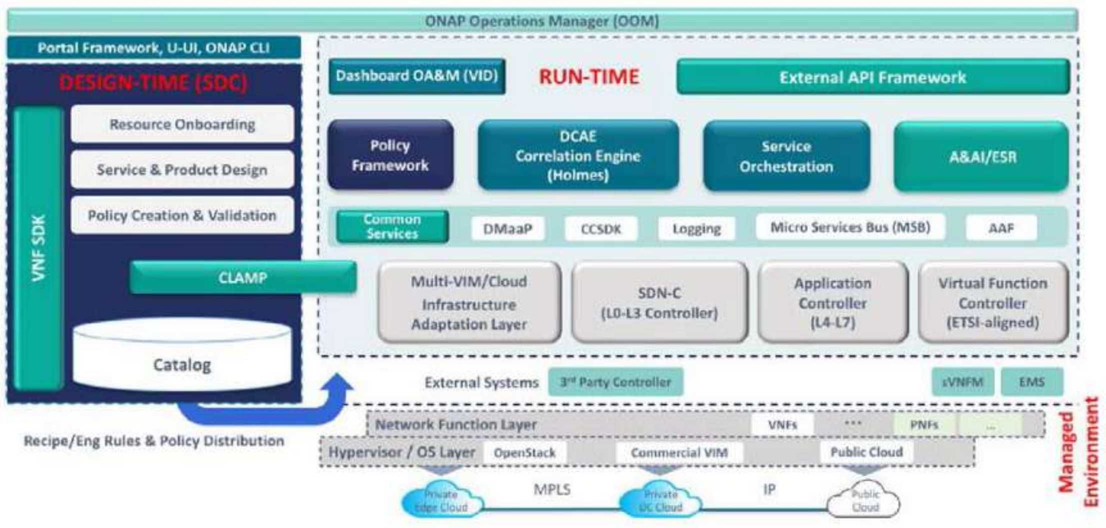


Figure 4: ONAP Platform Components (Amsterdam Release).

Differently from OSM, the ONAP platform provides a set of common functions (e.g., data collection, control loops, meta-data recipe creation, policy/recipe distribution, etc.) necessary to construct specific automated behaviours. In order to create a service, the developer must define service/operations-specific collection, analytics, and policies (including recipes for corrective/remedial action) using the Design Framework Portal.

The ONAP platform is composed of 3 core components, as depicted in Figure 4:

- The **Portal** is responsible for the access to the design, analytics and operational control/administration functions via a dashboard. The portal architecture provides several capabilities such as the application onboarding and management, the access management, and a UI dashboard. The Portal acts as a Software Development Kit (SDK), giving access to a set of built-in capabilities (Services/ API/ UI controls), tools and technologies. ONAP also provides a Command Line Interface (CLI) for operators requiring scripting capabilities.

- The **Design time Framework** provides the capabilities to model resources, services and functions using a common set of specifications and policies for controlling their behaviour and execution. It allows the platform to automatically handle their instantiation, delivery and lifecycle. ONAP provides a set of VNF packaging and validation tools in the VNF Supplier API and Software Development Kit (**VNF SDK**) component. The **Policy Creation** component deals with policy abstraction; these are conditions, requirements, constraints, attributes, or needs that must be provided, maintained, and/or enforced. Policy permits management/control of complex mechanisms. The **Closed Loop Automation Management Platform** (**CLAMP**) allows designing and managing control loops.

- The **Runtime Framework** executes the rules and enforces the policies distributed by the design and creation environment. This distributes the policy enforcement and templates among various runtime modules such as the Service Orchestrator (high-level), Controllers (to apply configuration and policies), Data Collection, Analytics and Events, Active and Available Inventory (providing real-time views of a system's resources, services, products and their relationships with each other), and a Security Framework. These components use common services that support logging, access control, and data management.

2.2.3 OpenStack Tacker

Tacker is an official OpenStack project that started off as a generic VNFM but has since expanded its scope to provide an NFVO as well, making it a complete MANO project. Being a part of OpenStack, it is also well integrated with other OpenStack components such as Nova [13], Horizon [14], Heat [15], Keystone [16], and so on. The main goal of Tacker is to deploy and operate VNFs and NSs on top of a virtualization infrastructure. Its main contributors are Brocade, Infinite, NEC, 99cloud, Huawei, Imagea, Nokia, and China Mobile. Because Tacker can be used as just a VNFM, it can also coexist with other open end-to-end service orchestration projects such as ONAP, OSM.

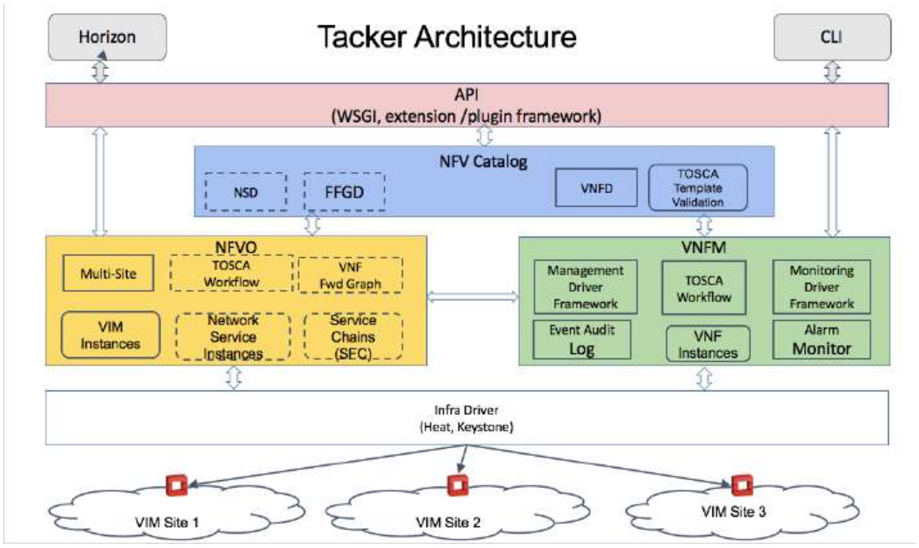


Figure 5: High-level Tacker Architecture.

<u>Core Functionalities</u>

- Installation, deployment and configuration of network services.
- Multi-site support.
- Provides independent infrastructure slices.
- Support for generic or specific VNF management.
- Runtime operations: fault management, and auto-scaling.
- A large amount of virtualization use cases e.g. core networks, M2M and Multimedia communication.

<u>Specifications</u>

- Generic VNF Manager for the life cycle management of the VNFs based on the corresponding descriptors.
  - VNF Instantiation and Termination using Heat - TOSCA to Heat translation in Tacker
  - VNF Configuration injection during instantiation, update and restart - Loadable VNF specific management-driver.
  - Loadable per-VNF Health Monitoring.
  - Self-Healing according to VNFD policy.
- NFV Information Model/Data Model
  - Tacker closely works with OASIS TOSCA NFV Working group.
  - Based on the just-released CSD03 version of OASIS TOSCA Simple Profile for NFV.
  - Participating in cross-SDO events.
  - Introduced TOSCA NFV Profile support into TOSCA-parser.
  - Transition from in-built translator to TOSCA-parser in progress.
- NFV Orchestrator managing the lifecycle of Network Service Descriptors and interfacing with one or more VNF Manager(s).
- VNF Auto Scaling
  - Auto-Scale VNF based on policy.
  - Continuous performance monitoring according to KPI described in VNFD.

  - Basic Auto-Scaling using common VM metric.

Tacker includes a user-user friendly dashboard which enables the management of the complete environment. To interact with OpenStack Tacker, each module offers a set of Representational State Transfer (REST) APIs that can be used to provide parameters and retrieve the responses. The modules also have scripts that act as a wrapper around those REST APIs and offer a command-line tool for passing the parameters or displaying the response, making the interaction a bit friendlier for users and for admins. Also, there is a graphical user interface available for Tacker.

2.2.4 SONATA

The Sonata platform is the open source outcome of the SONATA 5G PPP Phase 1 project ending in 2018 [17].

Sonata embeds a NFV framework providing a programming model and development toolchain for virtualized services, fully integrated with a DevOps-enabled service platform and orchestration system. An SDK is employed to ease the development of network services for third party developers with both a programming model and a set of software tools. The tight integration between the SDK and the service platform bridges the gap between design and development of services on the one hand and deployment and lifecycle management of them on the other hand. Using the set of tools and techniques provided by the SONATA architecture, service developers and operators can collaborate to optimize the design and implementation of services, test and debug services using runtime information like monitoring or performance data, and scale and adapt services to rapidly changing load and network resources. This concept, known as a DevOps workflow in the software engineering community, supports agile service development and operation and takes the SONATA architecture beyond existing NFV platforms that focus on either the development or the operation side.

In addition, it proposes the **Function Specific Manager** (FSM) / **Service Specific Manager** (SSM) concept to be implemented by a network function/service and shipped with the service package. Using these managers, the SONATA platform offers an increased level of flexibility to network function/service developers by adding programmability directly to the management and orchestration system. This goes beyond existing orchestration approaches based on decision taken at the MANO, where service management strategies are either limited to a predefined set of strategies or simple, customizable rules. FSMs/SSMs, in contrast, can be complete programs that can consume information like monitoring data, do complex computations to optimise their decisions, and instruct other components of the system to act accordingly.

Core Functionalities

- Representing the NFVO and VNFM functional blocks of the ETSI architecture.
- Network Service SDK to edit, package, debug and test, and deploy VNFs/NSs for third-party developers.
- Micro-service architecture for each service (catalogue, gatekeeper, MANO) based on Docker containers.
- DevOps Workflow designed for agile development and operation of network services.
- Multi-VIM, multi-vendor, multi-site support.
- Several use cases and pilots, such as Virtual Evolved Packet Core (vEPC), virtual Content Delivery Networks (vCDN),

Specifications

- SSM/FSM plugins managing the lifecycle of VNFs/NSs.
  - SSM/FSM are service/function generic manager for basic VNF/NS actions such as start, stop, scaling, placement.
- Customizable by design, including swappable modular plugins, such as lifecycle management, service monitoring, conflict resolution.
- Support for 5G Slicing and Recursion

- Slicing support delivers performance isolation and bespoke network configuration targeting vertical application as foreseen in 5G networks.
- Recursion support allows stacked tenant and wholesale deployments in new software networks business models.

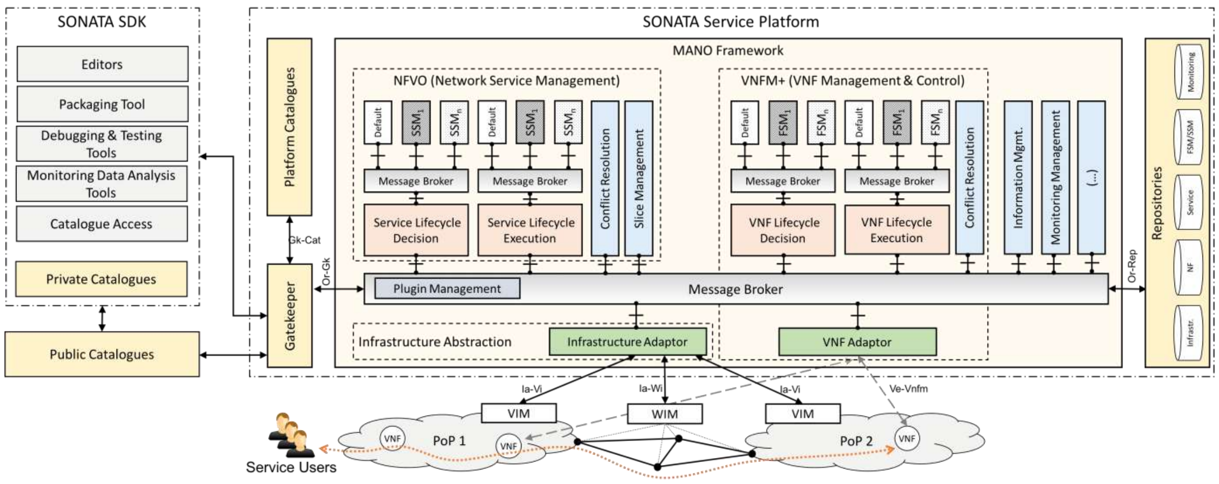


Figure 6: SONATA Service Platform Architecture.

The major component of the SONATA platform is the **Software Development Kit (SDK)**. The SDK allows developers to define complex services consisting of multiple VNFs. A service developer can then deploy and manage the via a gatekeeper components. Services and their components can also be published in catalogues to be reused by other service developers and providers. The SONATA's **Service Platform (SP)** has a customisable and modular design of MANO framework. The SP operator can modify the platform, to support a desired business model, by replacing components of the loosely coupled MANO framework (MANO plugins). A service developer can influence the orchestration and management functionalities of the platform pertaining to their own services, by bundling small management programs, so-called function- and service-specific managers (FSMs/SSMs), with their services. SONATA is designed for agile development and operation of network services. It enables a **DevOps workflow** by virtue of the full compatibility and integration between the SDK tools and the service platform. In this way, developers and operators can collaborate on design, development, deployment, and lifecycle management of network services, as well as optimise and adapt the design and implementation of the services based on collected monitoring information regarding the current state of the service and network resources.

2.2.5 Open Platform for Networks Functions Virtualization (OPNFV)

The Open Platform for NFV (OPNFV) is an open source project founded and hosted by the Linux Foundation, and composed of several TSPs and vendors (among others, Orange, Huawei, Intel, Ericsson, ZTE are present). OPNFV integrates and test the end-to-end stack to support NFV with verified capabilities and characteristics, establishes agile reference methodologies (requirements, documentation and propagation; continuous integration, testing, and continuous delivery), and offers a process and supporting tools for testing and validating NFVI and MANO products and solutions.

OPNFV is different to the other surveyed projects in the sense that it does not attempt to build a solution but rather to integrate a system as an open community effort together with the upstream projects (e.g., OpenStack) [18]**.**

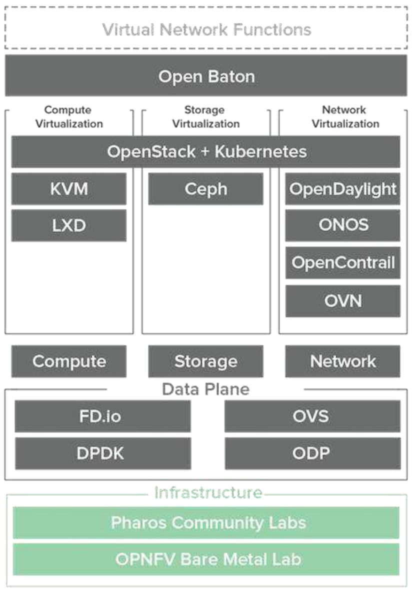


Figure 7: Upstream projects used in OPNFV (Euphrates Release).

OPNFV integrates multiple software projects for the same purpose such as OpenStack, OpenDayLight (ODL), FD.io. In this way, OPNFV enables choices for end users through multiple options, and it is not OPNFV's role to recommend one technology over another. As a testing and integration project, OPNFV brings together upstream components across compute, storage and network virtualization in order create an end-to-end platform. **Activities within OPNFV focus on integration of components and automated build and deployment of the integrated environment**. Continuous integration and automated testing of the platform for NFV use cases is key to ensure that the platform meets NFV industry needs. Another key focus is refining new requirements and features necessary to the platform and working closely with upstream communities to incorporate these capabilities within their code bases.

2.2.6 Concluding Remarks

The open source NFV projects surveyed in this section starts with different objectives, implying different architectural choices. However, all of them refer to ETSI NFV MANO architecture definition. A few of them have introduced some additional layers and components, which are out of scope of the MANO architecture. The table below provides a summary of this comparative study, especially highlighting:

- Mapping to the ETSI MANO architecture.
- Integration with SDN Controllers.
- Data and information models.
- Support for multi-site, multi VIM deployments.

Project	NFVO	VNFM	VIM	Descriptor Format	SDN Controllers	Multi-Vendor	Multi-Site
OSM	Resource Orchestrator	Juju	OpenVIM, OpenStack, VMWare	JSON, YAML	ODL, ONOS	Yes	yes
ONAP	NFVO	G-VNFM (Tacker)	OpenStack, AWS	HEAT, TOSCA, YAML,	ODL, ONOS	Yes	Yes
Tacker	NFVO	Generic VNF Manager	OpenStack	TOSCA	ODL	Yes	Yes
SONATA	NFVO	FSM/SSM	OpenStack	-	ODL	Yes (OpenStack preferred)	Yes
OPNFV	-	-	OpenStak, Kubernetes	-	ODL, ONOS, OpenContrail, OpenStack Neutron, OVN	-	-

Table 1: Open Source NFV projects comparison.

3 Requirements Extracted from the Use Cases

This section maps the high-level requirements extracted from the use case defined in D1.1 to a set of technical requirements that have to be expressed in the catalogue. By using the information model defined in Section 0, the NRG-5 platform shall be able to describe and meet the requirements (in terms of KPIs) for each Use Case (UC).

3.1 NRG-5 Use Cases Overview

The analysis starts by detailing the three proposed use cases and overviewing their requirements. Table 2 covers the KPIs for each UC using the 5G PPP definition and the values provided in D1.1 [1].

KPI Requirements	UC 1	UC 2	UC 3
User Density	High	Low	Medium – OBD, Scheduling High – Charging
User Data Rate	Medium	High	Low – OBD, Medium - Scheduling, Charging
Mobility	Static	High	High – OBD, N.A. - Scheduling, Charging
Infrastructure	High	Low	Medium
Traffic Type	Period Event driven	Burst – alarms Cont - video	Event driven – OBD Cont – Scheduler, Charging
Latency	Low	Low – alarms High – video	High – OBD Low – Scheduling, Charging
Reliability	High	High	High
Availability	High	High	High

Table 2: KPI requirements for the Use Cases proposed by NRG5 [1].

3.1.1 UC1: Realizing decentralized, trusted lock-in free “Plug & Play vision”

UC1 proposes that prosumers having an excess of energy produced by renewable energy resources and the availability of storage batteries could both sell their energy or provide flexibility / balancing services inside their local micro-grid.

To do this, any energy consumer or prosumer must be capable of negotiating dynamically the agreements to buy and/or sell energy within the energy market of its local micro-grid rather than from the general energy market, based on the aim of increased efficiency and resilience towards sustainable communities. In fact, when not modified by external (sometimes artificial) measures such as subsidies distorting the market competition, the local market could represent most of the times the preferred commercial choice. In the UC1, it is considered that the local micro-grid market has a priority by default against external/general markets in order to encourage efficiency and resilience. Only the energy needs that cannot be covered by local resources or that result much more costly than the general market are purchased from outside the local micro-grid. These agreements have to be considered as **micro-contracts** with a temporal and/or spatial validity. A consumer is able to select dynamically the most cost-efficient energy provider, while a prosumer is also capable of selecting the most profitable energy provider for selling back the energy produced locally.

UC1 aims at creating a lock-in free energy service based on this transactive oriented paradigm. Due to micro-contracts (packaged in the form of blockchain-based smart contracts), both consumers and prosumers are not bound anymore to a specific energy provider, having the capability to switch on-the-fly and on-demand the energy providers. The minimum time period to be bounded to a specific vendor is the minimum time for a micro-contract, e.g. 15 minutes but also down to one minute, which require an extremely dynamic and low latency ICT support, to solve the very short time processes associated with each fine-grained transaction.

Within the focus of the NRG-5 project, UC1 needs an easy, real time, and automated device identification so that network configuration can be achieved automatically, under a highly heterogeneous landscape in terms of devices and services. Next-generation Smart Meters such as NORM (developed under the Unbundled Smart Meter architecture) allow for the real-time monitoring of both the grid status and the energy consumption behaviour of stakeholders. These enhanced capabilities permit supporting several value-added services such as implementing powerful control mechanisms and/or flexible billing processes. Moreover, in a Smart Grid scenario fed by renewable energies, real-time Smart Meters will enable real time energy optimization and management operations.

To be noted that for the UC1 the most demanding requirements are reliability, availability and low latency of the 5G communication, as the micro-transactions and the real-time control relies essentially on the communication, to allow transactions in time frames down to 1 minute and real-time control with control loops down to 1 second.

3.1.2 UC2: Enabling aerial Predictive Maintenance for utility infrastructure

UC2 proposes to automate as much as possible the monitoring activities of critical scenarios related to the energy domain (e.g., activities performed on gas and electric infrastructures) by using drones. Several activities are presented, such as the predictive maintenance of infrastructure, the detection of incidents, the perimeter control, the worksite areas support, the research of intruders, and third party inspections. As of today, most of these activities can be only carried out by manually piloted drones.

To perform such duties, UC2 proposes to automate the flight plan execution, coordinate with multiple drones, handle and detect unexpected events. Moreover, information gained through the drones sensors should be automatically processed and analysed to raise alarms and status of operations. Since processing power on drones is limited and 5G will permit to have high data rate links, data processing and flight control are offloaded and performed on the network edge (i.e., near a 5G base station antenna) by exploiting the Mobile Edge Computing (MEC) paradigm. MEC guarantees the best balanced trade-off in terms of control latency vs. processing power.

For the UC2, the most demanding requirements are low latency and guaranteed data rates, as the flight control will require fast response times and high throughput to carry out video analysis in real-time.

3.1.3 UC3: Enabling resilience and high availability via Dispatchable Demand Response

UC3 deals with the challenges associated with the introduction of charging electric vehicles (EVs) at residential locations. In this use case, we foresee that a charging peak-load period could potentially overlap with the residential peak-load period, making energy management more challenging.

In UC3, an aggregation entity manages a portfolio consisting of EVs, Distribute Energy Storage (DES) and Distributed Energy Resources (DERs). This UC demonstrates the capability to minimize the imbalance in the grid due to the unpredictable renewables generation. Moreover, a method for fast fault isolation and smart grid recovery via power flow rerouting is required.

In order to satisfy the power network requirements it is possible to adopt a “power smoothing” strategy. Electricity storage technologies operating on short timescales (seconds, minutes, hours) are used to provide a buffer between electric supply and demand. These power quality management technologies can fill the gaps between actual electricity demand and an average, smooth demand curve that is easier for typical supply sources to follow.

Globally UC3 proposes to manage in real-time the flow of utility charge by taking into account the energy requirements of the EV charging point, the energy production and the charge reservation.

3.2 Requirements for the NRG-5 Platform

From the analysis of the UCs, it is possible to extract a set of requirements that map on a set of properties to be defined in the instantiation phase. This part is directly related with the catalogue capabilities and the related information model. The MANO platform should then be able to take decision and handle the lifecycle of the network service by taking into account the requirements as defined during the instantiation phase.

For now, we focus on the main requirements that we can address by acting on the NRG-5 platform, namely **latency**, **availability**, **reliability**, and **bandwidth**.

3.2.1 Latency

The respect of end-to-end latency can be ensured by reducing the distance and number of hops between the requesting user and the content. End-to-end latency includes all the elements in the end-to-end service (VNFs and infrastructure components) with the exception of the customer terminals. For NRG-5, this requirement translates into placing VNFs demanding critical latencies close to the user, by distributing them and applying caching strategies. Also, for better performance, such VNFs shall be placed on computing platforms capable of fast-path data plane packet processing frameworks such as Single-Root Input/Output Virtualization (SR-IOV) [19] and Intel Data Plane Development Kit (DPDK) [20], reducing the overhead due to the kernel traversal of virtualized environments. This requires the support from the NFVI and it is generally named under the umbrella term of Enhanced Platform Awareness (EPA) [21].

Also, latency must be monitored by the platform in order to be guaranteed in any situation. Whenever violations are detected, an alarm should be raised and the VNF automatically moved to a computing platform capable of guaranteeing the respect of required latency values. By reducing latency, it is also foreseen to have a positive impact on throughput and overall capacity due to the reduced queuing times.

With respect to the previous analysis, the **UC1 is the most demanding use case in terms of latency requirements**. In this UC, the transaction timeframes are down to each one minute, while fulfilling the transactions technical duties in real time requests micro-grid control with control loops down to one second.

This real-time control is from this point of view more demanding that the similar requirements at Transmission System Operator (TSO), e.g. when we look at the control loops for the Automatic Generation Control (AGC) at TSO level, which currently has an effective loop control of 2 to 4 seconds, but the system reaction time is even higher as the measurements acquisition accounts for additional similar time. The reason for this demanding control loops at micro-grid level relies on the fact that microgrids are less inertial due to the lower number of loads and generation units that makes the mediation less effective, thus the production-load equilibrium more volatile, requiring quicker time of reaction.

With one second control loop, the latency for exchanging (sending and receiving) messages needs to be between an order and two orders of magnitude less than the time of an entire control loop, thus requiring a maximum latency ranging between 10 to 50 ms.

An average of 20 ms latency with a maximum of 50 ms is expected to meet the target of acquiring data, making computations on balance control within the micro grid and sending new commands. An explanation of the process dynamics is proposed in Figure 8.

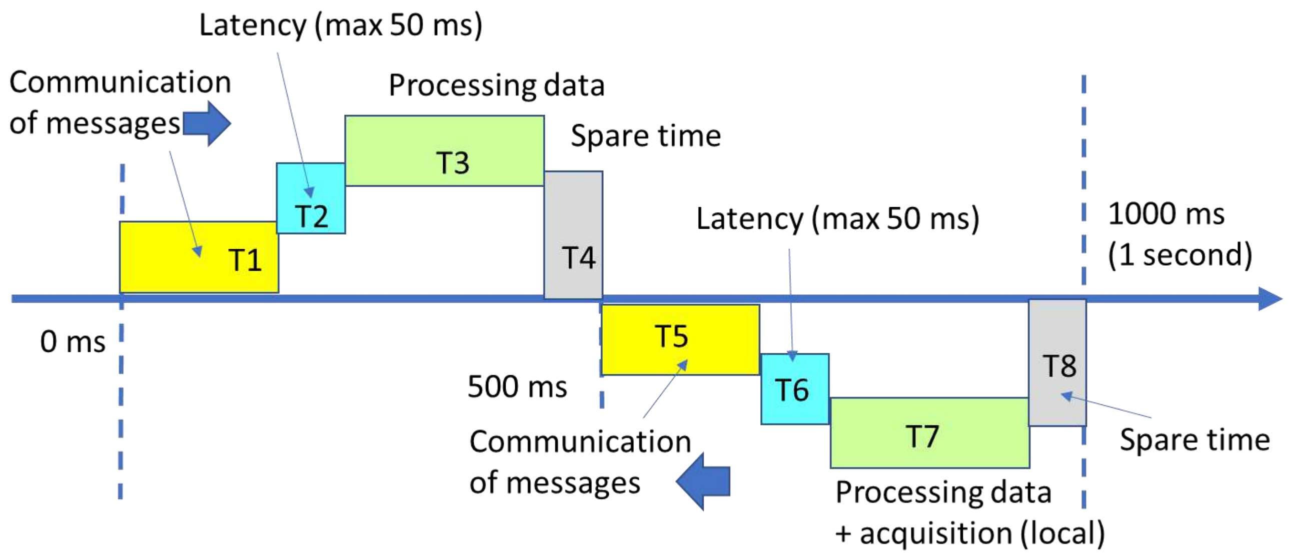


Figure 8: Dynamics of process control at highest rate, with latency.

Such a low latency is usually obtained in wired networks, such as Ethernet, but with new 5G communication it is intended to test a low latency requirement in order to analyse the limits of the new technology and to determine practical implementation parameters (e.g. longer control loops such as 2 to 5 seconds and longer commercial time frames for the transactions such as 2 to 5 minutes).

3.2.2 Reliability and Availability

Service reliability and availability levels are strictly connected together. Reliability is defined as the probability that a certain amount of data from a service is transmitted successfully within the latency constraint (erroneous, lost or late). Availability is defined as the fraction of users for which the reliability level is above the acceptable level. In the case of a radio link it may define the coverage area [22]. Typically, **mission-critical services such as those related with the energy distribution needs to be supported by high reliability and availability values** (five nines and beyond) because the total cost of system failures can be tremendous.

For software networks, these metrics are strictly bound on the underlying infrastructure reliability. Service continuity must be assured by a secure and always available network with redundancy. In order to meet the specific requirements of the three presented use case, the NRG-5 platform must provide high service availability and reliability levels, regardless of the heterogeneous underlying infrastructure on which the VNFs are deployed. Whenever the resilience of the underlying infrastructure is uncertain, the survivability of software services must be compensated with advanced runtime adaptive functionalities and highly dynamic service adaptation mechanisms. The logical interconnection of the VNFs should be maintained by the NFVI, e.g. in case of failure the connection should be restored within an acceptable tolerance time.

Indeed, multiple issues can undermine the system stability, ranging from hardware to software faults, natural disaster, security weakness and malicious attacks. In most cases, service and/or network unavailability is not an option, and appropriate mechanisms must be implemented on the platform to guarantee the reliability and availability of critical VNFs.

As also happens for the case of latency, reliability and availability levels cannot be fully predicted at service deployment time. Monitoring, testing and statistical analysis are among the mechanisms which the platform can adopt to achieve the appropriate accuracy levels.

Current NFV systems embed practically no mechanisms to support reliability in an end-to-end manner over heterogeneous and multi-domain infrastructures. This issue is mainly due to the lack of a method to gather and expose dynamically the reliability capabilities of the infrastructure and of the virtualized infrastructure management. This issue cannot be solved in a conventional way, due to the extremely high complexity in term of possible states impeding the definition of a state machine and the impossibility of appropriate testing and validation.

In order to fulfil high reliability for the system, the temporary failure of system control due to communication unavailability requires the detection of such situations and temporary exit from micro-grid control until the communication parameters are restored.

Indicatively, the availability should be between four and five nines (respectively 52, and 5 minutes of downtime per year year). A Quality of Service (QoS) mechanism will be needed to be implemented for the micro-grid control over the 5G network, giving statistics related to reliability and availability as well as for the latency in the communication infrastructure.

3.2.3 Bandwidth

Dedicated and guaranteed bandwidth is required for certain capacity hungry applications (especially true for UC2). As of today, processing workloads at high speeds in a virtualized environment represents a challenge. The software stack must be highly optimized to run at wire-speed with minimum sized packets. In this context, memory management and scheduling are key concerns when developing platforms geared towards NFV.

In fact, network-heavy services break the classic paradigm of cloud computing, where the compute operations interrupt the virtual machine (VM) when there is a new packet from the network, requiring a kernel context switch, and a lock to be acquired to ensure proper memory protection. Together with the fact that data is copied several times from the kernel to the user space, they prevent the ability to scale to support processing of millions of packets per second. Thus, to achieve the required bandwidth, these inefficient operations must be avoided. Key operations requiring optimization include buffer management, ring management, queue management, flow classification, and network driver architecture. An example of these techniques is Intel DPDK [20], a set of open source libraries and drivers developed to address the requirements of the data plane, or forwarding engine, of VNFs for fast packet processing.

4 Information Model

4.1 Introduction

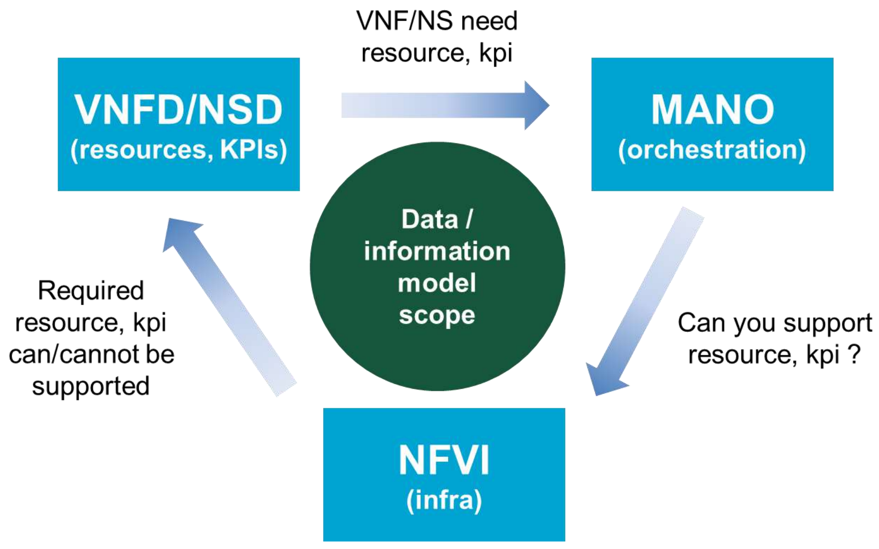
Information and data models provide a high-level abstraction to formalize the description of the entities involved into a process, without any constraint on the specific implementation, device or protocol used. For this reason, the information and data models are typically defined before the actual implementation of any system, laying the foundation for the information exchange taking place between functional entities into a complex architecture.

More in detail, an information model is required in NRG-5 both to agree on the high-level information that are needed to be supported, and to ensure the interoperability among the different functional blocks composing the NRG-5 architecture and between their communication protocols.

In general, an information model is needed in any NFV deployment in order to describe the features of the virtualized instances and to guarantee the flexibility of deployment and portability over different NFVI environments (having varied networking and computing resources, using different virtualization techniques). As depicted in Figure 9, any VNF/NS should be able to describe its resource requirements and expected KPIs (minimal/mandatory and optional). The NFVI should be able to abstract and expose the platform features, allowing thus the MANO (NFVO) to take optimal orchestration decisions.

Additionally, any VNF/NS package must also carry a set of information needed for the lifecycle management of the associated network function/service.

- Functional descriptors (VNFD/NSD);
- Metadata, scripts, and proprietary artifacts;
- Eventually, software images for its VNF components (VNFC).



VNF/NS need resource, kpi
VNFD/NSD (resources, KPIs)
MANO (orchestration)
Data / information model scope
Required resource, kpi can/cannot be supported
Can you support resource, kpi ?
NFVI (infra)

Figure 9: Requirements and scope of the NRG-5 data/information model.

4.2 Virtual Network Function and Service Descriptors

As already hinted in Section 2.1, the concept of descriptor is widely used to specify and describe the components of a NS. A **VNFD/NSD** describes a VNF/NS in terms of both deployment and runtime requirements. The VNFM uses these templates for instantiating an instance of the requested network function/service and to handle its lifecycle. The information provided in the VNFD/NSD is also used by the NFVO to manage and orchestrate network services and the virtualized resources on the NFVI. VNFD also abstracts connectivity, interfaces and KPIs requirements that may be used by NFV MANO to establish appropriate virtual links within the NFVI between its VNFC instances, or between a VNF

instance and the endpoint interface to the other NFs. The VNFD contains also the information needed for the lifecycle management, as the VNF identification, but as well the references to other components, as the internal network description (e.g., between VNFCs composing the same VNF, or between VNFs composing the same NS), the VDU description for each VNFC (corresponding to a VM type) by defining the VM flavours, the software images to be used, and the number of VMs to activate.
In general, a NS is composed of one or more VNFs, each one composed in turn of one or multiple components (VNFC). Moreover, a VNFC define a VDU.

In order to be as concrete as possible, we present below an example of the information model adopted in OSM, which will serve as well as the basis for the NRG-5 descriptors. Figure 10 exemplifies the structure of a NSD composed of two VNFs chained together and described by the related VNFD[1]. Please note that the exact meaning of each field will be explained later on.

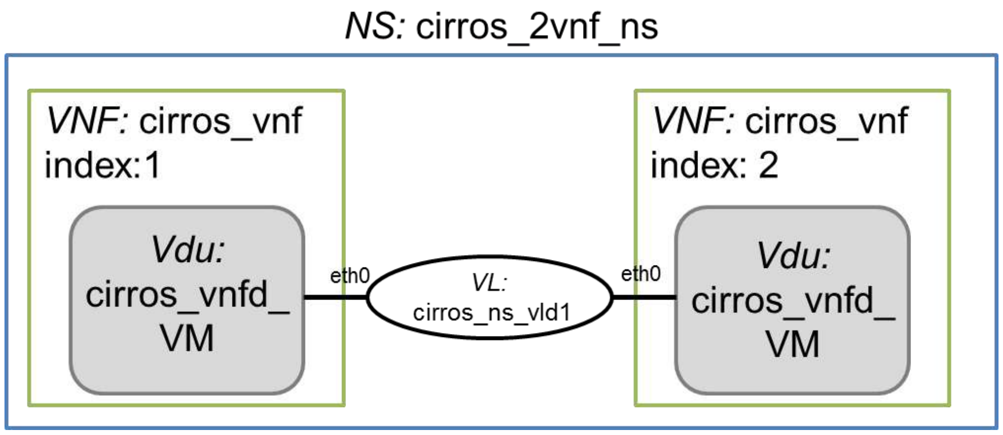


Figure 10: Graphical representation of the NS to deploy.

In the listing proposed in Figure 11 is possible to read the identification information for a NS univocally identified as `cirros_2vnf_nsd` (`id`, `name`, `description`, `vendor`, `version`, and `logo` fields). Moreover, the `constituent-vnfd` section defines the VNFD that constitutes the NS (`member-vnf-index` and `vnf-id-ref` fields providing the identification), together with the virtual network created among the constituent VNFs (defined in the `vld` section).

[1] This example is taken from the OSM website [35] and its syntax is based on YAML [33].

```
nsd:nsd-catalog:
    nsd:
    -   id: cirros_2vnf_nsd
        name: cirros_2vnf_ns
        short-name: cirros_2vnf_ns
        description: Generated by OSM pacakage generator
        vendor: OSM
        version: '1.0'

        # Place the logo as png in icons directory and provide the name
here
        logo: osm_2x.png

        # Specify the VNFDs that are part of this NSD
        constituent-vnfd:
            # The member-vnf-index needs to be unique, starting from 1
            # vnfd-id-ref is the id of the VNFD
            # Multiple constituent VNFDs can be specified
        -   member-vnf-index: 1
            vnfd-id-ref: cirros_vnfd
        -   member-vnf-index: 2
            vnfd-id-ref: cirros_vnfd

        vld:
        # Networks for the VNFs
            -   id: cirros_2vnf_nsd_vld1
                name: cirros_2vnf_nsd_vld1
                short-name: cirros_2vnf_nsd_vld1
                type: ELAN
                mgmt-network: 'true'
                # vim-network-name: <update>
                # provider-network:
                #     overlay-type: VLAN
                #     segmentation_id: <update>
                vnfd-connection-point-ref:
                # Specify the constituent VNFs
                # member-vnf-index-ref - entry from constituent vnf
                # vnfd-id-ref - VNFD id
                # vnfd-connection-point-ref - connection point name in the
VNFD
                -   member-vnf-index-ref: 1
                    vnfd-id-ref: cirros_vnfd
                    vnfd-connection-point-ref: eth0
                -   member-vnf-index-ref: 2
                    vnfd-id-ref: cirros_vnfd
                    vnfd-connection-point-ref: eth0
```

Figure 11: Example NSD composed of two VNFs chained together.

Next, the listing in

Figure **12** proposes the definition of the VNF invoked by the NSD `cirros_2vnf_nsd`. Similarly as before, it is possible to find the identification information for the VNF `(id`, `name`, `description`, `vendor`, `version`, and `logo` fields), the name of the management interface (`mgmt-interface`), the components of the VNF in terms of VDUs (`vdu` section providing in particular the requirements for the VM flavour), and the network interfaces of the VNFs (`connection-point` section).

```
vnfd:vnfd-catalog:
    vnfd:
    -   id: cirros_vnfd
        name: cirros_vnf
        short-name: cirros_vnf
        description: Simple VNF example with a cirros
        vendor: OSM
        version: '1.0'

        # Place the logo as png in icons directory and provide the name
here
        logo: cirros-64.png

        # Management interface
        mgmt-interface:
            cp: eth0

        # Atleast one VDU need to be specified
        vdu:
        -   id: cirros_vnfd-VM
            name: cirros_vnfd-VM
            description: cirros_vnfd-VM
            count: 1

            # Flavour of the VM to be instantiated for the VDU
            # flavor below can fit into m1.micro
            vm-flavor:
                vcpu-count: 1
                memory-mb: 256
                storage-gb: 2

            # Image/checksum or image including the full path
            image: cirros034
            #checksum:

            interface:
            # Specify the external interfaces
            # There can be multiple interfaces defined
            -   name: eth0
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
                    bandwidth: '0'
                    vpci: 0000:00:0a.0
                external-connection-point-ref: eth0

        connection-point:
            -   name: eth0
                type: VPORT
```

Figure 12: Example VNFD composed of a single VDU.

In addition, configuration scripts could be provided for the lifecycle management by monitoring specific events. For instance, when VNFs are operating, measurements can be collected both from the VNF itself and from the hosting infrastructure. If the value of a meter is above or below a predetermined threshold during a period of time, specific actions can be performed, such as applying scaling policies, e.g., to scale up or down an instance of a VNFC whenever specific conditions are matched (especially load-dependent).

4.3 Modelling Languages

The *Organization for the Advancement of Structured Information Standards* (OASIS) created the *Topology and Orchestration Specification for Cloud Applications* (TOSCA) language [23] to describe the topology of cloud based web services, their components, and their relationships in a uniform and vendor-independent way. TOSCA is an information/data model based on YAML descriptor. The TOSCA standard includes specifications to describe processes that create or modify cloud services. However, since TOSCA is cloud-derived and focus on the definition of web services and compute requirements, rather than network functionalities. Therefore, it needs modifications in order to provide a complete description of a service in the NFV sense [24].

On the other hand, the *Yet Another Next Generation* (YANG) [25] is the data modelling language standard promoted by the Internet Engineering Task Force (IETF), also based on YAML. YANG is a very precise data model specification for modelling configuration and state change information. While it was originally designed for configuration of appliances, it can be reused for NFV orchestration.

TOSCA and YANG addresses different needs (orchestration vs. configuration), while sharing similarities in their functionalities to define data models. In the NFV context, they work usually together in a complementary way. However, as of today there is no generally accepted and completely converging data model to be used for NFV orchestration based either on TOSCA, YANG, or on a mix of both. To overcome these requirements, a parent data model for both languages is being developed. The ETSI GS NFV-IFA 011 [4] is the reference specification providing the requirements for the structure and the format of a VNF/NS Package to describe the VNF/NS properties and the associated resource requirements in an interoperable template.

In the following, the information model adopted by each of the open source NFV platforms introduced in Section 2.2 is presented.

4.4 Information Model from Open Source projects

4.4.1 OSM Information Model

For release 3.0, OSM broadly align with the ETSI NFV MANO phase 1 specification [26], defining a precise data and information model by making use of the YAML form of the YANG model for both VNFD and NSD. During the development of this release, the community assessed the ETSI NFV IFA 011 VNFD [4] and ETSI NFV IFA 014 NSD [27] Information Models.Such objects can also be automatically converted into *NETCONF* and *XML* objects, *protobufs,* and *GObject*.

To deploy a NS, the related NSD is processed by the SO component of OSM, composed of one or more VNFs, and eventually of VNFFGs and VVLs between the components. As explained before, the VNFD must define a VNF in terms of deployment and operational behaviour requirements. Additionally, VNFD shall specify the connections between Virtual Deployment Units (VDUs) using the internal Virtual Links (VLs). In OSM, a VDU can relate to both a VM and a Container. OSM uses an archive (e.g., tar.gz format) to pack NSD and VNFD together with related information. This archive consists of the NSD/VNFD, the initial configuration scripts and other auxiliary data.

A high-level analysis of OSM descriptors for NS, VNF, VDU, and VNFFG along with their respective data models is presented below. For additional references, please refer to the OSM wiki page [28], and to [29].

4.4.1.1 Network Service Descriptor (NSD) Data Model

The NSD (*nsd:nsd*) is the top-level construct used for designing the service chains, referencing all other descriptors that describe components that are part of that NS. The following elements are defined apart from the top-level network service: VNF information element, VL information element, VNFFG information element.

OSM key Name	OSM type	High level Description	Group of fields	Description of the fields
id	String	Unique description of this Network Service.		
name	String			
short-name	String			
vendor	String			
logo	String			
description	String			
version	String			
connection-point	List	List of the connection points of the NS.	• name • type	The *Name* and the *type* of the NS connection-point. Only value VPORT (virtual port) is supported.
vld	List	List of the Virtual Link Descriptors of the NS.	• id • name • short-name • vendor	ID is the [key] identifier for the VLD. Vendor holds the name of the provider of the VLD.
constituent-vnfd	List	List of the VNF Descriptors that are part of this Network Service.	• Member-vnfd-index	Holds the unique id of the VNFD.
			• vnfd-id-ref	Holds the path of the VNFD [id].
			• start-by-default	States if the VNFD is started as part of NS instantiation.
scaling-group-descriptor	List	List of scaling groups of the NS. The scaling groups consist of one or more VNFs. For each scaling group a different scaling action may be applied.	• Name	Name of the scaling-group .
			• scaling-policy	Contains Scale-in/out criteria for this network service.
			• vnfd-member (ui32)	Contains the list of the member VNF index and the number of each VNF's instances when a scaling action targets this group.
			• min-instance-count	Minimum and maximum instances of that are allowed.
			• max-instance-count	
			• scaling-config-action	Contains the scaling trigger type (pre/post - scale in/out) and a reference to the NSC primitive name.
placement-groups	List	List of placement groups. Each placement group defines the compute resource placement strategy in cloud environment.	• name • requirement • strategy • member-vnfd	Contains the resource placement strategy (COLLOCATION or ISOLATION) for a particular set of NS VNFDs.
ip-profiles-list	List	List of IP profiles that describe the IP characteristics for the Virtual Link.	• name • description • ip-profile-params	Contains IP parameters for each profile. (IPv, subnet, gateway, DNS list,DHCP,subnet-prefix-pool/ VIM-specific reference)
vnf-dependency	List	List of VNF dependencies.	• vnf-source-ref • vnf-depends-on-ref	Describes which VNF depends on the source VNF.
vnffgd	List	List of VNF forwarding graph descriptor.	• id • name • short-name • vendor • description • version • RSP • classifier	Contains fields that uniquely describe each FG. Each VNFFG model consists of a list of rendered service path (RSP) and a list of classifier components. RSP is an ordered list of references to SF. The SF reference exists in the form of references to connection-points of the constituent-VNFDs. Classifier defines a list of rules for the VNFFGD.
monitoring-	List	List of network parameters for the	• id	Contains fields that refer to

param		NS.	• name • monitoring-param-ui-data • monitoring-param-value • aggregation-type • vnfd-monitoring-param,	parameters for: • **UI-data** (eg. Histogram, bar, units/Mbps) • **values** (eg. INT,STRING) • **aggregation-types** (eg. AVG,SUM) • Reference to VNFD or VNFD monitoring parameter
input-parameter-xpath	List	List of xpath to parameters inside the NSD that can be customized during instantiation.	• xpath • label • default-value	Xpath that specifies the element in a descriptor, along with a descriptive string (label) and a default value for this input parameter.
parameter-pool	List	Pool of parameter values that must be pulled from during configuration.	• name • range	Name of the conf value pool and a Range of values from which to populate the pool. (Start/End values)
service-primitive	List	Network service level configuration primitives.	• name, • parameter • parameter-group • vnf-primitive-group • user-defined-script	Holds the name of the service primitrive, a grouping of parameters that are logically grouped in UI, a List of service parameters grouped by the VNF and a user defined script. Parameters of a Service-Primitive include: • name • data-type (string,int,boolean) • mandatory (boolean) • def_value
initial-config-primitive	List	Set of configuration primitives to be executed when the network service comes up.	• seq • name • user-defined-script • parameter	Holds the sequence number (seq) for the configuration primitive, its name, a user-defined script and a list of parameters with the subfields (name, value).
terminate-config-primitive	List	Set of configuration primitives to be executed before during termination of the network service.	• seq • name • user-defined-script • parameter	
cloud-config	List	Configures the list of users and public keys to be injected as part of network service instantiation.	• key-pair (name,key) • user (name, user-info, key-pair)	

Table 3: Descriptor details for a NS in OSM.

4.4.1.2 VNFD Data Model

The VNFD (vnfd:vnfd) is a deployment template that describes the attributes of a single VNF. The VNFM uses the VNFD during the process of VNF instantiation and during lifecycle management of a VNF instance. The information provided in the VNFD is also used by the NFVO to manage and orchestrate network services and virtualized resources on the NFVI. The VNF includes one or more VDUs (the VM that hosts the network function), virtual links providing internal (between its VNFC instances) and external (between the VNF and the outside network) connectivity, connection points, Platform resource requirements, special characteristics related to platform awareness and monitoring parameters. Each of these components (called nodes) has specific requirements, attributes, and capabilities that are defined in the VNF descriptor. The external connection points are used by the NSD to chain VNFs.

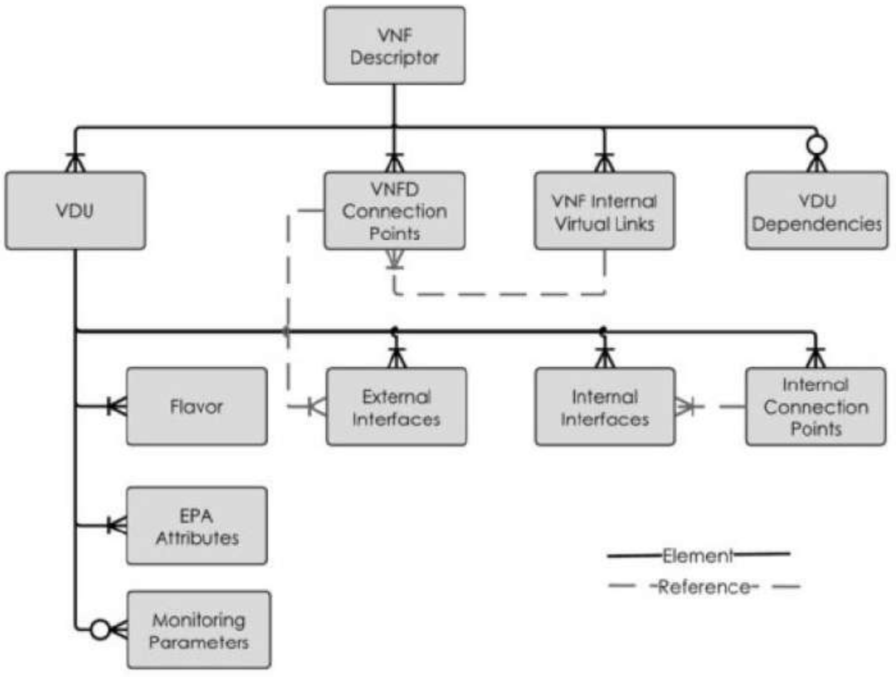


Figure 13: VNFD high-level object model.

VNFD Enhanced Platform Awareness (EPA): EPA provides an improvement in the performance of guest VMs by enabling the fine-grained matching of VNF requirements to the platform capabilities, before its instantiation.

Without optimization, VMs may be allocated from physical hosts anywhere within the same datacentre, and both the host and the physical links between these hosts can be oversubscribed. The CPU cores within a virtual machine might belong to different sockets on the physical host, leading to cache and memory access issues. This variability can lead to VNFs with completely different performance characteristics, even when they are placed in the same cloud infrastructure.

By enabling deterministic performance, OpenStack EPA attributes can increase the efficiency of the network function for high-touch tasks, such as packet forwarding and security. EPA attributes are discovered during the initial allocation of virtual machines from the Virtualized Infrastructure Manager (VIM).

This feature is particularly useful for VNF requiring specialized computation and/or networking performance. During the VNF instantiation process, the VNF request characteristics that are defined in the descriptor file and compared to the virtual machine capabilities in order to allocate workload placement across the corresponding VMs. This design improves NFV performance, supporting advanced placement. Among others, the data model can define the use of technologies such as hugepages, CPU pinning, NUMA awareness, PCI pass-through, Data Direct I/O (DDIO), Cache Monitoring/Allocation Technology. Additional **requirements to EPA due to the highly specialized networking workloads involved in NRG-5 are the data-plane acceleration techniques like DPDK and SR-IOV**.

This design improves NFV performance, supporting advanced placement such as:

- Placing high data rate workloads, such as load balancing and bearer plane forwarding, on VMs that support NUMA affinity, Launch high-bandwidth workload on a VM with DPDK capabilities, hugepage setup, CPU pinning, and PCI pass through or single root I/O virtualization (SR-IOV).
- Placing best-effort workloads, such as statistics gathering or log output, on “vanilla” VMs.
- Placing workloads that form part of the same network service (same service chain) in the same switching domain.

Specific characteristics related to EPA attributes and performance capabilities can be defined in the VNFD under the VDU configuration section.

VDU Data Model: A VDU defines the individual VNF components and capture information about VM image, VM flavour, and EPA attributes. VDUs are virtual machines that host the network function, such as the virtual machine specification, the computation properties (e.g., the RAM size, the disk size, the memory page size, the number of CPUs, the number of cores per CPU, and the number of threads per core), the storage requirements, the initiation and termination scripts (if any), the high availability redundancy model, and the limits for the scale in scale out policies.

Virtual Link Descriptor (VLD) Mode (used both in VNFD and NSD)**:** A virtual link descriptor (VLD) is a deployment template that describes the resource requirements needed for a link between VNFs, PNFs and endpoints of the NS, which could be met by various link options that are available in the NFVI.

Virtual Network Function Component (VNFC): Software that provides VNFs can be structured into software components, the implementation view of software architecture. These components can then be packaged into one or more images, the deployment view of software architecture. These software components are called VNFCs. VNFs are implemented with one or more VNFCs, where each VNFC instance generally maps 1:1 to a VM image or a container, as defined in the VDU.

OSM key Name	OSM type	High level Description	Group of fields	Description of the fields
id	String	Unique description of this Network Service.		
name	String			
short-name	String			
vendor	String			
logo	String			
description	String			
version	String			
vnf-configuration	container	Information about the VNF configuration for the management interface.	• config-method	Defines the configuration method for the VNF • script (BASH, EXPECT) • Juju charm to use with the VNF
			• config-access	IP, username and password to be used to configure this VNF
			• config-attributes	• config-priority • config-delay
			• service-primitive	Holds the name for the config primitive, a list of parameters and a user defined script.
			• initial-config-primitive	Initial set of configuration primitives.
			• config-template	Configuration template for each VNF.
mgmt-interface	container	Interface over which the VNF is managed.	• endpoint type (IP, vdu-id,cp) • port • dashboard-params (path, https, port)	
internal-vld	List	List of Internal Virtual Link Descriptors (VLD).	• id • name • short-name • vendor • description • version	que description of the internal virtual link
			• type	N - multipoint service that connects a set of VDUs
			• root-bandwidth	ELAN this is the aggregate bandwidth

			• leaf-bandwidth	ELAN this is the bandwidth of branches
			• internal-connection-point	of internal points in this VLD represented as id-refs
			• virtual-connection-points	of (available) virtual-connection points associated with this virtual Link. • name, id, short-name • type (VPORT) • port-security-enabled (if true RO passes the value to the VIM to filter traffic.) • static-ip-address • associated-cps (list of id-refs of cp associated with vcp)
			• provider-network	• physical-network • overlay-type (identifies the type of the overlay network - LOCAL, FLAT, VLAN, VXLAN, GRE) • segmentation-id
			• init-params	• vim-network-name • ip-profile-ref
ip-profiles	List	List of IP profiles that describe the IP characteristics for the Virtual Link.	• Name • description • <u>ip-profile-params</u>	Contains IP parameters for each profile. (IPv, subnet, gateway, DNS list,DHCP,subnet-prefix-pool/ VIM-specific reference)
connection-point	List	List for the **external** connection points	• name • id • short-name	Unique description of the cp.
			• type	VPORT
			• port-security-enabled	When set to True, the resource orchestrator passes the value to the VIM when the connection-point is created to filter traffic. (Openstuck VIM only)
			• static-ip-address	IPv4 or IPv6
vdu	List	List of virtual deployment units (VDUs).	• id • name • description • count • mgmt-vpci • vm-flavor • guest-epa • vswitch-epa • hypervision-epa • host-epa • alarm • image-properties • cloud-input • supplemental-boot-data • internal-connection-point • internal-interface • external-interface • volumes	VDUs are virtual machines that host the network function, such as: • Virtual machine specification • Computation properties (RAM size, disk size, memory page size, number of CPUs • number of cores per CPU, number of threads per core) • Storage requirements • Initiation and termination scripts • High availability redundancy model • Scale out/scale in limits
vdu-dependency	List	List of VDU dependencies, from which the orchestrator determines the order of startup for VDUs.	• vdu-source-ref • vdu-depends-on-ref	A reference to the VDU on which the source VDU depends on.
service-	enum	Type of node in the service		

function-chain		function chaining (SFC) architecture (UNAWARE, CLASSIFIER, SF, SFF)		
service-function-type	String	Type of service function. This field is emporarily set to string data type for ease of use.		
monitoring-param	List	List of network parameters for the VNF.	• http-endpoint • monitoring-param • monitoring-param-ui-data • monitoring-param-value	Contains fields that refer to parameters for: • **http-endpoint** (list of http endpoints to be used by monitoring params) • **monitoring-param/ui-data** (includes parameters for JSON queries, http-endpoint reference and UI monitoring parameters as described for NSD)
placement-groups	List	List of placement groups at VNF level. Thew group construct defines the compute resource placement strategy in cloud environment	• name • requirement • strategy • member-VDUs	Contains the resource placement strategy (COLLOCATION or ISOLATION) for a particular set of NS VNFDs. Also contains a list of VDUs that area part of this placement-group.

Table 4: Descriptor details for a VNF in OSM.

4.4.2 ONAP Information Model

ONAP follows a model driven approach based on TOSCA template YAML files for the topology of the service defined in [30]. While using a different modelling language than OSM, ONAP still describes the NSs and VNFs following the guidelines of ETSI NFV IFA011 [4]. For this reason, we omit the high level explanation of the data model and we simply provide the list for reference.

A notable difference is represented by the EPA requirements that natively permit to match platform exposing networking related capabilities such as SR-IOV, Hyper-Threading, and DPDK.

4.4.2.1 Network Service Descriptor (NSD) Data Model

ONAP attribute	Type	Description	Group of fields	Description of fields
id	Identifier	It Globally uniquely identifies an instance of the NSD.		
Designer	String	Designer of this NSD		
Version	String	Identifies the version of the NSD.		
Controllerinfo	String	Identifies controller(s) compatible with the NS described in this version of the NSD.		
Name	String	Identifies the name of the NSD		
Vnfdid	Reference to Vnfd	VNFD information element(s) of this NSD. It Globally uniquely identifies an instance of the NSD	See VNFD Data Model below	
PnfdId	Identifier	Identifier of the PNFD information element. It Globally	pnfid	Identifier of this Pnfd information element. It uniquely identifies the PNFD.

		uniquely identifies an instance of the NSD	provider	provider of this PNFD
			Version	Identifies the version of the PNFD
			Description	Description in human readable format
			Pnf_type	Type of PNF
			Request_reclassification	Determines whether VNF can request reclassification by the VNF forwarder
			Nsh_aware	Describes whether this VNF can process NSH headers.
			Management_address	Describes management port address of this PNF.
VirtualLinkDesc	NSVLD	Specifies the constituent VLDs.	Id	
			Vldid	
			Description	
			LayerProtocol	
			cpRole	
			sfcEncapsulation	
			Direction	
			interfaceName	
CpdId	Identifier	References the descriptor of VNF or PNF external connection points.	Id	Identifies this CPD information element within a NSD.
			VldId	References the VLD information element inside the NSD or the parent NSD.
			Description	Specifies human-readable information on the purpose of the connection point
			layerProtocol	Identifies a protocol that the connection points corresponding to the CPD support for connectivity purposes (e
			cpRole	Identifies the role of the connection points corresponding to the CPD in the context of the traffic flow patterns in the VNF, PNF or NS
			sfcEncapsulation	Defines the encapsulation of SFC
			Direction	Identity the direction of this CP including input, output, bidirectional
			InterfaceName	Specifies physical interface name of CP for PNF.
Vnffgd	VNFFGD	The VNFFGD information element describes a topology of connectivity of a NS and optionally forwarding rules applicable to the traffic over this topology.	Id	Identifier of this VNFFGD.
			Description	Description of this VNFFG.
			numberOf_Endpoints	Count of the external endpoints included in this VNFFG
			DependentVirtualLink	Identifies the reference to a NSVLD used in this Forwarding Graph
			ConnectionPoint	Reference to a set of connection points forming the VNFFG
			constituentVnfs	Reference to a set of VNFDs used in this VNFFG.
			ConstituentPnfs	Reference to a set of PNFD

				used in this VNFFG.
			NfpdId	Reference to the forwarding paths associated to the VNFFG.
PlanId	Plan	Specifies a list of life cycle management workflow.	• id • workflow	Workflow attribute specifies the workflow for this NS creation

Table 5: Descriptor details for a NS in ONAP.

4.4.2.2 Virtual Network function Descriptor (VNFD) Data Model

ONAP Attribute	Type	Description	Group of fields	Description of fields
id	Identifier	Identifies this vnfd information element within a NSD.		
vendor	String	Identifies the vendor of VNFD		
version	String	Identifies the version of the VNFD		
vnfmInfo	String	Identifies VNFM(s) compatible with the VNF described in this version of the VNFD.		
Vdu	VDU	Virtualisation Deployment Unit.	Id	Human readable descriptions.
			Name	
			Description	
			Intcpd	Describes network connectivity between a VNFC instance
			virtualComputeDesc • id • requestAdditional Capabilities • VirtualMemory • VirtualCpu	Describes CPU, Memory and acceleration requirements of the Virtualisation Container realizing this Vdu.
			VirtualStorageDesc • id • typeOfStorage • rdmaEnabled • SwImageDesc ○ id ○ name ○ version ○ checksum ○ containerFormat ○ diskFormat ○ minDisk ○ minRam ○ size ○ swImage ○ operatingSystem	Describes storage requirements for a VirtualStorage instance attached to the virtualisation container created from virtualComputeDesc defined for

			○ supportedVirtualisationEnvironment	
			bootOrder	The key indicates the boot index. The Value references a descriptor from which a valid boot device is created.
			MonitoringParameters(TBI)	Describes monitoring parameters of this VDU.
			configurableProperties • additionalVnfcConfigurable Property (keyValuePair)	Describes the configurable properties of all VNFC instances based on this VDU.
intVirtualLinkDesc	VirtualLinkDesc	Represents the type of network connectivity mandated by the VNF provider between two or more CPs which includes at least one internal CP.	id	Identifies this CPD information element within a NSD.
			description	Specifies human-readable information on the purpose of the connection point.
			virtualLinkDescFlavour	Describes a specific flavour of the VL with specific bitrate requirements
			connectivityType • layerProtocol (e.g. Ethernet,MPLS,IPv4) • flowPattern (e.g. Line, Tree, Mesh)	Describes the connectivity type of this VL.
			testAccess	Specifies test access facilities expected on the VL.
vnfExtCpd	vnfExtCpd	Describes network connectivity between VNF instances.		

Table 6: Descriptor details for a VNF in ONAP.

Additional VNFD information model elements in ONAP:

- AddressData
  - AddressData
  - L3AddressData
    - iPAddressAssignment
    - iPAddressType
    - floatingIpActivated
    - numberOfIpAddress
- RequestedAdditionalCapabilityData
  - requestedAdditionalCapabilityName
  - supportMandatory
  - numaEnabled
  - preferredRequestedAdditionalCapabilityVersion
  - targetPerformance Parameters
- VirtualMemoryData
  - virtualMemSize
  - virtualMemOversubscriptionPolicy
  - numaEnabled

- VirtualCpuData
    - cpuArchitecture
    - numVirtualCpu
    - virtualCpuClock
    - virtualCpuOversubscriptionPolicy
    - virtualCpuPinning
- VirtualCpuPinningData
    - cpuPinningPolicy
    - cpuPinningMap

4.4.3 OpenStack Tacker Information Model

Tacker is an OpenStack service for NFV Orchestration with a general purpose VNF Manager to deploy and operate VNFs and NS on an NFV Platform. It is based on ETSI MANO Architectural Framework. Descriptors in OpenStack Tacker are represented using TOSCA modelling language. The following information is based on Pike series release notes (0.8.0) of OpenStack [31].

4.4.3.1 Network Service Descriptor (NSD) Data Model

NSD in Tacker can be used for creating multiple (related) VNFs in one shot using a single TOSCA template. However, VLs / neutron networks using NSD (to support inter-VNF private VL) and VNFFGD support in NSD is limited and expected to be fully realized in the near future. In the current release, NSD support is introduced in order to allow Tacker to provide end-to-end TOSCA-based network services.

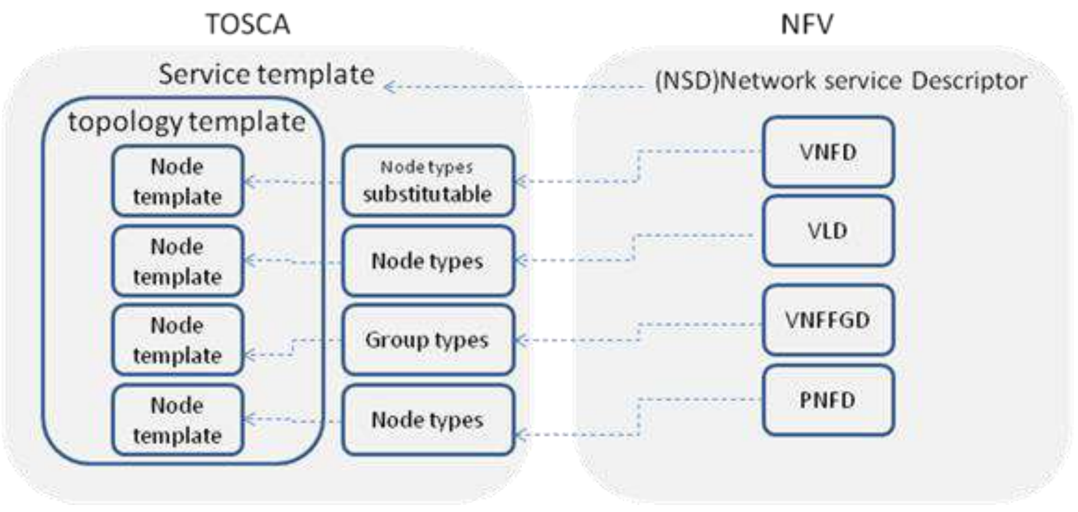


Figure 14: General Mapping between TOSCA and NFV.

As shown in Figure 14, a TOSCA-defined NSD can include the VNFD, the VNFFGD, the VLD and the PNFD (not supported in the current version).

Tacker attribute	Type	Description	Group of fields	Description of fields
Id	String	Unique Identification of the network service		
Name	String	Human readable name		
Description	String	Describes the Network Service		
Metadata	List	List of import statements for importing other definitions files	• id • vendor • version	Unique identifications of the ID, the vendor and the version of this

				network service.
Imports	String	List of import statements for importing other definitions files	VNFD's (list)	
topology_templates		Describes the topology of the NS	• inputs (flavor ID) • substitution_mappings ○ node_type ○ requirements ○ capabilities ▪ Forwader	Based on the **requirements** of the imported descriptors, NSD template defines node_templates (eg. VL) which substituting the required ones that are specified by the imported VNFD's Forwarder, refers to the forwarding path declared by the VNFFGD

Table 7: Descriptor details for a NS in OpenStack Tacker.

4.4.3.2 VNF Forwarding Graph (VNFFG) Model

The VNFFG template is based on TOSCA standards and is written in YAML. It is on-boarded in a VNFFG catalog.

In a full **Network Services Descriptor** (NSD), NSD would include information about each VNFFGD as well. However, until that implementation, VNFFGD is described in a separate template. The VNFFGD template has the following fields:

- *Node_template* (which is a child of topology_template in NSD): For Tacker purposes, it includes only the Forwarding Path subsection which describes the properties and path of a Forwarding Path.
  - **Forwarding Path:**
    - **id** (path ID). This path ID will be used in future implementations of the Network Service Header (NSH) to identify paths via the Service Path Identifier (SPI) attribute.
    - **policy** (traffic match policy to flow through the path). The only currently supported type is ACL (access-list).
    - **path** (chain of VNFs/Connection Points). Path defines an ordered list of nodes to traverse in a Forwarding Path. Each node is really a logical port, which is defined in the path as a Connection Point (CP) belonging to a specific VNFD.

- *Groups***:** In Tacker and TOSCA, the VNFFG itself is described in this section. There may only be a single VNFFG described in each VNFFGD under this section.
  - **VNFFG:** VNFFG maps the Forwarding Path to other node types defined in the properties section. Properties of the VNFFG include:
    - **number_of_endpoints**: Number of CPs included in this VNFFG;
    - **dependent_virtual_link**: The Virtual Link Descriptors (VLD) that connect each VNF/CP in this Forwarding Graph.
    - **connection_point**: List of Connection Points defined in the Forwarding Path.
    - **contituent_vnfs**: List of VNFD names used in this Forwarding Graph (also defined in Forwarding Path).

4.4.3.3 Virtual Network Function Descriptor (VNFD) Data Model

A valid VNF in Tacker must include at least a VDU, a connection point and a VL. Each component is referred to as a node and can have certain type, capabilities, properties, attributes and requirements. These components are described under the **node_templates** inside the VNFD template.

Tacker attribute	Type	Description	Group of fields	Description of fields
tosca_definitions_version	String	This defines the TOSCA definition version on which the template is based.		
tosca_default_namespace	String	It mentions default namespace which includes schema, types version etc. (optional)		
description	String	A short description about the VNFD template.		
metadata	String	A name to be given to the VNFD template.	• id • vendor • version	Unique identifications of the ID, the vendor and the version of this network service.
topology_template		Describes the topology of the VNF under node_template field.	• node_template	Describes node types of a VNF. and is based on OASIS -TOSCA template.
node_template	List	Node_template field is a child of topology_template. It describes node_types of a VNF. •tosca.nodes.nfv.VDU.Tacker •tosca.nodes.nfv.CP.Tacker •tosca.nodes.nfv.VL	• vdu • cp • vl	Describes the properties and the capabilities of each component
VDU	Node type	Virtual Deployment Unit Descriptor template. It defines the properties and the capabilitie of the VDU.	• properties ○ image (type, file) ○ flavor ○ availability_zone ○ disk_size ○ mem_size ○ num_cpus ○ config_drive • config • capabilities ○ mem_page_size ○ cpu_allocation ▪ cpu_affinity ▪ thread_count ▪ socket count ▪ core_count ▪ thread_count ○ numa_node_count ○ numa_nodes (id, vcpus, memsize) • monitoring_policy (name) ○ monitoring_delay ○ count ○ interval ○ timeout ○ actions (failure:respawn,terminate,log) ○ retry (number) ○ port	**Properties** describe the image type, the vm flavor and the availability zone (e.g nova) **Config** Under config section a VDU can be configured as a specific Network Function. (eg. Firewall) **Capabilities** define, VDU_compute properties. **Monitoring_policy** specifies the monitoring configuration options for the VNF. Name attribute defines (ping, noop, http-ping).
CP	Node type	Connection Point Descriptor Template Connection point is used to connect the internal virtual link or outside virtual link.	• properties ○ anti_spoofing_protection ○ management (true,false) ○ order (of cp in VDU) ○ type (sriov, vnic) ○ security_groups ○ mac_address ○ ip_address • requirements	A CP always requires a virtual link and a virtual binding associated with it.**VirtualLink** states the VL node to connect to. **VirtualBinding** states the VDU

			○ Name ○ virtualLink ○ virtualbinding	node to connect to.
VL	Node type	Virtual Link Descriptor Template Virtual link provides connectivity between VDUs. It represents the logical virtual link entity.	• properties ○ Name ○ Vendor ○ Network_name	Name is a description, vendor states the vendor generating this VL and Network_name corresponds to the name of the network to which VL is to be attached.

Table 8: Descriptor details for a VNF in OpenStack Tacker.

4.4.3.4 Node types

A VNF includes **VDUs**, **connection points** and **virtual links**. Hence a valid VNFD must have these 3 components. Each component is referred as a node and can have certain type, capabilities, properties, attributes and requirements. These components are described under **node_templates** in the VNFD template. **node_templates** is a child of **topology_template.**

4.4.3.5 VNF Scaling

VNF resources in terms of CPU core and memory are hardcoded in the VNFD template through image flavour settings, causing to provision the VNF either for its typical usage or for its maximum usage. The former leads to service disruption when load exceeds the provisioned capacity. And the latter leads to underutilized resources and waste during normal system load. Tacker provides a way to scale the number of VNFs on demand either manually or automatically. Scaling properties include:

- min_instances: Minimum number of instances to scale in.
- max_instances: Maximum number of instances to scale out.
- default_instances: Initial number of instances.
- cooldown: Wait time (in seconds) between consecutive scaling
- increment: The number defines the increment of scaling action.
- targets: List of scaling nodes.

4.4.4 Sonata Information Model

SONATA is developing a NFV framework that provides a programming model and development toolchain for virtualized services, fully integrated with a DevOps-enabled service platform and orchestration system. SONATA is an ETSI based MANO open source framework.

Main differences of SONATA project with respect to ETSI framework:

- SDK: That offers an extensive toolset for validation, monitoring, profiling and packaging of NFV-based services. Also offers an emulation environment to deploy services, on an emulated infrastructure topology, a graphical web-based interface, analysis tools to process monitored service data, an example services repository for trying out **SONATA** environment and tools to support FSM/SSM development.
- VNFD that is based on the T-NOVA flavour of the ETSI VNFD, but adapted and extended to meet the SONATA specific needs.
- Message broker between Service Specific Manager (SSM) and Function Specific Manager (FSM).
- Multiple SSM/FSM plugins to define precisely how to handle specific lifecycle events.

Through the SDK and SSM/FSMs, SONATA Service Platform provides modularity and flexibility.

SONATA descriptors are specified in a machine-readable format using JSON schema and can be found in the SONATA GitHub repositories [17]. A high level analysis of the **SONATA** NS and VNF Descriptors is presented in the table below.

4.4.4.1 Network Service Descriptor (NSD) Data Model

SONATA Attribute	SONATA Type	High level Description	Group of fields	Description of the fields
name	String	Unique description of this Network Service.		
description	String			
version	String			
connection-points	Array	A list of all external connection points of the NS.	• name • type • virtual_link_reference	The *Name* and the *type* of the NS connection-point..
virtual_links	Array	List of the Virtual Link Descriptors of the NS.	• id • name • short-name • vendor	ID is the [key] identifier for the VLD. Vendor holds the name of the provider of the VLD.
network_functions	Array	List of the VNF Descriptors that are part of this Network Service.	• member-vnfd-index	Holds the unique id of the VNFD.
			• vnfd-id-ref	Holds the path of the VNFD [id].
			• start-by-default	States if the VNFD is started as part of NS instantiation.
			• vnf_vendor • vnf_version • vnf_name • description (optional)	As part of the primary key, the **name, vendor** and **version**, identify the VNFD.
forwarding_graphs	Array	List of VNF forwarding graph descriptors.	• id • name • short-name • vendor • description • version • RSP • classifier	Contains fields that uniquely describe each FG. Each VNFFG model consists of a list of rendered service path (RSP) and a list of classifier components. RSP is an ordered list of references to SF. The SF reference exists in the form of references to connection-points of the constituent-VNFDs. Classifier defines a list of rules for the VNFFGD.
monitoring-parameters	Array	List of network parameters for the NS.	• id • name • monitoring-param-ui-data • monitoring-param-value • aggregation-type • vnfd-monitoring-param,	Contains fields that refer to parameters for: • **UI-data** (eg. Histogram, bar, units/Mbps) • **values** (eg. INT,STRING) • **aggregation-types** (eg. AVG,SUM) • Reference to VNFD or VNFD monitoring parameter

Table 9: Descriptor details for a NS in Sonata.

4.4.4.2 Virtual Network Function Descriptor (VNFD) Data Model

SONATA attribute	SONATA type	High level Description	Group of fields	Description of the fields
name	String	Unique description of this Network Service		
vendor	String			
author	String			

description	String			
version	String			
virtual_links	Array	List of Internal Virtual Link Descriptors (VLD).	• id	VNFD-unique identi_er for the internal VLD.
			• type	The type of the virtual link, such as point to point multipoint etc.
			• internal-points-reference	Reference to one or more internal connection-points.
			• virtual-connection-points	List of (available) virtual-connection points associated with this virtual Link. • name, id, short-name • type (VPORT) • port-security-enabled (if true RO passes the value to the VIM to filter traffic.) • static-ip-address • associated-cps (list of id-refs of cp associated with vcp)
			init-params	• vim-network-name • ip-profile-ref
connection-points	List	List for the **external** connection points	• id	Unique description of the cp.
			• type	The type of the connection point.
virtual_deployment_units	List	List of virtual deployment units (VDUs).	• id • resource_requirements • vm_image • vm_image_md5 • connection_points • scale_in_out	**ID**: unique identifier of the VDU **resource_requirements:** Additional information that identifies the flavor of the VM instance. **Vm_image**: image name **vm_image_md5:** The MD5 checksum of the VM image **connection_points:** List of connection points. **scale_in_out:** Specifies the minimum and maximum number of VDU instances.
monitoring-rules	List	List of network parameters for the VNF.	name	The name of the monitoring rule.
			description (optional)	An arbitrary description of this monitoring rule.
			duration	The duration the condition has to be met before an event is _red.
			duration_unit (optional)	The unit of the duration, such as seconds, minutes, and hours.
			condition	The condition, i.e. a Boolean expression, that must be met to _re the event.
			notification	A list of notifications that are fired when the condition is met.
VNF lifecycle events	Array	VNF workows for speci_c lifecycle events such as start, stop, scale out, update, etc.	vnf_container	The VNF container that is associated with the lifecycle event.
			events	The actual event such as start, stop, scale out, update, etc.
deployment flavours	Array	The avours of the VNF that can be deployed.	flavour key	A VNF-unique id of the deployment flavour which can be used for references.
			vdu reference	A reference to the VDU, vdu:id.

Table 10: Descriptor details for a VNF in Sonata.

4.4.5 OPNFV Information Model

As mentioned in section 2, OPNFV enables choices for end users through multiple options, and it is not OPNFV's role to recommend one technology over another [18]. As a testing and integration project, currently hosting more than 40 upstream projects, OPNFV brings together upstream components across compute, storage and network virtualization in order to create an end-to-end platform. The MANO frameworks that OPNFV supports are OpenStack Tacker, ONAP and OpenBaton. All the aforementioned frameworks provide a comprehensive implementation of the ETSI NFV Management and Orchestration (MANO) specification.

OPNFV Data Model (VNF and NS Descriptors) depends on the particular MANO framework that is being selected and integrated by an OPNFV-project. Given the importance of standardizing the VNF onboarding process, the MANO working group, along with the Models project (OPNFV project) is working on standardizing VNF onboarding for OPNFV. Models project's focus on promoting information/data model support in open source and the alignment with standards has been up streamed to ONAP (which Information/Data model has been presented in Section 4.4.2).

4.5 Comparison of the NFV Platforms

All of the candidates are open source frameworks, which is crucial for long-term viability and ensures that will always be on the cutting-edge of NFV technology.

It must be also noted that most of the aforementioned frameworks are evolving rapidly and their development is expected to keep progressing in the future, either by enhancing their functionality or by covering a broader spectrum of the NFV architecture. Although, the platforms' announced additions and upgrades that could be beneficial to the scope of the NRG-5 along with the development time plan for each platform was taken into consideration for the final decision, given the time constrains of the project (NRG-5) and the uncertainty of the effectiveness and the delivery date of the upcoming updates a guarded approach was adopted.

Lastly, it must be mentioned that there is an expected difficulty on verifying documentation and specification characteristics without deploying and use-case testing these platforms.

4.5.1 Assessment of candidate solutions

A high level comparison of the evaluated platforms is presented in Table 11.

Comparison Criteria	OSM	Tacker	ONAP	SONATA	OPNFV
Sustainability	Supported by a large operator-led, active and open community. It is hosted by ETSI, thus it's closely aligned with ETSI NFV. It is also supported by telcos such as Telefónica, British Telecom, Telekom Austria Group, Korea Telecom, and Telenor, and vendors such as Intel, Mirantis, RIFT.io, Brocade, Dell, RADware, etc.	Supported by the OpenStack community and backed by major telecom corporations.	Supported by the Linux Foundation and backed by many major companies.	Supported by 5G PPP partners. Involvement of external contributors and further sustainability might be challenging after the project ends (2018).	Supported by the Linux Foundation and Industry.
Support NFVI environment	OpenVIM, OpenStack, VMware, AWS	OpenStack	OpenStack, AWS. However, Multi-cloud/Multi-VIM project addresses the aspect of multi-VIM support	Openstack	OpenStack, Kubernetes

Technical documentation and support (installation & maintenance)	Yes	Yes	Yes	Yes	Yes
Integration with SDK tools	No	No	Yes, VNF-SDK project.	Yes, at the core of the platform	No
Background expertise	No, However a demo-deployment of the platform was successfully made.	No	No	Yes, members of the NRG-5 consortium are also members of SONATA.	No
Support hypervisor and container-based technologies	LXD			Yes. (Containers are used for SONATA Service Platform components while VNFs are deployed within OpenStack VMs	KVM, LXD
Support VNF Chaining	Yes	Yes	Yes	Yes	Yes, however depends on the VNFM adopted for a project. In previous releases Tacker VNFM was used for managing VNFs lifecycle. Latest OPNFV release uses ONAP for VNFM.
NSD support	Yes	NSD is not comprehensively supported by Tacker.	Yes	Yes	Yes, but depending on the orchestrator that is integrated for the particular project.
Support VNFFGs update during runtime	Yes (experimental need verification)	Yes, Partially implemented.	No	Not yet, it's part of future updates	Not yet, it's part of future updates
VNF Scaling (out/in)	Yes, in OSM (rel 2) was an experimental feature, but in OSM (rel 3) is supported. However the scaling action is triggered manually.	Yes, manual or automatic. Auto-scaling feature is supported only with alarm monitors.	No, it isn't supported in the current release (rel 1) There are plans to support it in the second release (rel 2).	Yes, via the use of SSM/FDM plugins.	No, Depends on the underlying components that are selected. Yardstick and Bottlenecks projects
Support VNF chaining	Yes	Yes (not by itself)	Yes (not by itself)	Yes	Yes
VNF Monitoring	Yes	Yes, Using Tacker Mistral, an integral part of Tacker MANO system.	Yes	Yes, via the use of SSM/FDM plugins	Yes
VNF-Healing	Yes	Auto-healing for individual VNFs is supported by Tacker using (ping, http-ping). However as reported in the documentation it works with serious issues after a VNF respaw process. Tacker provides auto-healing for VNFFG in single site, multi-sites support will be taken	Yes	Yes, via the use of SSM/FDM plugins	Yes, based on policy.

		into account in future release.			
Support Multiple-PoP (Points of Presence)	No, However it supports a Multi-PoP NFVI/VIM Emulation Platform	Yes	Yes	Yes	Yes, Multisite Virtualized Infrastructure (Multisite) Project, currently covers this aspect.
Service oriented 5G network Slicing	Arctos Labs, Netrounds and RIFT.io have developed ETSI OSM PoC #1 with Telenor on Intel Architecture to showcase a solution that embodies management, orchestration, and testing of 5G network slices in a virtualized environment.			Yes, supports 5G slicing	

Table 11: Comparison of capabilities of surveyed NFV platforms against NRG-5 requirements.

After a careful review of the available open source solutions and their capabilities**, the consortium selected Open Source MANO (OSM) to represent the basic NFV platform to extend for NRG-5**. In the context of the specific requirements, some basic enhancements and additions over the OSM basic information model are required.

4.6 NRG5 Extensions to the Information Model

In the following tables, we overview the initial extensions to the OSM data model that will be supported by the target release of the NRG-5 platform. As a first step, the information model should be able to express the requirements that will be mirrored by the development work performed in the context of Tasks 3.3 and 3.4 in the following project periods.

In order to guarantee the **end-to-end latency** constraints for the network service, which includes all the elements in the end-to-end service (VNFs and infrastructure components), we need to define its value and the inbound and outbound points where to measure such latency. In a similar way, for capacity-sensitive application **bandwidth** should be guaranteed (eventually through slicing and reservation). While this metric is dependent on the performances of the service chain, to reduce latency further it is important to define explicitly the **placement** of the deployment.

Availability refers to the end-to-end service availability. It is needed to define the service chain based on which the availability requirements can be decomposed into requirements applicable to individual VNFs and their interconnections. These characteristics should be maintained in VM migration, failovers and switchover, scale in/out, etc. scenarios. The ETSI-NFV-REL defined three **service availability levels** and **failure recovery time values** that are mirrored in the NRG-5 extensions. They are based on the relevant ITU-T recommendations and reflect the service types and the customer agreements a network operator should consider.

4.6.1 Network Service Descriptor (NSD) Data Model Additions

NRG-5 extension key name	Type	High level Description	Group of fields	Description of the fields
e2e-latency	Array	The maximum expected e2e latency for the network chain	• latency • name1 • name2	The maximum latency in ms allowed between traffic entering from *Name1* and exiting to *Name2* the NS connection-points. Only value VPORT (virtual port) is supported.
service_placement	String	The explicit placement of the whole chain over a specific location		
service_availability_level	String	The Service Availability Level defined as expressed in ETSI GS NFV-REL 001		
service_recovery_level	String	The Service Recovery Level defined as expressed in ETSI GS NFV-REL 001		
e2e-bandwidth	Array	The guaranteed bandwidth for the service.	• bandwidth • connection point	Guaranteed *bandwidth* in Mbps that must be guaranteed at the *connection point*

Table 12: Initial set of NRG-5 proposed extensions for NSD.

4.6.2 Virtual Network Function Descriptor (VNFD) Data Model Additions

NRG-5 extension key name	Type	High level Description	Group of fields	Description of the fields
vnf-latency-telemetry	Array	The maximum expected e2e latency for the network function	• latency • connection point 1 • connection point 2	The maximum *latency* in ms allowed between traffic entering from *connection point 1* and exiting to *connection point 2,* the VNF connection-points.
function_placement	String	The explicit placement of the network function over a specific location		
redundancy_strategy	String	Redundancy strategy for the VNF. It could take the following values: "1+1 Redundancy"; "1:1 Redundancy" "Best Effort"		

Table 13: Initial set of NRG-5 proposed extensions for VNFD.

5 Interfaces and APIs

This section presents a first definition of the high-level methods (and related arguments) to be supported by the open APIs exposed by the components of the NRG-5 platform. The current section overviews the interactions taking place between the catalogue and the service developers, between the catalog and the MANO, and between the MANO and the OSS/BSS.

5.1 The Role of Catalogues

The catalogues are used to store the information needed to instantiate, manage and run the NRG-5 NSs. Following the ETSI NFV reference architecture depicted in Figure 1, each NRG-5 catalogue communicates with the NFV MANO, and specifically with the NFVO, via the reference point *Se-Ma.* This reference point is used by the MANO to retrieve information regarding the VNF deployment template, the VNFFG, the service-related information and the NFV infrastructure information model to perform both management and orchestration.

Multiple catalogues can interact at the same time with the NRG-5 platform, each one covering a different function and being adapted to the specific requirements of the UC and the stakeholder involved in its deployment. These different aspects could span from the actual information stored in the catalogue, to the different security and access policies implemented in the catalogue. The NRG-5 platform needs to hold several catalogues to run the services for specific stakeholders. Moreover, private catalogues must be available to service developers to deploy and test innovative NSs.

For instance, the following four types of catalogues/repositories are defined in the ETSI NFV MANO document [26]:

- **Network Service Catalogue** storing the definitions for NS and supporting the following templates: NSD (referencing the VNFs, VLs, and the VNFFGs), the VNFFGD (referencing VLs), and the VLDs.
- **Virtual Network Function Catalogue** storing the definitions for VNFs and supporting the following templates: VNFD, software images, manifests and meta-data.
- **Virtual Network Function Instances Repository** storing information of running VNF/NS instances. Records of the instances are constantly updated via the lifecycle management operations.
- **Virtual Network Function Instances Resource Repository** holding information on the available/reserved/allocated NFVI resources as abstracted by each VIM operating in each NFVI-PoP domain; thus supporting NFVO Resource Orchestration role.

Moreover, the SONATA project has proposed to add a catalogue/repository storing the descriptors and monitoring information for the SONATA specific *SSM/FSM* lifecycle managers [32].

It has to be noted that, regardless of the function covered and the specific configuration required by each deployed catalogue/repository, all of the different instances share the same technical basis and have a common functioning through a unified interface model.

5.2 Interactions between the Catalogue and the NRG-5 Platform

To support the lifecycle management and operation of NRG-5 services, the catalogue in the platform shall provide the capabilities to store the NSD, VNFD, their related assets (disk images, configuration files), and all sort of descriptors. These resources are required to select, develop and instantiate NRG-5 services. The following CRUD (Create Retrieve Update Delete) oriented interactions are considered:

- **Submit (create) a package:** permitting to a service developer to push a package into the catalogue. This step is needed before being able of utilize the NS.
- **Retrieve a package:** the package files can be downloaded from the catalogue.
- **Retrieve a descriptor:** meta-data concerning can be downloaded from the catalogue.
- **Delete a package:** stored packages can be removed from the catalogue. Special care needs to be taken for running and instantiated components.
- **Update a package:** the package files can be updated with new versions.

An additional **search** interaction is considered in order to discover the items stored at the catalogue and matching certain search parameters.

5.3 Interfaces to the Developers

The table below enumerates the interfaces that the NRG-5 platform offers to developers to interact with the NRG-5 catalog. The interfaces are used to retrieve the virtual Network NRG5 Function (vNNF) and NS descriptors, but also their related VNF images and configuration artifacts. These interfaces offered to the developers provide a first set of methods with which it is possible to retrieve the lists of descriptor, list of images, cancel and update already available resources.

Operation	Args	Description
get_nsds	-	List of all NS descriptor
post_nsd	-	Store a NS descriptor
del_nsd	id	Erase an NS descriptor
update_nsd	id	Update and NSD using version and vendor
get_status_nsd	id	List all the status of NS
get_vnfsds	-	List of all the available VNFDs
post_vnfsd	-	Store a VNF descriptor
del_vnfds	id	Delete a VNFD
update_vnfds	id	Update a VNFD using the naming trio, i.e., name, vendor & version
get_status_vnfds	id	List all VNFDs with a given status
get_packages	-	List all the available package descriptors
post_packages	id	Store a package in the Catalogues
del_packages	id	Cancel a packages
update_packages_descriptor	id	Update a package descriptor
get_status_nsd		List all packages with status 'active' or 'inactive'
onboard_vnnf	id	Onboards a vNNF
onboard_ns	id	Onboards a NS
get_vnnf_onboarding_status	--	Provides the status for the vNNF onboarding operation
get_ns_onboarding_status	-	Provides the status for the NS onboarding operation
list_vnnfs	-	Provides a list of all the onboarded vNNFs along with a brief description for each one
list_nss	-	Provides a list of all the onboarded NSs along with a brief description for each one
get_vnnf_info	id	Provides all the information on the onboarded vNNF
get_ns_info	id	Provides all the information on the onboarded NS
decommission_vnnf	id	Retire a vNNF
decommission_ns	id	Retire a NS

Table 14: Interfaces exposed to service developers by the NRG-5 catalog.

5.4 Interfaces to the MANO

NFV MANO is broken up into three functional blocks:

- **NFV Orchestrator**: Responsible for on-boarding of new NSs and VNFs packages; NS lifecycle management; global resource management; validation and authorization of network functions virtualization infrastructure (NFVI) resource requests.
- **VNF Manager**: Oversees lifecycle management of VNF instances; coordination and adaptation role for configuration and event reporting between NFVI and E/NMS.
- **Virtualized Infrastructure Manager** (VIM): Controls and manages the NFVI compute, storage, and network resources.

In the NRG5 approach the NFVO interacts with the packages in the catalog via the *Se-Ma* reference point and it is effective during the deployment or instantiation. The orchestrator requests the NSD and the related VNFD from the store, as a first step to gather all resources for the NS instantiation.

5.5 Orchestrator to the OSS layer

This section includes the high level definition of the operations offered by each API exposed by the orchestrator to the OSS layer.

The Northbound interface is based on REST and it allows performing actions over the following entities:

- **Tenant**: Intended to create groups of resources.
- **Datacentres**: Is a VIM that contains a specific pool of resources.
- **VNFs**: Software-based network function composed of one or several VMs that can be deployed on an NFV datacentre.
- **Scenarios**: topologies of VNFs and their interconnections.
- **Instances**: Each one of the VNFs deployed in a datacentre.

Operation	Args	Description
get_network_topology	-	Provides the topology of the network
get_deployed_vnnfs	Tenant id	Provides the running vNNFs filtered based on tenant
get_physical_nodes	-	Provides the list of active physical nodes in the NFVI
get_deployed_vnnfs	-	Provides the running vNNFs

Table 15: Interfaces exposed to the MANO platform by the NRG-5 catalog.

6 Conclusion

This deliverable presented the initial definition of the NRG-5 automated NS/VNF deployment strategy, as the result of the activities carried out in Task 3.1. This document proposes a set of required extension to open source NFV platforms, a foundational analysis on the information model to represent energy-related telecommunication services, together with several technical requirements extracted from the proposed use cases of the project. While this deliverable provided an early description of the functional blocks of the NRG-5 platform together with the related APIs, and of the information model to be adopted, the actual implementation will be described in future deliverables, which will also benefit from the work performed by the other tasks of the project.

In order to gain efficiency, before delving into the definition of such platform, we needed to perform a thorough analysis of the existing NFV platforms and of their related information models. Indeed, the many open source projects dedicated to NFV deployments testify to the interest brought by the industry on the subject; however, it must not be forgotten that researches and experimentations are still ongoing today. Being able to monitor and follow these advancements during the project timeline represents a very important challenge.

Section 2 provides a brief overview of the NFV architecture as specified by ETSI ISG NFV working group representing the basis of any NFV deployment. While the basic architecture remains the same and is inspired by such specification, as of today, different projects exist. For this reason, we overviewed the most known NFV platforms to date, highlighting their particularities and shortcomings.

Requirements from the use cases have been extracted, paying special attention to those that cannot be specified by the information models available today. By taking into account the specific use cases supported by NRG-5, in Section 3 we identified several key requirements related to 5G and the energy domain that have to be expressed and guaranteed during service execution, namely the end-to-end latency, the availability, the reliability, and the bandwidth KPIs.

The information model is the way to describe both the deployment and the runtime properties of a NS. The Section 4 introduced the need for a coherent information model and overviewed the different models proposed by the NFV platforms under analysis. It provided also an initial set of extensions in order to meet the requirements arising from the use cases.

Finally, in Section 5, a first draft of the required programming interfaces (APIs) between the NRG-5 platform and the catalogue and between the catalogue and the service developers has been presented.

7 Acronyms

5G PPP – Fifth Generation Public Private Partnership

AGC - Automatic Generation Control

API – Application Programming Interface

BSS - Business Support System

CRUD – Create Retrieve Update Delete

DER – Distributed Energy Resource

DES – Distributed Energy Storage

DPDK – Data Plane Development Kit

ECOMP – Enhanced Control, Orchestration, Management and Policy

ESCO – Energy Service Company

ETSI – European Telecom Standards Institute

EV – Electrical Vehicle

FSM – Function-Specific Manager

HOT - Heat Orchestration Template

KPI - Key Performance Indicator

MANO – Management & Network Orchestration

MEC – Mobile Edge Computing

NETCONF – Network Configuration Protocol

NF – Network Function

NFV - Network Function Virtualization

NFVI – Network Function Virtualization Infrastructure

NFVO - Network Function Virtualization Orchestrator

NS - Network Service

NSD – Network Service Descriptor

OASIS – Advancement of Structured Information Standards

ODL – Open Day Light

ONAP - Open Network Automation Platform

ONOS – Open Network Operating System

OPNFV - Open Platform for NFV

OSM - Open Source MANO

OSS - Operator Support System

PNFD - Physical Network Function

QoE – Quality of Experience

QoS – Quality of Service

REST - Representational State Transfer

RO – Resource Orchestrator

SDK - Software Development Kit

SDN - Software Defined Networking

SO – Service Orchestrator

SR-IOV - Single-root input/output virtualization

TOSCA - Topology and Orchestration Specification for Cloud Applications

TSO – Transmission System Operator

UC – Use Case

VCA – VNF Configuration and Abstraction

VDU – Virtual Deployment Unit

VIM – Virtualized Infrastructure Manager

VLD - Virtual Link Descriptor

VM – Virtual Machine

VNF - Virtual Network Function

VNFD - Virtual Network Function Descriptor

VNFFGD - Virtual Network Function Forwarding Graph Descriptor

VNFM - Virtual Network Function Manager

VNNF - Virtual Network NRG5 Function

YAML – Yet Another Markup Language

YANG - Yet Another Next Generation

Reference

[1] NRG-5 Consortium, "D1.1, Use Case scenario analysis and 5G requirements," 2017.

[2] NRG-5 Consortium, «D1.2, “Reference Architecture & Functional Decomposition",» 2018.

[3] ETSI GS NFV 002, “NFV Reference Architecture v1.1.1,” 2013.

[4] ETSI GS NFV-IFA 011, “Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; VNF Packaging Specification V2.3.1,” 2017.

[5] “SONATA NFV: Agile Service Development and Orchestration in 5G Virtualized Networks,” [Online]. Available: http://www.sonata-nfv.eu/.

[6] “5G TANGO - 5G Development and Validation Platform for global Industry-specific Network Services and Apps,” [Online]. Available: http://5gtango.eu/.

[7] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turk et R. Boutaba, «Network Function Virtualization: State-of-the-Art and Research Challenges,» *IEEE Communications Surveys and Tutorials,* vol. 18, n° %11, pp. 236--262, 2016.

[8] "Main Page - KVM," [Online]. Available: http://www.linux-kvm.org/. [Accessed 17 1 2018].

[9] "Tacker - OpenStack NFV Orchestration," [Online]. Available: https://wiki.openstack.org/wiki/Tacker.

[10] "Open Source MANO website," [Online]. Available: https://osm.etsi.org/.

[11] "Open Network Automation (ONAP) website," [Online]. Available: https://www.onap.org/.

[12] "Open Platform for NFV (OPNFV)," [Online]. Available: https://www.opnfv.org/.

[13] "OpenStack Compute (nova)," [Online]. Available: https://docs.openstack.org/nova/latest/.

[14] “Horizon: The OpenStack Dashboard Project,” [Online]. Available: https://docs.openstack.org/horizon/latest/.

[15] "Heat’s purpose and vision," [Online]. Available: https://docs.openstack.org/heat/latest/.

[16] "Keystone, the OpenStack Identity Service," [Online]. Available: https://docs.openstack.org/keystone/latest/.

[17] "SONATA - NFV GitHub repository," [Online]. Available: https://github.com/sonata-nfv/.

[18] A. Kapadia et N. Chase, Understanding OPNFV, Accelerate NFV Transformation using OPNFV, Sunnyvale, CA 94085, U.S.A.: Mirantis Inc., 2017.

[19] "Overview of Single Root I/O Virtualization (SR-IOV)," [Online]. Available: https://docs.microsoft.com/en-us/windows-hardware/drivers/network/overview-of-single-root-i-o-virtualization--sr-iov-.

[20] "DPDK - Data Plane Development Kit," [Online]. Available: https://dpdk.org/.

[21] "Enhanced Platform Awareness," [Online]. Available: https://networkbuilders.intel.com/network-technologies/enhancedplatformawareness.

[22] K. C. Glossbrenner, «Availability and Reliability of Switched Services,» *IEEE Communications Magazine,* vol. 31, n° %16, pp. 28-32, 1993.

[23] TOSCA, OASIS, "Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0," 2013.

[24] OASIS TOSCA, Tosca Simple Profile for Network Functions Virtualization (NFV), version 1.0, 2015.

[25] IETF, RFC 6020 - YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF), 2010.

[26] ETSI GS NFV-MAN 001, Network Functions Virtualisation (NFV); Management and Orchestration V1.1.1 (2014-12), 2014.

[27] ETSI GS NFV-IFA 014, “Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification; V2.3.1,” 2017-18.

[28] "OSM Information Model," [Online]. Available: https://osm.etsi.org/wikipub/index.php/OSM_Information_Model.

[29] Open Source MANO, "OSM Information model - release TWO," 2017. [Online]. Available: https://osm.etsi.org/wikipub/images/2/26/OSM_R2_Information_Model.pdf.

[30] "VNF Modeling Requirements," [Online]. Available: http://onap.readthedocs.io/en/latest/submodules/vnfrqts/requirements.git/docs/Chapter5.html.

[31] OpenStack, "VNF Descriptor Template Guide," no. https://docs.openstack.org/tacker/latest/contributor/vnfd_template_description.html.

[32] SONATA NFV, D2.2 Architecture Design, 2015.

[33] B. Ingerson, C. C. Evans et O. Ben-Kiki , Yet Another Markup Language (YAML) 1.0, 2001.

[34] ETSI GS NFV-IFA 011, «Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; VNF Packaging Specification; v2.3.1,» 2017-18.

[35] "OSM Release THREE - Deploying your first Network Service," [Online]. Available: https://osm.etsi.org/wikipub/index.php/OSM_Release_THREE.

[36] "Heat Orchestration Template (HOT) specification," [Online]. Available: https://docs.openstack.org/heat/pike/template_guide/hot_spec.html#hot-spec.