Implementation of VNF Descriptor Extensions for the Lifecycle Management of VNFs

Emanuele Miucci, Giuseppe Monteleone, Giovanni Fausto Andreotti, Paolo Secondo Crosta

ITALTEL S.p.A.
Milan, Italy
e-mail: {emanuele.miucci, giuseppe.monteleone, fausto.andreotti, paolosecondo.crosta}@italtel.com

***Abstract*—This paper focuses on improvements in management and orchestration within the Network Function Virtualization (NFV) domain. The key benefits are related to automation in the Virtual Network Function (VNF) lifecycle, adaptation to different network traffic loads and new models for improving network resilience. These could be achieved by introducing some extensions of the NFV Information Model. In particular, we propose the introduction in the VNF Descriptor (VNFD) of an Information Element (IE) providing the dependencies between Virtual Deployment Units (VDUs) that allows managing the VDUs instantiation process in a more efficient way. We also suggest an extension related to the execution of script(s) - including the possibility to pass parameters - in response to particular events detected by the VNF Manager (VNFM). Since each single component of a VNF may need the execution of specific operations based on its redundancy scheme, we propose a new Information Element describing the high availability features. In addition, we give an example of how to carry information about auto-scaling rules directly in the VNFD defining a possible structure of the *autoScale* IE. Finally, in order to support the validity of the proposed approach, we provide two practical use cases related to the instantiation and scaling lifecycle events of a VNF designed by Italtel and providing Session Border Controller functionality. The main output of the paper is the definition of four extensions to the VNF Information Model - easy to fit into the ETSI specification framework - which has been used to improve the operations in the VNF lifecycle management, as demonstrated in a real implementation scenario.**

Keywords–Network Function Virtualization; Orchestration; Lifecycle Management; VNF Descriptor; User-data; Auto-scaling Rules.

I. INTRODUCTION

Network Function Virtualization (NFV), in addition to Software Defined Networking (SDN), is a rapidly emerging approach in the telecommunication field. By adopting NFV, Communication Services Providers (CSP) expect to achieve consistent cost reductions with respect to the current situation in which network equipment consists of proprietary black boxes, containing a bundle of proprietary Software (SW) and customized Hardware (HW) provided by a single Telecom Equipment Manufacturer. The adoption of the NFV concept is just the starting point to introduce in the Telco world the benefits that virtualization has brought in the Information Technology (IT) sector. Besides significant cost reductions, NFV also raises great expectations on the possibility (a) to achieve a never experienced network flexibility and service agility, (b) to introduce automation in all lifecycle of Network Functions (NFs) from deployment, installation and commissioning to operational phases [1], (c) to adapt the network to different traffic loads thanks to a novel cloud elasticity model, and (d) to develop new models for improving network resilience [2].

In fact, the objective of NFV is to allow Service Operators to achieve a high reduction in capital investments along with greater operational agility by implementing challenging architectural updates and deep changes in service models and operating procedures [3].

Virtualization is not a new technology. What is new is the way to use virtualization in Telco environments. NFV is a paradigm able to switch from manual, complex and error prone configuration processes to a new level of deployment automation, increasing flexibility, agility and the possibility to minimize complexity and errors. After the deployment, when the function is in operation it is possible to perform monitoring, scaling, healing, failover, continuous delivery and infrastructure upgrades.

In the NFV architecture, specified by the European Telecommunications Standards Institute (ETSI) NFV Industry Specification Group (ISG) [4], three main domains are identified:

- Virtualized Network Functions (VNFs), as the software implementation of network functions capable of running over the Network Functions Virtualization Infrastructure (NFVI).
- NFVI, including hardware resources (Compute, Storage, Networking) and the virtualization layer that provides Virtual resources (Virtual Compute, Virtual Storage, Virtual Networking) supporting the execution of the VNFs.
- NFV Management and Orchestration (MANO), which covers the lifecycle management of VNFs and Network Services (NS), managing the resources of NFVI and focusing on all virtualization-specific management tasks necessary in the NFV framework.

In this paper, we will focus on the Management and Orchestration domain. In particular, we propose some extensions of the NFV Information Model, specifically referring to the VNF Descriptor. These extensions can be used to increase efficiency in the lifecycle management of Network Functions and to improve the overall system reliability. Our experience as VNF provider conducted us to identify some flaws in the NFV Information Model and corresponding specific enhancements that future implementations could benefit from. In order to achieve this goal, we introduce some additional Information

Elements (IEs) in the VNF Descriptors (VNFDs) that allow a deeper control of VNFs.

The paper is organized as follows: Section II gives an overview of the ETSI standard model for NFV, with regard to the MANO architecture, including the lifecycle management of Virtual Network Functions and a general description of the NFV Information Model. Section III, the main part of the paper, provides a rationale for the extensions to VNF Descriptors, as well as some practical examples to support the validity of the proposed approach. Section IV provides insights and validation of the proposed extensions based on a real deployment. Finally, Section V draws the conclusions.

II. ETSI NFV Architectural Model

The ETSI NFV architectural framework [5] [6] is shown in Figure 1.

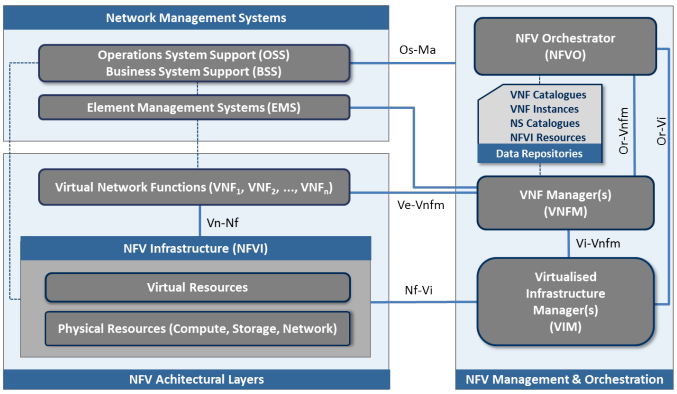

Figure 1. ETSI NFV architectural model.

The functional blocks in the framework can be grouped into three main entities:

1) NFV Architectural Layers,
2) NFV Management and Orchestration,
3) Network Management Systems.

These entities, as well their constituent functional blocks, are connected together using a set of defined reference points. The NFV Architectural Layers include the NFVI and VNFs. NFVI is the combination of both hardware and software resources, which make up the environment in which VNFs are deployed, while VNFs are implementations of NFs that are deployed on those virtual resources.

A. NFV-MANO Framework

The NFV MANO [7] consists of three functional blocks, the Virtualized Infrastructure Manager (VIM), the VNF Manager (VNFM) and the NFV Orchestrator (NFVO), and four data repositories (NS Catalogues, VNF Catalogues, VNF Instances and NFVI Resources). Each of them performs well-defined functions, in particular:

1) VIM: It manages and controls NFVI physical and virtual resources in a single infrastructure domain. This implies that an NFV architecture may contain more than one VIM, with each of them managing or controlling NFVI resources from a given infrastructure provider. In principle, a VIM may be specialized in handling a certain type of NFVI resource (e.g., compute-only or storage only), or could manage multiple types of NFVI resources (e.g., nodes in the NFVI).

2) VNFM: Each VNF instance lifecycle is associated to a VNFM. The VNFM is responsible for the management of the lifecycle of VNFs. A VNFM may manage a single or multiple VNF instances of the same or different types. It is also possible that a single VNFM handles all the active VNF instances for a certain domain.

3) NFVO: It is aimed at combining more than one virtualized network function to create end-to-end network services. To this end, the NFVO functionality can be divided into two broad categories: (a) resource orchestration, and (b) service orchestration. Resource orchestration is used to provide services that support accessing NFVI resources in an abstract manner regardless of the type of VIMs, as well as governance of VNF instances sharing resources of the NFVI infrastructure. Service orchestration deals with the creation of end-to-end services by composing different VNFs, and the topology management of the network services instances.

4) Data Repositories: These are databases that keep different types of information in the NFV MANO. Four types of repositories can be considered: (a) the NS Catalogue is a set of pre-defined templates, which define how network services may be created and how their lifecycle is managed, as well as the functions needed for the service and their connectivity; (b) the VNF Catalogue is a set of templates, which describe the deployment and operational characteristics of available VNFs; (c) the NFVI Resources repository holds information about available/allocated NFVI resources, and (d) the VNF Instances repository holds information about all function and service instances throughout their lifetime.

B. VNF Lifecycle Management

NFV is based on the principle of separating network functions from the hardware where they run on by using virtual hardware abstraction. The virtualization of network functions will change their lifecycle management by introducing automation and flexibility. Lifecycle management of VNFs is possible after a preliminary operation, the so-called VNF Package on-boarding. After that, by accessing to a VNF Catalogue it is possible to create one or more VNF instances of a VNF. A VNF instance corresponds to a run-time instance of the VNF software, i.e., all the VNF components are instantiated and the internal and external network connectivity configured.

During its lifecycle a VNF instance can be in one of the following states:

- instantiable, i.e., the on-boarded process for the VNF has been correctly performed,
- instantiated, i.e., not configured, configured & not in service, configured & in service,
- terminated.

The lifecycle is controlled by a set of operations, described in the following list:

- VNF Instantiation,
- VNF instance Scaling (horizontal/vertical),
- VNF instance Update or Upgrade,
- VNF instance Healing,
- VNF instance Termination.

It is worth mentioning that a subset of lifecycle management (LCM) operations, as VNF Instantiation and Termination, are always available for every single VNF instance.

Some other operations, such as VNF Scaling, are performed if required by the Deployment Flavour of the VNF instance. Some procedures related to VNF lifecycle management provide both manual and automatic mechanisms. Scaling, for instance, can be manually requested or automatically performed when triggered upon the occurrence of specific events defined as criteria for matching rules and actions for scaling. All the operations during the lifecycle are handled by specific workflows that are built on the basis of the tasks to perform and the associated parameters (i.e., in case of instantiation, the VNF ID, the Deployment Flavour, etc.). A workflow has a starting point and different tasks that can be performed sequentially or in parallel, in order to perform all the necessary activities.

A VNFD is used for defining workflows for the automation of specific phases. For instance, by modelling the VNFD using a description language it is possible to describe the VNF with a service template in terms of components (e.g., Virtual Deployment Units or VDUs), relationships (dependencies, connections) and management processes. The management processes can be defined as plans describing how a VNF instance is instantiated and/or terminated considering that the VNF is a complex application composed by different nodes.

C. VNF Information Model

In this subsection, we provide a brief description of the IEs used to carry the necessary information to manage a VNF. In fact, one of the ways the IEs can be used is as part of descriptors in a catalogue or template context.

The IEs to be handled by the NFV MANO, including the ones contained in the VNFD, need to guarantee the flexible deployment and portability of VNF instances on multi-vendor and diverse NFVI environments, e.g., with diverse computing resource generations, diverse virtual network technologies, etc. To achieve this goal, hardware resources need to be properly abstracted and VNF requirements must be described in terms of such abstractions.

With reference to Figure 1, the Vi-Vnfm reference point [8] enables the interaction between the VNFM and the VIM, providing the methods to operate cloud resources on the NFVI, in particular computing, storage and networking resources. After the upload of the VNF Package, the VNFM under operator's request or by a request coming from the Or-Vnfm reference point [9] can start to perform the lifecycle management of a VNF via the Ve-Vnfm reference point [10].

The VNF Package contains all files and artefacts needed to perform the lifecycle management for the associated VNF:

- descriptor (VNFD),
- metadata, scripts and other proprietary artefacts,
- optionally, SW images of the VNF Components (VNFCs).

Recently the Solutions working group of ETSI NFV has approved the ETSI GS NFV-SOL 004 document [11], in which are specified further practical details about the structure and format of the VNF Package.

The VNFD is a template, which describes a VNF in terms of its deployment and operational behaviour requirements. It is primarily used by the VNFM in the process 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 virtualised resources on the NFVI. The VNFD also contains connectivity, interface and Key Performance Indicators (KPIs) requirements that may be used by NFV MANO functional blocks to establish appropriate virtual links within the NFVI between its VNFC instances, or between a VNF instance and the endpoint interface of the others NFs. The VNFD contains all the information needed for the lifecycle management, such as:

- basic information for VNF identification;
- internal virtual links description;
- VDUs description: these data are used to deploy the VNFCs on the NFVI (generally as Virtual Machines - VMs). For each type of VDU, it is defined the flavour of the corresponding VM, the SW image for the VM and the number of VMs to activate. Configuration scripts are also provided for the lifecycle management phases and triggered during the instantiation or by specific events;
- meters or measurements. In fact, when the VNF is in operation, measurements can be collected from the VNF itself and/or from the infrastructure, e.g., the number of session attempts per second; the contemporary active sessions; CPU, RAM, disk usage, etc.
- scaling policies. In general, they are based on the values of the measured parameters and can be used, for example, to add an instance of a VNFC when specific conditions are matched.
- alarms and associated actions. Criteria may be defined in terms of rules to check on the measurements, e.g., the value of a meter is greater than a specific threshold for a specified period of time; etc. When a rule is matched, specific actions can be performed, such as the activation of an healing policy, etc.

The ETSI GS NFV-IFA 011 [12] is the reference specification that provides requirements for the structure and the format of a VNF Package. In particular, it describes how the VNF properties and the associated resource requirements are mapped in an interoperable template, i.e, the already mentioned VNFD.

Its focus is on VNF Packaging, meta-model descriptors and package integrity and security considerations. This specification gives a holistic end-to-end view of the package lifecycle from design to runtime, thus capturing development as well as operational views. This is the result of the analysis performed by the working group aimed to use and potentially refine end-to-end VNF Package lifecycle management operations based on use cases and related actors and NFV Architectural Framework functional blocks impacted. This specification has been recently revised in order to correct errors, ambiguities, misalignments, thus applying some editorial modifications (i.e., corrections of category F and D as described in ETSI TWPs Annex L). However, this edition does not add or modify features, nor does it extend the scope of the former version we dealt with hereafter.

In the following, we describe the extensions - in terms of newly added and/or newly defined IEs - of this specification, along with some related examples of use in order to show the benefits introduced in the lifecycle management of VNFs.

III. ETSI NFV Information Model Extensions

The aim of this section is to provide a detailed description, the rationales, the relationships and the benefits of each proposed extension of the VNF Information Model.

Generally, a VNF is composed by one or more VNFCs. The VNF is described by means of its VNFD, that includes IEs for the description of the VDUs.

As an example, we describe hereafter a real implementation of a decomposed VNF. We refer to a virtualized Session Border Controller (SBC) implemented by Italtel that we will use in the description of the following use cases. This VNF provides its functionalities thanks to five different interworking VNFCs, as depicted in Figure 2:

- a front-end load balancer (FELB) that receives network traffic and distributes it to the other components;
- an operation and maintenance module (OAM);
- a control plane component managing the signalling traffic (SIG) that externalizes the states;
- a states database (States DB) for storing the information of the active signalling sessions;
- a border gateway (BGW) function engaged when audio or media transcoding is required for the incoming traffic.

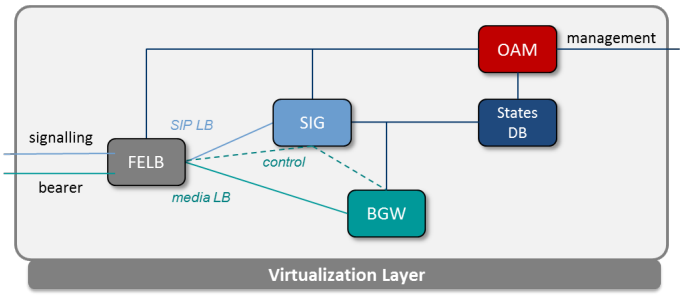

Figure 2. SBC logical components.

These components can be organized and managed to implement protection mechanisms in order to guarantee redundancy and to support high availability requirements.

TABLE I. Overview of Information Model Extensions

ETSI GS NFV-IFA 011 Specification Extensions	
dependencies	IE in the VNFD (vnfd).
metadataScript	IE in the VDU section (vnfd:vdu).
highAvailability	IE in the VDU section (vnfd:vdu).
autoScale	IE in the VNFD (vnfd).

The implementation of this decomposed VNF suggested the Information Model extensions to the ETSI GS NFV-IFA 011 [12] reference document - *dependencies*, *MetadataScript*, *highAvailability* and *autoScale* IEs -, summarized in Table I, with the relationships within the descriptor at the VNFD (vnfd) or VDU (vnfd:vdu) level. In the following, we provide a specific description for each of them.

A. Dependencies

In this subsection, we provide the description of the *dependencies* and *VduDependencies* IEs. It is possible to make use of these extensions whenever it is necessary to express dependencies among the VDUs during the instantiation process. In fact, sometimes it is necessary to coordinate the process of instantiation with information that is available - at platform level (e.g., IP addresses) or application level - at specific times. As originally proposed in the ETSI MANO specification [7], we believe it is needed to include in the VNFD an IE providing the dependencies between VDUs since it describes constraints that affect the structure of a VNF.

Table II shows the structure of the *dependencies* IE that has to be added to the VNFD standard description [12].

TABLE II. Dependencies Information Element

Attribute of the *dependencies* VNFD IE		
Attribute	**Description**	
dependencies	**Qualifier**	M
	Cardinality	0...N
	Content	VduDependencies
	Description	Describes dependencies between VDUs. Defined in terms of source and target VDU, i.e., target VDU "*depends on*" source VDU. In other words, sources VDU shall exist before target VDU can be instantiated / deployed.

The *VduDependencies* IE provides indications on the order in which VDUs associated to the same VNFD have to be instantiated. The contents of a *VduDependencies* type shall comply with the format provided in Table III.

TABLE III. VduDependencies Information Element

Attributes of the *VduDependencies* IE		
Attribute	**Description**	
vdu-id	**Qualifier**	M
	Cardinality	1...N
	Content	Identifier
	Description	The listed VDUs shall be instantiated before the VDUs listed in the target parameter.
depends-on	**Qualifier**	M
	Cardinality	0...N
	Content	0...N
	Description	The listed VDUs shall be instantiated after the VDUs listed in the source parameter have been completely instantiated.

In Figure 3, it is shown a sequence diagram based on a real implementation of the Italtel VNF SBC. It is worth to mention that in this case, all the VNFCs should be instantiated after the OAM component, since it has a central role coordinating the communications with the VNF Manager on behalf of all the other VNFCs.

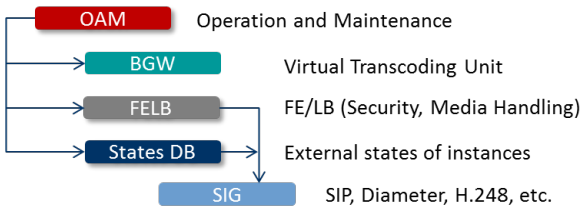

Figure 3. SBC components instantiation sequence diagram.

In Figure 4, the dependencies explained above have been expressed by using the JavaScript Object Notation (JSON) [13].

```
"dependencies": [{
    "vdu-id": "felb",
    "depends-on": "oam"
    }, {
    "vdu-id": "sig",
    "depends-on": "oam"
    }, {
    "vdu-id": "states",
    "depends-on": "oam"
    }, {
    "vdu-id": "bgw",
    "depends-on": "oam"
    }, {
    "vdu-id": "sig",
    "depends-on": "felb"
    }, {
    "vdu-id": "sig",
    "depends-on": "states"
}]
```

Figure 4. *dependencies* IE example.

Alternatively, ETSI envisages the use of a scripting language to express dependencies on virtual resources, but at the time of this writing, no consensus has been reached yet about the format to be used and the standardization process of a Domain Specific Language (DSL) is still underway.

B. MetadataScript

In this subsection, we provide the description of the *metadataScript* and *LifeCycleMetadataScript* IEs. These extensions are related to the execution of script(s) in response to particular events detected on a VNFM reference point. The ETSI GS NFV-IFA 011 specification [12] already supports the execution of scripts - but only at the VNF level - with the *LifeCycleManagementScript* IE. Scripts can be launched in response to lifecycle events or external stimulus detected by the VNFM. These LCM scripts should be embedded in the VNF Package and used in the LCM execution environments provided by generic VNF Managers. Although in par.6.2.6, this specification provides a list of requirements (VNF_PACK.LCM.001) for the scripting Domain Specific Language, no practical details are yet available.

Table IV shows the structure of the *MetadataScript* IE that has to be added to the VDU standard description.

TABLE IV. METADATASCRIPT INFORMATION ELEMENT

Attribute of the *metadataScript* VDU IE		
Attribute	**Description**	
dependencies	**Qualifier**	M
	Cardinality	0...N
	Content	LifeCycleMetadataScript
	Description	Includes a list of events and corresponding scripts producing metadata required during the VDU lifecycle.

A *LifeCycleMetadataScript* IE, instead of the *LifeCycleManagementScript* formerly defined in the original specification, has been defined and extended to comply with specific needs originated from practical use cases.

The attributes of the *LifeCycleMetadataScript* IE shall follow the indications provided in Table V. The advantages of this extension, compared with the existing standard specification, are (a) the possibility to execute script(s) at the VDU level, and (b) the possibility to pass parameter(s) to the script(s).

TABLE V. LIFECYCLEMETADATASCRIPT INFORMATION ELEMENT

Attributes of the *LifeCycleMetadataScript* IE		
Attribute	**Description**	
event	**Qualifier**	M
	Cardinality	1
	Content	String
	Description	Describes a VNF lifecycle event or an external stimulus detected on a VNFM reference point.
script	**Qualifier**	M
	Cardinality	1
	Content	Script
	Description	Script name.
role	**Qualifier**	M
	Cardinality	1
	Content	String
	Description	Describes the role of the VDU in redundancy scheme(s). Possible values are ”Active” or ”Passive”.
parameter	**Qualifier**	M
	Cardinality	0...N
	Content	String
	Description	VDU specific parameters passed to the script. Each of them represents the run-time value of a NFVI resource (e.g., IP address, VNFC instance name, etc.).

In Figure 5, it is provided an example based on a real implementation of the Session Border Controller VNF. It is worth noting that this IE allows the VNFM a complete flexibility in the lifecycle management process of different VNFs/VNFCs: in this example, the script for the instantiation of the SIG component needs information from the infrastructure (i.e., the IP address of the connection point *cp_sig_int*, the *hostname* of the VNFC and the related *domain_name*), which will be available only at run-time.

According to our experience, the proposed syntax is general and can be easily adapted in order to suit different VNFM

providers.

```
"LifeCycleMetadataScript": [{
   "event": "INSTANTATION",
   "script": "instantiate_sig",
   "role": "active",
   "parameters": [
     "$$param.cp_sig_int.ipaddress",
     "$$param.hostname",
     "$$param.domain_name"
   ]
}]
```

Figure 5. *LifeCycleMetadataScript* IE example.

C. HighAvailability

In this subsection, we provide the description of the *highAvailability* attribute. Availability is defined as the state to perform a required function at a given instant of time or at any instant of time within a given time interval, assuming that the external resources, if required, are provided. This attribute is important for telecom operators that want to offer their customers services that perform as expected whenever the service is requested.

Comparing the VDU IE originally proposed in the ETSI MANO specification [7] with the one described in ETSI GS NFV-IFA 011 [12], the *high_availability* IE is no longer specified. The reason provided by ETSI is based on the assumption that the VNFM alone can hardly manage the multitude of redundancy schemes: high availability policies should be performed by each single VNFC at the application level.

Instead, in our opinion, an attribute specified at the VDU level allows the VNFM to execute specific operations tailored for each single instance, thus simplifying the implementation of the VNF itself.

Table VI shows the structure of the *highAvailability* IE that has to be added to the VDU standard description.

TABLE VI. HIGHAVAILABILITY INFORMATION ELEMENT

Attribute of the *highAvailability* VDU IE		
Attribute	**Description**	
highAvailability	**Qualifier**	M
	Cardinality	0...N
	Content	Enum
	Description	Defines redundancy model to ensure high availability. Possible values are "ActiveActive" or "ActivePassive". *ActiveActive*: implies that two instance of the same VDU will co-exists with continuous data synchronization. *ActivePassive*: implies that two instance of the same VDU will co-exists without any data synchronization. If not provided: no specfic redundancy model is applied.

For example, the statement "*highAvailability*": "*ActivePassive*" implies the active part to request a set of parameters, which can be different from the configuration set, which is needed by the passive part. Furthermore, the active and passive counterparts would require a different set of instantiation/configuration scripts. As shown in Figure 5, this condition could be easily enforced by using an additional attribute defined in the *LifeCycleMetadataScript* IE, i.e., the *role* attribute, which can assume "*active*" (or "*passive*") values, as described in Table V.

D. AutoScale

An important aspect of the scaling, as VNF lifecycle operation, is that scaling may be automatically performed by the VNFM (auto-scaling) [14]. A specific attribute in the VNFD, namely *isAutoscaleEnabled* of the *VnfConfigurableProperties* IE, shall be used (set to true) in order to enable this feature for the VNF. If auto-scaling is allowed, the *autoScale* IE can be used to describe the auto-scaling rules, i.e., the conditions at which to perform a scaling operation. These rules are generally based on the values of the VNF indicators, or on the values of VDU/VL monitoring parameters defined in the VNFD, or on the combination of them. Moreover, the rules may be directly included in the VNFD or expressed using external scripts.

Recently the IFA working group, developing the IFA-023 [15] document, has proposed a stable study on how to integrate policy management in NFV MANO architecture. In this way, policies will give to MANO functions more automatic characteristics, as required in a virtualized network environment. The model adopted consists of two logical entities: a Policy Administration Point (PAP), which defines the policy, and a Policy Function (PF), which evaluates it thus making the consequent decisions. In this context, the auto-scale rules become an auto-scaling policy pre-defined in the VNFD by the VNF provider, which can be seen as the PAP, and enforced by the VNFM acting as PF. Each time the VNFM detects the occurrence of the condition(s) of the auto-scaling policy, it executes the action(s) as indicated in the policy itself.

TABLE VII. AUTOSCALE INFORMATION ELEMENT

Attribute of the *autoScale* VNFD IE		
Attribute	**Description**	
autoScale	**Qualifier**	M
	Cardinality	0...N
	Content	Rule
	Description	Rule that determines when a scaling action needs to be triggered on a VNF instance e.g., based on certain VNF indicator values or VNF indicator value changes or a combination of VNF indicator value(s) and monitoring parameter(s).

In this subsection, we provide the description of the *autoScale* IE. More precisely, we provide the description of the *Rule* attribute of this IE. In fact, the *autoScale* is already defined in IFA-011 while a detailed description of the *Rule* is missing. Table VII resumes the structure of the *autoScale* IE as specified by IFA-011 document.

As mentioned in the description of Table VII, the *Rule* (conditions and actions) may be expressed as a script: this allows to provide specific instructions using a DSL. Anyway, both IFA-011 and IFA-023 documents do not contain further information about structure or domain specific language to be used to create auto-scaling rules/policy.

In light of the above, we propose a possible structure of the Rule attribute in the *autoScale* IE.In particular, they should be expressed by indicating the following attributes:

- *ruleId*, the unique identifier for the rule within the VNFD.
- *description*, this attribute gives a human readable description of the rule itself.
- *deploymentFlavour*, an explicit reference to the Deployment Flavour(s) for which the rule will take effect. Since the cardinality can be greater than one, we make explicit reference to the Deployment Flavour(s) for which the rule will take effect.
- *conditions*: a list of *statement*, i.e., the conditions that must occur to perform the *actions* and the number of consecutive samples (*nTime* attribute) for which they have been verified (default 1). The identifiers in the *statement* are the VNF indicators and the monitoring parameters contained in the VNFD. Logical/arithmetical operators are used as defined in the specification "ISO/IEC 9899" [16].
- *actions*: attribute containing the indication of the type of scaling and a reference to the involved scaling aspect.

Figure 6 shows an example of rule to perform automatic scale out of the VDU SIG for the Italtel SBC.

```
"autoScale": [{
   "ruleId": "SIG_AUTOSCALE_OUT",
   "description": "Autoscale Rule to
       ↪ scale OUT the vdu SIG",
   "flavourId": ["dfTest"],
   "conditions": [{
      "statement": "(KPI_SIG_1 > 30)",
      "nTimes": 2
   }],
   "actions": [{
      "type": "SCALE OUT",
      "aspectId": "sig_scale"
   }]
}]
```

Figure 6. Proposed structure of the *autoScale* IE in the VNFD.

The attributes *conditions* and *actions* are the core of the auto-scaling rule/policy. The former contains a list of *statement*, i.e., the conditions that the VNFM must verify to perform the *actions* and the number of consecutive samples for which they have been verified. In this specific case the number of consecutive samples is equal to 2 to avoid an immediate reaction in case of a burst in the measured value.

Thanks to this simple data model extension, we can map, directly inside the VNFD, the details about auto-scaling in/out rules for the VNF components to be scaled (the SIG component in the specific case).

IV. Validation Use Cases

In this section we provide a description of two different use cases in order to validate the benefits introduced by the proposed extensions. The former practical example take into account *LifeCycleMetadataScript*, *highAvailability* and *dependencies* extensions showing how the information carried by those IEs can be used during the instantiation lifecycle event. The latter provides an example of the *Rule* attribute composing the *autoScale* IE, thus specifying the scripts conditions and actions applied to a real deployment.

A. Cloud-Init Use Case

In Subsection III-B, the mechanism that associates the execution of a script to a lifecycle event has been described (*LifeCycleMetadataScript*). This association, contained in the VNFD, is extremely powerful for a generic VNFM and useful for a VNF provider that needs some specific configuration in a given step of the lifecycle of its virtual network function. In particular, the instantiation phase of a VNF is a very delicate process and this mechanism shall provide all the data in order to bring the VNF in a valid state, in which all the VMs have to be up and running.

Several software tools make possible to reach the above-mentioned target. Some of them consist of an agent installed in the VMs composing the VNF through which the VNFM is able to command the execution of scheduled scripts/commands [17]. Other tools, already available on recent Linux distributions, consume scripts containing user-data section. One of the most popular formats for this kind of scripts is the *cloud-config* file format while *cloud-init* is a well know Linux program designed to run these scripts [18]. They are particularly useful for initial configuration on the very first boot of a server.

Since this second solution is extremely helpful in real contexts, we decided to add it when we developed our own VNF and VNFM. As a confirmation of the goodness of our decision, recently ETSI IFA working group has started a new discussion on the possibility to introduce a user-data section in the next version of the VNF Descriptor, together with a VNFM built-in mechanism to pass real-time values as proposed in Subsection III-B.

Another important aspect related to the introduction of this feature is that, if the VIM is Openstack [19], it natively accepts user-data section as optional attribute of the VM creation command.

```
#cloud-config
users:
  - name: username#1
    <user#1 options>
  - name: username#2
    <user#2 options>
  ...
write_files:
  - content: |
      <lines to write to the file>
    path: <dir>/<filename>
  ...
runcmd:
  - command#1
  - command#2
  ...
```

Figure 7. Basic *cloud-config* file structure.

In Figure 7 we report the basic structure of a cloud-config

file that Cloud-init is able to interpret. There are several parts of this script that it worth to note:

- each cloud-config file must start with the string "*#cloud-config*" on the first line. In this way, cloud-init will be immediately aware to interpret the rest as cloud-config file;
- "*users*" key is followed by a list of items, identified by a "-" in a new indentation level, representing the different users enabled on the VM. Various options are available for this section;
- "*write_files*" key is used to open a section describing all the data, placed after the "*content*" line, that will be written in a file specified in the "*path*" line;
- "*runcmd*" key contains the list of commands that will be executed on the VM by the cloud-init process.

Note that indentation is a key aspect in a cloud-config file: it is necessary to differentiate and interpret the various levels in the script.

In the rest of the section, we will show a practical use case in which the cloud-config scripts are executed during the instantiation phase of the Italtel SBC. Furthermore, we will show how the first three IEs introduced in Section III can work together providing essential info in order to obtain a VNF running and eventually ready for service configuration. As already said, each VDU described in the VNFD of the Italtel SBC has a *LifeCycleMetadataScript* IE containing the reference to the cloud-config script associated to the "INSTANTIATION" event. The "*parameters*" attribute is the list of parameters that will be replaced in this script by the VNFM with the run-time values returned by the VIM during the instantiation of the VNF.

```
"LifeCycleMetadataScript": [{
   "event": "INSTANTATION",
   "script": "instantiate_oam_active",
   "role": "active",
   "parameters": [
      "$$param.cp_oam_int.ipaddress",
      "$$param.hostname",
      "$$param.domain_name",
      "$$param.passive.cp_oam_int.ipaddress",
      "$$param.passive.hostname"
      ]
   },{
   "event": "INSTANTATION",
   "script": "instantiate_oam_passive",
   "role": "passive",
   "parameters": [
      "$$param.cp_oam_int.ipaddress",
      "$$param.hostname",
      "$$param.domain_name",
      "$$param.active.cp_oam_int.ipaddress",
      "$$param.active.hostname"
      ]
}]
```

Figure 8. Proposed structure of the *autoScale* IE in the VNFD.

Furthermore, if a VDU has the *highAvailability* IE equal to "*ActivePassive*" it is possible to differentiate the scripts mechanism for the two instances having separate items in the JSON array describing the *LifeCycleMetadataScript* IE: one for the VNFC with the "*active*" role, and the other for the "*passive*" one.

Figure 8 shows an example of this configuration for the VDU OAM: since it has an Active/Passive redundancy model, cloud-config scripts with different commands will be executed during the instantiation of the master and the slave instance. Note that it is possible to customize the parameters lists based on what kind of information the two VMs need at their boot. In our example, for high availability purpose, the passive VNFC OAM needs to know the hostname and IP address of a connection point related to the active counterpart and vice versa. In other configurations, a VDU (e.g., VDU_A) may need some infrastructure details related to another VDU (e.g., VDU_B). In this case, the *VduDependencies* IE is extremely useful since, by defining proper dependency rules, the manager is able to instantiate the VNFC related to the VDU_B, before the component coming from the VDU_A. In this way, all the infrastructure information needed by the VDU_A will be available to the VNFM at the right time.

Once all the parameters in the cloud-config script are replaced with the run-time values, the manager will add it to the virtual resource allocation request sent to the VIM. The infrastructure manager will create the VM on the NFVI and, at the very first boot, cloud-init process will start reading the cloud-config file and performing all the tasks listed in it. Repeating this procedure for all the VNFCs, the result will be a VNF up and running, able to receive further configuration data at service level.

B. VNFD Auto-scaling Use Case

As already mentioned, the VNF Descriptor has a key role in the lifecycle management. In the last subsection we have shown how the details contained in it about architecture, virtual resources required and actions to be taken in certain conditions, are used by the VNFM in order to instantiate a VNF ready to work.

As it is depicted in Figure 9, the VNFM uses the measurement information in the VNFD to collect data and the rules to apply the consequent actions. They are the execution of workflows that can be preconfigured or expressed with lifecycle management scripts.

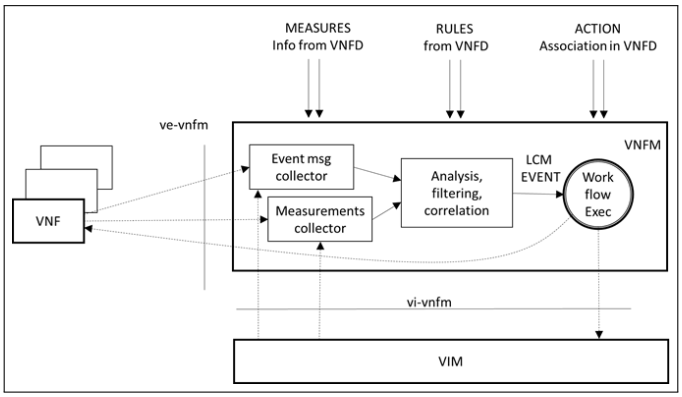

Figure 9. VNFM architecture.

A lifecycle event, that is particularly attractive in cloud environment, is the "SCALING". It represents the ability to

dynamically extend or reduce resources allocated to a VNF. There are four types of scaling:

- OUT - adds new instances (e.g., VM);
- IN - removes instances;
- UP - increases resources already allocated (e.g., vCPU of a VM);
- DOWN - decreases resources already allocated.

Although not all the VNFs support these operations, they are very useful in those scenarios in which the incoming traffic increases and new components are necessary to guarantee the execution of the network function with the same level of performance. The Italtel SBC has two VNFCs that scale in/out: the SIP signaling module (SIG), and the transcoding media gateway (BGW).

In the rest of the section, we show a practical application of the auto-scaling use case. By varying the incoming signaling traffic managed by the SBC VNF, the VNFM will perform the SIG scale in/out following the info contained in the *autoScale* IE.

In Figure 10 it is reported the average vCPU load (in percentage) of the two SIG components initially instantiated by the VNFM. At this point, the VNF is fully configured, but there is no incoming traffic.

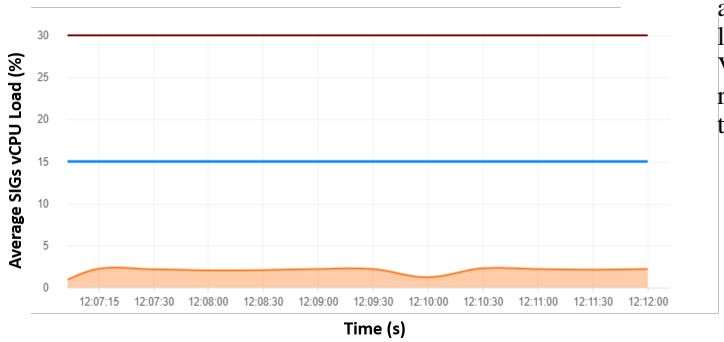

Figure 10. Average vCPU Load (%) of SIG VNFCs without incoming traffic.

In particular, the ”scale out” operation, i.e., the instantiation of a new instance of a SIG component, is triggered when the vCPU load (averaged on the different instances of SIG VNFC already deployed on the NFVI) measured by the VNFM is greater than the predefined threshold of 30% (red line in Figure 10) for two consecutive sampling periods (by default scheduled every 30 seconds). On the other hand, the ”scale in” operation is performed when the load goes below the threshold of 15% (blue line in Figure 10) for two consecutive samples.

This behavior is depicted in Figure 11, in which there are two aspects that are worth noting:

- after the scale out command, the VNFM will wait another two sampling periods before executing another scaling procedure: this is due to the fact that the new VM can take some time to boot and start all its applicative processes. To perform scaling without considering any guard interval would lead to an unstable system;
- once the new SIG component is up and running, the overall traffic will be distributed on an higher number of instances of this VNFC type and so, maintaining the same amount of incoming traffic, the average vCPU load will decrease remaining between the two thresholds, in the interval desired and designed as the optimal way of working of this specific VNFC.

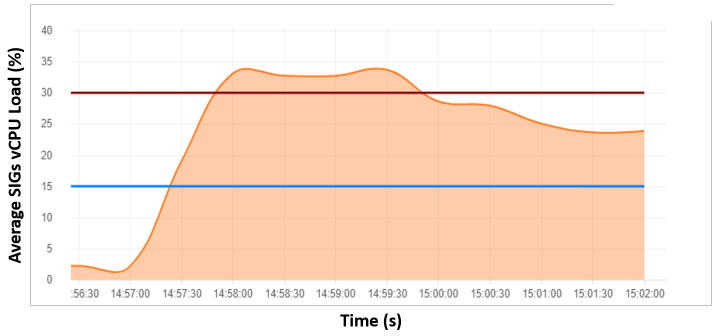

Figure 11. Average vCPU Load (%) of SIG VNFCs when incoming traffic increases.

The same threshold mechanism is defined to automatically scale in the SIG components if the selected performance metric becomes lower than 15%. This is the case reported in Figure 12, where the amount of incoming traffic decreases to such a value that the average vCPU load falls below the blue line for two consecutive sampling periods. Consequently, the VNFM commands to the VIM the termination of all the virtual resources associated to one of the VNFCs SIG instantiated on the NFVI, whose service is no longer required.

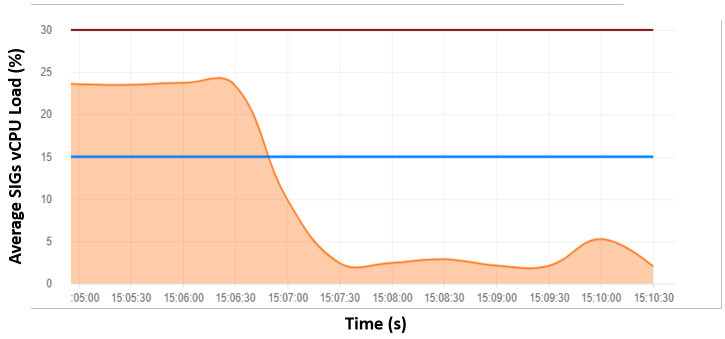

Figure 12. Average vCPU Load (%) of SIG VNFCs when incoming traffic decreases.

V. Conclusions

In this paper, some improvements have been proposed in Management and Orchestration of Virtual Network Functions, based on extensions of the Information Model specified by ETSI. The need for these additional IEs in the VNFD/VDU descriptor(s) has been originated from a real implementation of a novel NFV-compliant Session Border Controller solution. The key points addressed have been more flexibility in the management of network functions and increased reliability of virtualized systems. As a result of this work, we provided a detailed description, rationales, relationships and possible benefits coming from the new attributes, as well as practical examples to support the validity of the proposed approach. The extensions have been applied and successfully validated

in two real use cases related to different lifecycle events of the Italtel SBC solution. These examples demonstrate how the proposed extensions can be very useful to improve VNF life-cycle management and easy to fit into the ETSI specification framework.

REFERENCES

[1] G. F. Andreotti, P. S. Crosta, E. Miucci, and G. Monteleone "NFV Information Model Extensions for Improved Reliability and Lifecycle Management," SOFTNETWORKING 2017, April 23 - 27, 2017 - Venice, Italy.

[2] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An Approach for Service Function Chain Routing and Virtual Function Network Instance Migration in Network Function Virtualization Architectures," in IEEE/ACM Transactions on Networking , vol.PP, no.99, pp.1-18.

[3] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," in IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 236-262, Firstquarter 2016.

[4] ETSI - NETWORK FUNCTIONS VIRTUALISATION [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv [Nov. 28, 2017].

[5] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV 002 V1.1.1: Network Functions Virtualization (NFV); Architectural Framework," October, 2013.

[6] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV 003 V1.2.1: Network Functions Virtualization (NFV); Terminology for Main Concepts in NFV," December, 2014.

[7] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV-MAN 001 V1.1.1: Network Functions Virtualization (NFV); Network Functions Virtualization Management and Orchestration," December 2014.

[8] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV-IFA 006 V2.1.1: Network Functions Virtualization (NFV); Management and Orchestration; Vi-Vnfm reference point Interface and Information Model Specification," April 2016.

[9] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV-IFA 007 V2.1.1: Network Functions Virtualization (NFV); Management and Orchestration; Or-Vnfm reference point Interface and Information Model Specification," October 2016.

[10] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV-IFA 008 V2.1.1: Network Functions Virtualization (NFV); Management and Orchestration; Ve-Vnfm reference point Interface and Information Model Specification," October 2016.

[11] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV-SOL 004 V2.3.1: Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification", July 2017.

[12] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV-IFA 011 V2.3.1: Network Functions Virtualization (NFV); Management and Orchestration; VNF Packaging Specification," August 2017.

[13] The JSON Data Interchange Format [Online]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf [Nov. 28, 2017].

[14] P. Tang, F. Li, W. Zhou, W. Hu and L. Yang, "Efficient Auto-Scaling Approach in the Telco Cloud Using Self-Learning Algorithm," 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, 2015, pp. 1-6.

[15] ETSI Industry Specification Group (ISG) NFV, "ETSI GR NFV-IFA 023 V3.1.1: Network Functions Virtualisation (NFV); Management and Orchestration; Report on Policy Management in MANO," July 2017.

[16] ISO/IEC 9899, Information technology Programming languages C.

[17] Open Baton: NFV compliant MANO framework. Generic VNF Manager [Online]. Available: https://openbaton.github.io/documentation/vnfm-generic/ [Nov. 28, 2017].

[18] Cloud-init: The standard for customising cloud instances [Online]. Available: https://cloud-init.io/ [Nov. 28, 2017].

[19] Openstack: Open source software for creating private and public clouds [Online]. Available: https://www.openstack.org/ [Nov. 28, 2017].