ETSI GS NFV-EVE 011 V3.1.1 (2018-10)

Network Functions Virtualisation (NFV) Release 3; Virtualised Network Function; Specification of the Classification of Cloud Native VNF implementations

Reference
DGS/NFV-EVE011

Keywords
cloud, NFV

Contents

Intellectual Property Rights

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies a set of non-functional parameters to classify and characterize any VNF implementation including, for example, level of separation of logic and state, degree of scale-out, memory footprint, use of accelerators, and more. The present document contains normative provisions using this set of non-functional parameters in order to classify the VNF implementations as cloud native.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI GS NFV-REL 006 (V3.1.1): "Network Functions Virtualisation (NFV) Release 3; Reliability; Maintaining Service Availability and Continuity Upon Software Modification".

[2] ETSI GS NFV-IFA 010 (V3.1.1): "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Functional requirements specification".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are not necessary for the application of the present document, but they assist the user with regard to a particular subject area.

[i.1] ETSI NFV Network Operators Council White Paper (02-2017): "Network Operator Perspectives on NFV priorities for 5G".

[i.2] ETSI GS NFV-SWA 001 (V1.1.1) "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture".

[i.3] NFV White Paper: "Network Function Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for action", Oct 22-24, 2012.

NOTE: Available at: https://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[i.4] ETSI GS NFV-REL 003 (V1.1.2): "Network Functions Virtualisation (NFV); Reliability; Report on Models and Features for End-to-End Reliability".

[i.5] ETSI GS NFV-REL 001 (V1.1.1): "Network Functions Virtualisation (NFV); Resiliency Requirements".

[i.6] ISO/IEC 17788:2014: "Information technology - Cloud computing - Overview and vocabulary".

[i.7]	ETSI GS NFV-EVE 004 (V1.1.1): "Network Functions Virtualisation (NFV); Virtualisation Technologies; Report on the application of Different Virtualisation Technologies in the NFV Framework".
[i.8]	ETSI GR NFV-EVE 010 (V3.1.1): "Network Functions Virtualisation (NFV) Release 3; Licensing Management; Report on License Management for NFV".
[i.9]	ETSI GR NFV-EVE 008 (V3.1.1): "Network Functions Virtualisation (NFV) Release 3; Charging; Report on Usage Metering and Charging Use Cases and Architectural Study".
[i.10]	3GPP TS 23.501 (V15.1.0): "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; System Architecture for the 5G System; Stage 2 (Release 15)".
[i.11]	3GPP TS 23.502 (V15.1.0): "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Procedures for the 5G System; Stage 2 (Release 15)".
[i.12]	ETSI GS NFV 003 (V1.4.1): "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

3 Definition of terms and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS NFV 003 [i.12] and the following apply:

cloud native VNF: VNF with a full adherence to cloud native principles, or a VNF that is transitioning to cloud native principles

NOTE: The definition captures the understanding of the term as used in the present document.

data repository: volumes of storage in a dedicated entity that persists the VNF data and exposes access to the stored data

NFV micro-service: atomic service module, delivered as an all-inclusive software package, that covers a specific and coherent functional scope, is consumable over network interfaces, is managed independently from other micro-services, and runs as a computing process

published API: publicly available application programming interface that provides developers with programmatic access to a software application or web service

remote storage: data storage approach where the Data repository used by VNF/C instances is located in different NFVI-node/s than the VNF and is accessible over network interfaces

VNF Product Characteristics Descriptor (VNFPCD): artefact describing the non-functional characteristics of a VNF product

NOTE: Non-functional characteristics described in the VNFPCD include qualitative characteristics regarding VNF resiliency, performance, scalability, design, capacity, security, usability.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS NFV 003 [i.12] and the following apply:

API	Application Programming Interface
CPU	Computer Processor Unit
DOPFR	Dynamic Optimization of Packet Flow Routing
EM	(network) Element Manager
LCM	Lice Cycle Management
MANO	Management and Orchestration
NSD	Network Service Descriptor
OS	Operating System

PaaS	Platform as a Service
SLA	Service Level Agreement
UE	User Equipment
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNFPCD	VNF Product Characteristics Descriptor

4 Overview

The present document focusses on the characterization of Virtualised Network Functions (VNF) as part of their configuration and deployment in "the Cloud". Such VNFs are assumed to be implemented using generic cloud computing techniques beyond virtualization [i.1]. For example, the VNFs can be built with re-usable components as opposed to a unique - and potentially proprietary - block of functions.

Cloud native VNFs are expected to function efficiently on any network Cloud, private, hybrid, or public. The VNF developer is therefore expected to carefully engineer VNFs such that they can operate independently in the desired Cloud environment. Cloud environment can be implemented based on hypervisor/VM or container technology. This is an indication of the "readiness" of VNFs to perform as expected in the Cloud. The objective of the present document is to develop the characterization of the "Cloud Readiness" of VNFs.

From an operator perspective, it is essential to have a complete description of cloud native readiness of VNFs; this description will help operators in their VNF selection process. To do this, it is essential that a set of non-functional parametric characterizations be developed that appropriately describe the cloud native nature of VNFs. Non-functional parameters describe the environmental behaviour of VNFs residing in the Cloud. They do not describe the actual working functions of the VNF; rather they describe how the VNF can reside independently in the Cloud without constant operator involvement.

The present document considers not only the "pure" cloud native VNF implementations (e.g. no internal resiliency or state) but also some transition implementations to cloud native such as the VNFs with internal resiliency.

Non-functional characteristics of a cloud native VNF are described through a VNF Product Characteristic Descriptor (VNFPCD) created by the VNF provider. Usage of the cloud native VNFPCD is as follows:

- The cloud native VNFPCD is used by an operator to decide on what VNF product to deploy to fulfil a particular functionality, when the decision is based on non-functional parameters.
- The VNFPCD can be used in a VNF market place for a standardized description of the VNF products non-functional characteristics and as such can be checked/searched for automatically.

The intent of the present document is to identify a minimum set of non-functional parameters by which VNFs are characterized as cloud native. The non-functional parameters are classified according to the specific environmental behaviour of the VNF.

Each behaviour then provides a list of specific non-functional parameters along with specific requirements such that the cloud native nature of the VNF can be satisfactorily established.

5 Non-functional parameters for cloud native VNF Classification

5.1 Resiliency

5.1.1 Introduction

One of the benefits expected from NFV is the option to repair service failures through automated reconfiguration of the service to move traffic loads to new VNF/VNFC software instances [i.3]. VNFCs are essentially components of applications deployed via cloud computing. The cloud computing paradigm [i.4] treats all applications as replaceable commodity units using the same mechanisms (e.g. create new instance and transfer load). In this paradigm, operation of the system is expected to continue despite the presence of failures. Some cloud operating environments continually exercise failover mechanisms during normal operations [i.5]. In such an environment, a cloud native VNF contributes to the resiliency of the network services. The high resiliency mechanisms are internal to the VNF itself; alternately the network service (NS) provides the mechanisms such that VNFs can be replaced quickly without negative effects on the NS's resiliency.

The resiliency of a VNF is impacted by characteristics such as the level of separation of logic and state. VNF state handling is addressed in clause 5.5. A cloud native VNF is responsible for meeting its resiliency goals, taking into account the expected availability of the targeted virtualization environment. To comply with the VNF resiliency expectations, the VNF design is expected to satisfactorily overcome problem situations such as the following:

- Resource outage caused by a single failure platform problem including potential failures of:
  - Hypervisor or other components of the virtualisation layer;
  - Compute resources;
  - Storage (outage or inaccessibility with or without data loss);
  - Connections (inter or intra VNF);
- Other outage situations would include multiple failures or outage of complete NFVI PoP;
- Significant planned downtime for NFVI PoP or parts of it such as the infrastructure goes through hardware and software upgrades;
- Failures of MANO:
  - functional blocks;
  - interworking between functional blocks;
- Failure of interworking between MANO functional blocks with the VNF;
- Planned downtime due to MANO upgrades.

A number of software resiliency characteristics are considered here to demonstrate how cloud native VNFs can achieve their resiliency expectations:

- VNFs with high resiliency expectations implemented using internal mechanisms;
- VNFs with low resiliency expectations are covered by external mechanisms, but need to support those mechanisms by providing certain information.

NOTE 1: VNFs with low resiliency guarantee may still implement internal mechanisms, e.g. for redundancy, but in their case e.g. a VNF internal single point of failure can be easier to accept because they can rely on external mechanisms in some situations.

NOTE 2: In some cases, both internal and external mechanisms for service recovery might be in place. External mechanism needs to be triggered any time virtualised resources fail, as that cannot be repaired by the VNF. In this respect, the description of VNFs with high or low resiliency expectations cannot really classify the cloud native VNF, but rather the implemented mechanisms for service recovery. Nevertheless, this clause uses an outside view on the VNF for the description.

NOTE 3: Aspects of geographic redundancy of VNFs are not covered here, since these aspects are not useful to realize a classification of VNFs.

The main mechanisms and requirements are listed in the following clauses.

5.1.2 Intra-VNF redundancy

In many cases resiliency is achieved on VNF level through redundancy of VNFCs, by distributed VNF architecture. By having multiple VNFC instances, it is possible to spread the VNFC instances out across servers, racks, data centres, and geographic regions. This level of redundancy can mitigate most failure scenarios and has the potential to provide a service with acceptable availability. Careful consideration of VNFC modularity also minimizes the impact of failures when an instance does fail.

In this case cloud native VNFs will be developed with sufficient levels of redundancy such that these VNFs perform in compliance with the resiliency requirements. It is critical that the redundancy mechanisms that are built into these VNFs are suitable to achieve the resiliency required by the service provider. The level of resiliency of the VNF constitutes a comparison criterion between different VNF implementations. VNF redundancy characteristics that need to be accurately described include the following:

- Redundancy Model: Active-active, active-spare, N+M redundancy.

NOTE 1: The redundancy model could be different per VNFC type in a cloud native implementation.

- Recovery Time: The mean time between the moment when a failure is detected and until the service provided by the VNFC is available again.
- Single Point of Failure: Typically, VNFs can only meet high resiliency expectations by providing redundancy for all components, links, etc. If a VNF can meet the resiliency expectations without redundancy on a certain point (e.g. because the failure probability is low enough), this single point of failure needs to be documented by the VNF provider.
- Impact of failure: Expected number of connections/sessions/flows/subscribers/, etc. impacted by common failures, e.g. VNFC/VNF/links instance failures.

NOTE 2: The impact of a failure will vary largely depending on the failure, the load on a VNFC/VNF and the VNF deployment flavour.

Details are discussed in ETSI GS NFV-REL 003 [i.4].

5.1.3 Inter-VNF redundancy

In this case, resiliency is achieved on NS level through redundancy of VNFs. Like described on VNFC level in clause 5.1.2, it is possible to spread the VNF instances out across servers, racks, data centres, and geographic regions. In this model, single VNFs may fail, since all services can quickly be moved to another VNF without violating resiliency expectations of the NS.

NOTE: The same holds true in case of PNFs being involved.

In case only inter-VNF redundancy is used i.e. cloud native VNFs do not implement redundancy internally by duplicating components, then they support other functional blocks, e.g. partner VNFs, load balancers, management systems, to take the necessary actions in case of failures.

These models are also discussed in ETSI GS NFV-REL 003 [i.4].

Possible mechanisms include:

- Health-check protocols e.g. between VNFs or to load balancers. The used protocols and way of decision-making for failovers need to be documented by the VNF provider.

- Storage for state data outside the VNF if necessary: Dependency on database or redundant file systems need to be documented by the VNF provider.
- As above, recovery times and risk factor need to be accurately documented by the VNF provider.

In this case the NSD needs an appropriate flavour with appropriate instantiation levels of these VNFs, in the appropriate locations to meet the SLA expectations.

5.1.4 Monitoring and failure detection

A cloud native VNF is responsible for accurate monitoring and detection of failures affecting the VNF and its components, by either supporting it in the VNF or exposing the necessary capabilities to other entities. Specifically, some of the cloud native VNFs are expected to be equipped with appropriate mechanisms to monitor and support the general operational health of the VNF. This is critical for supporting the implementation of many resiliency patterns essential to the maintenance of network service for the identification of unusual conditions that might indicate failure or the potential for failure.

Monitoring and failure detection characteristics include the following:

- Monitoring metrics: Detailed set of metrics provided by the monitoring capability that describes the state and health of the VNF;
- Error logging: The logging mechanism capture critical events at an appropriate level of detail;
- Failure detection scheme: The ability to detect appropriate levels of failures at:
  - VNF/VNFC level;
  - Intra VNF connections;
  - Inter VNF connections;
- Mechanisms to support fault analysis;
- Mechanisms for failure prediction (optional).

5.1.5 Healing

Cloud computing technology allows for several mechanisms for healing.

The redundancy mechanisms described in previous clauses already provide a certain level of healing. However, after a failure has occurred and the service has been recovered by performing some failover (at the VNFC or VNF level), the redundancy is lost (in case of active-standby) or reduced (in case of active-active or N+M). A second subsequent failure in that stage may lead to an outage.

However, for VNFs implemented in a cloud native approach the redundancy is not affected and no healing is required in the failover scenarios.

ETSI GS NFV-REL 001 [i.5] describes the two stages with remediation and recovery in detail:

- In case the cloud native VNF provides internal mechanisms for redundancy, these mechanisms should be supplemented with automated recovery actions. Typically, an automatic recovery action within the resource assigned to the VNF will be tried and the affected resource, if it cannot recover, needs to be replaced by a new resource allocated via the VIM. The resource replacement process needs to be controlled by MANO;
- In case of redundancy mechanisms outside the VNF (i.e. after a failure, another VNF takes over the service), typically an automatic repair action will be tried, and a notification is needed about success or failure of the automatic recovery. If the VNF cannot be recovered from the failure, MANO needs to terminate the VNF, free all resources and instantiate a replacement VNF.

5.1.6 Requirements

Requirement 5.1.1: Cloud native VNFs shall provide the necessary redundancy mechanisms to meet their resiliency expectations.

Requirement 5.1.2: The VNFPCD of a cloud native VNF shall support a description of redundancy characteristics and models of the VNF.

Requirement 5.1.3: The VNFPCD of cloud native VNF shall support a description of recovery times for intra-VNF redundancy or inter-VNF redundancy.

NOTE 1: Since the mean time for detection will vary for different failure scenarios and load situations, mean times for common failures is documented. In case of the external mechanisms used for the VNF, it should suffice to document the times needed for a VNF to take over a certain load (e.g. number of sessions) from a failed VNF.

Requirement 5.1.4: The VNFPCD of a cloud native VNF shall support a description of the impact of common failures on the VNF.

NOTE 2: Common failures include but are not limited to failures of instances of VNFC, VNF, internal and external links. The impact of failure will vary largely depending on the failure, the load and the instantiation level.

Requirement 5.1.5: The VNFPCD of a cloud native VNF shall support a description of any single point of failure in the VNF architecture or that may exist caused by VNF configuration for its virtual resources.

Requirement 5.1.6: Cloud native VNFs shall provide mechanisms for monitoring and failure detection to fulfil the resiliency expectations, either via supporting them directly in the VNF or by exposing this data to other entities that perform monitoring and fault detection.

NOTE 3: VNF internal mechanisms are out of scope of the present document.

Requirement 5.1.7: The VNFPCD of a cloud native VNF shall support an accurate description of fault monitoring and failure detection characteristics of the VNF.

Requirement 5.1.8: Cloud native VNFs shall be able to provide notifications about any problems discovered related to assigned resources (e.g. failing automatic recovery actions), so resource replacement can be initiated.

Requirement 5.1.9: Cloud native VNFs shall provide notifications about any outcomes related to fault healing (e.g. start and complete time of failover, and success or failing automatic recovery actions).

5.2 Scaling

5.2.1 Introduction

1) The ISO/IEC 17788 [i.6] identifies six key characteristics of cloud computing: on-demand self-service, broad network access, resource pooling, multi-tenancy, rapid elasticity and scalability, and measured service. One of the benefits expected from NFV is the ability for rapidly scaling services in and out [i.3] i.e. rapid elasticity or on-demand / automated deployment of new instances.

It is expected that the cloud native VNFs support scaling. There are different mechanisms for scaling. One of the mechanisms is to support scaling by adding/removing VNFC instances, in response to changed demand loads. Another mechanism is to add/remove VNF instances to an NS and share the load between those.

A number of VNF characteristics may affect its ability to scale out VNFCs including VNFC deployment constraints (e.g. affinity/anti-affinity rules, use of accelerators and resource requirements).

Clause 5 of ETSI NFV-SWA 001 [i.2] identifies three basic models for VNF scaling as follows:

- Auto scaling: The VNFM triggers scaling according to the rules in the VNFD;
- On-demand scaling: The VNF Instance or its EM monitor the states of the VNF Instance's constituent VNFC Instances and triggers a scaling operation through explicit request to the VNF Manager to add or remove VNFC instances (scale out or scale in, respectively) or increase or decrease resources of one or more VNFC instances (scale up or scale down, respectively);
- Scaling based on a management request: manually triggered or OSS/BSS triggered.

5.2.2 Scale-out and scale-in

One of the benefits of cloud native design patterns is the ease of scaling. It is expected that cloud native VNFs support scale-out/in. Scale-up/down is dependent on dynamically changing the performance of assigned resources and is not described here or required from cloud native VNFs:

1) Scaling triggers: As described in ETSI NFV-SWA 001 [i.2], scaling can be initiated/triggered in different ways:
   a) in case of auto-scaling by the VNFM;
   b) in case of on-demand scaling by the VNF itself;
   c) by a management request.
2) Scaling decision: The VNF exposes on which information the decision about scaling is based. Examples include number of connections, sessions, flows, subscribers, UEs.
3) Maximum Scale Level: Is there a maximum scaling level, where scale-out is limited or is the scaling only limited by resource availability?
4) Minimum Scale Level: Which is the minimum level for scale-in, i.e. the smallest possible VNF deployment?
5) Scale-out and scale-in performance (excluding the NFVI/VIM related performance):
   a) How quickly can the VNF take new VNFC instances in service when scaling out?
   b) How quickly are resources released when scaling in?
6) Granularity of scaling: Which size of scaling steps is possible, e.g. need VNFC instances be added in pairs or groups (e.g. to implement redundancy schemes or account for dependencies)?
7) Limiting resources: What are the limiting resources when scaling infinitely, when there is no Maximum Scale Level?
8) The scaling unit (e.g. VNFC or group of VNFCs) supports graceful termination during the scaling process.

The VNFPCD includes a description of the above scaling information, part of which is the necessary scaling data defined in the VNFD and consumable by MANO.

5.2.3 Scaling in different dimensions

Different load characteristics (e.g. number of connections, sessions, flows, subscribers, UEs) will affect the VNFCs differently, so in some traffic situations, only certain VNFCs need to be scaled. This implies certain requirements on the parameters described in the previous clause:

- Scaling triggers: The scaling trigger needs to specify which VNFCs will be scaled;
- Scaling decision: Depending on the type of threshold that was crossed different scaling will be triggered;
- Scaling level: There will be separate levels and granularity for the groups of VNFCs that can be scaled independently.

The cloud native VNF might expose which VNFCs or groups of VNFCs can be scaled independently including the necessary dependency on traffic characteristics. In the present document no further details are elaborated.

5.2.4 Scaling on NS level

In case of scaling of the NS by adding/removing VNF instances, the VNF needs to provide scaling support:

- Scaling decision: The VNF needs to provide load monitoring, for the decision when additional VNF instances are needed or can be stopped;
- Scaling level: The number of VNF instances sharing the load in a NS is not limited by the VNF implementation.

5.2.5 Requirements

Requirement 5.2.1: The VNFPCD shall support a description of scaling triggers (auto-scaling/on-demand/management-request) it supports.

Requirement 5.2.2: The VNFPCD shall support a description of the types of conditions that trigger scaling.

Requirement 5.2.3: The VNFPCD shall support a description of traffic/load information the VNF provides to support a scaling decision outside the VNF instance.

Requirement 5.2.4: The VNFPCD shall support a description of which groups of VNFCs can be scaled independently, including their relation to traffic/load characteristics.

Requirement 5.2.5: The VNFPCD shall support a description of the maximum and minimum scaling level, that is maximum and minimum number of VNFCs for each group that can be scaled independently.

Requirement 5.2.6: The VNFPCD shall support a description of the size of a scaling step, e.g. number of VNFCs.

Requirement 5.2.7: The VNFPCD shall support a description of the resource that causes the scaling limit.

In the present document, the scaling under different deployment flavours has not been handled.

5.3 Composition

5.3.1 Introduction

A key design principle for virtualizing services is the composition of the cloud native VNF from modular VNFCs. Composition of VNFs using NFV granular functions [i.2] enables instantiating and deploying only the essential VNFC functions as needed for the service, thereby making service delivery nimbler. It provides flexibility with deploying VNFs as needed for the service. It enables grouping functions in a common cloud data centre to minimize inter-component latency.

The term micro-service is often used in the context of composing services, though it is not commonly defined. In the context of the present document, a NFV micro-service is assumed to be a service with the smallest size possible, and the smallest possible service consists of one micro-service. In the context of NFV, such a micro-service, called NFV micro-service, is defined as an atomic service module, delivered as an all-inclusive software package, that covers a specific and coherent functional scope, is consumable over network interfaces, is managed independently from other micro-services, and runs as a computing process.

5.3.2 Cloud native VNF composition

A VNF can be a large construct and therefore when designing it, it is important to think about the components from which it will be composed. A good overview of the architecture of a VNF is provided in clause 4 and annex B of ETSI GS NFV-SWA [i.2]. When designing the components of the VNF it is important to create VNFCs with single capabilities and independence as described below.

Individual VNFCs are assumed to each provide a single capability with a clearly defined scope and functionality. If there are optional features in the VNF it could be preferred to provide them by a separated VNFC. Redundancy and independent scalability need also to be considered.

VNFCs should be designed for independence. Dependencies of VNFCs on each other should be kept to a minimum. The goal is that VNFCs can be independently deployed, configured, upgraded, scaled, monitored, etc.

One VNFC can be composed of one to many NFV micro-services.

It is possible to design a cloud native VNF in such a way that it uses NFV micro-services provided by a PaaS (Platform-as-a-Service) implementation.

Container technology is one lightweight and high performance OS-level virtualisation technology, which is expected as execution environment to support NFV micro-services in a cloud native VNF. See more details on usage of containers in clause 5.8.

Lightweight APIs expose a small set of functionalities (i.e. services). Lightweight APIs have low overhead API mechanisms and can be efficiently called from a consumer. This is beneficial for NFV micro-services based system since NFV micro-services need to interact quite often with each other to achieve a more complex function.

The VNFCs contained in the cloud native VNF ideally implement the smallest possible function.

5.3.3 Requirements

Requirement 5.3.1: The VNFPCD of a cloud native VNF shall support a description of the composition characteristics for the VNF including details on individual VNFCs and dependencies between VNFCs.

Requirement 5.3.2: The VNFCs in a cloud native VNF should communicate via lightweight APIs.

Requirement 5.3.3: A cloud native VNF may be composed of a combination of NFV Micro-Services.

5.4 VNF design for location independence

5.4.1 Introduction

In a cloud native VNF, it is expected that VNFC functionality is independent of the resource location. This provides MANO with the capability to allocate the resources in a flexible way as resources are available.

In other words, cloud native VNFs can be instantiated in any location that meets the performance, latency or regulatory requirements of the network service.

5.4.2 Location independence

In general, VNFCs can be deployed independently of the locations. However, several constraints might be needed to be taken into account and to be documented in the VNFPCD as follows:

- Some VNFCs need hardware acceleration capabilities to be available at the location where it is deployed.
- Some VNFCs depend on other VNFCs, where from a performance perspective it might be important to have an affinity between them, in the sense the deployment should be near to each other.
- For redundancy purposes anti-affinity might be needed.
- Location constraints due to functionality, e.g. content caching, or compression for transmission.
- Regulatory constraints reflected on the location.

5.4.3 Requirements

Requirement 5.4.1: A VNFC of a cloud native VNF shall be deployable independent of location as long as resource and placement constraints are met.

Requirement 5.4.2: The VNFPCD of a cloud native VNF shall support a description of the deployment constraint characteristics for the location of the VNF.

5.5 VNF state handling

5.5.1 Introduction

In most cloud environments it is preferred to have stateless applications, since the benefits of cloud computing can much easier be utilized. However, the nature of telecommunication services leads to VNF's need of keeping states, reflecting e.g. the current network usage.

It is critical for a cloud native design, how the VNF manages all the state information that is necessary for its operation. This translates to specifying the methods used for each of the application states that are handled within that VNF (or VNFC), as there may not always be one single method that applies to keeping all the different states of that VNF (or VNFC).

The data repository where state information is kept by a VNF can be modelled as a stateful VNFC being part of the same VNF, or as an external VNF.

5.5.2 State management

Parameters relevant for state management in cloud native VNFs are the following:

- For each state, the level of state required to facilitate the re-direction of traffic from one instance of a VNFC to another;
- Level of redundancy for storage of the required state(s):
    - The level of data consistency (C) across redundant copies (when available);

NOTE: Data repositories are expected to have at most two out of Consistency (C), Availability (A) and Partition Tolerance (P) in a distributed environment.

- Separation of logic and state for decoupling state management and processing. This allows for more resilient application business logic VNFCs, faster start of VNFCs, better cloud resource usage and makes it easier to scale-out;
- Remote state storage, which means that an application logic of the VNF does not require local persistent storage or local volume associated with it. This helps for alleviating state loss from an exceptional abortion or an intentional restart, especially when such stateless VNFCs run in containers;
    - the required data repository holding the externalized state may itself be a stateful VNFC in the same VNF or may be a separate VNF. Even when following the first approach (stateful VNFC in the same VNF) it is still possible to share state across different VNF instances (of the same VNF type) as soon as data replication channels are set among those stateful VNFCs (at NS level).
- Loading the required state from a data repository. A VNFC or VNF can acquire required state from a remote storage (e.g. a distributed DataBase) when requested to handle a traffic request (related to that specific state).

Stateless realization of application logic of the VNFC comes with three main penalties:

- Latency increase (hence KPI degradation);
- Footprint increase (CPU cycles required to parse/unparse and marshal/unmarshal data, etc.);
- Network bandwidth increase.

Benefits mentioned before on stateless realization (more resilient VNFCs, faster instantiation, etc.) compete with these penalties, so it is a per-kind-of-data analysis to run, which may result in different data realization patterns per kind of VNF/VNFC state:

- E.g. VNF or VNFC config data cached in each VNF or VNFC instance is retrieved from a configuration repository at VNF or VNFC instantiation time, while NF-level traffic session state would be retrieved per traffic request.

5.5.3 Requirements

Requirement 5.5.1: Cloud native VNF shall include sufficient levels of state management for facilitating traffic re-direction between VNFCs as well as sufficient levels of redundant storage for state data.

Requirement 5.5.2: Cloud native VNFCs shall externalize their state for fast restart of an VNFC instance, for easy scale-out of VNFC instances, for better cloud resource consumption and for higher resiliency.

Requirement 5.5.3: Cloud native VNF should use a data repository to store information in a way to protect from state loss when encountering exceptional or intentional restart. The storage of state information in a data repository shall provide at least same resiliency as the VNF.

Requirement 5.5.4: When appropriate for the type of states handled by the application of the VNF, the cloud native VNF should use a data repository for its state information, to alleviate state loss when encountering either exceptional or intentional restart.

Requirement 5.5.5: When appropriate for the type of state handled by the application of the VNF, cloud native VNF should access state data from a data repository to restore the service it is providing.

5.6 Published APIs

5.6.1 Introduction

VNFs need the ability to interact with each other. One way of interaction between VNFC or between VNFs is through APIs. Also, since VNFs can be defined by different organizations, the APIs to use the functionality of a VNF need to be understood by the consumer VNF.

NOTE: In the present document, the assumption is that the definition of the APIs is not in scope.

APIs enable to access key functionalities of a component and therefore the level of abstraction exposed at the API is important, but that depends on the designer of an API to decide on what makes sense to expose from a component.

Published APIs, as defined in clause 3 of the present document, can be "open" along different dimensions. Therefore, the level of openness, the organization defining them, and the level of conformance are relevant parameters. Published APIs can be defined by standard organizations, industry standards, or open source communities.

Most APIs also rely on a certain underlying information or data model. The underlying data model is important for interoperability and therefore it needs to be exposed and might need to be extensible such that new features can be integrated well enough. Anyway, any change in an API might cause severe interoperability issues, so APIs need to support a way of versioning and eventually preserve backward compatibility.

Finally, for an API, non-functional properties and characteristics of the API need to be clearly described and declared (e.g. the maximum response time).

5.6.2 Requirements

Requirement 5.6.1: The VNFPCD of a cloud native VNF shall support a description of the non-functional properties and characteristics of the APIs associated with the VNF.

Requirement 5.6.2: APIs associated with the cloud native VNF shall conform to published API characteristics.

Requirement 5.6.3: In case a published API exists from an organization such as a standards body, a de facto industry standard, or an open source community, that API should be preferred.

Requirement 5.6.4: APIs associated with the cloud native VNF should document their support for versioning in the VNFPCD.

Requirement 5.6.5: APIs associated with the cloud native VNF should document their support backward compatibility in the VNFPCD.

5.7 Management aspects of Cloud Native VNFs

5.7.1 Introduction

One of the properties of cloud native systems is to be micro-services oriented to adopt an architecture designed for cloud infrastructure.

The term "micro-services" lacks a common view across the industry. It has been used to describe a way to develop an application as a set of collaborating but independently deployable modules with the smallest size possible. This also means cloud native designed software systems use a loose-coupling between the NFV micro-services in order to reduce dependencies to a minimum.

Although this described concept of NFV micro-services present serious drawbacks in developing complex systems for instance regarding the size of the units, the number and the transactions between them, some common beneficial characteristics have been identified when developing services/applications as a set of independent modules:

- Automated deployment;
- Independent deployment and management (from other services);
- Fast deployment times by using technologies like containers (containers can be OS-containers or higher-level containers as defined in ETSI GS NFV-EVE 004 [i.7]);
- Enable elasticity, i.e. to be able to scale, in or out, independently of other services in the same application;
- Reduce side effects across collaborating services, e.g. scaling only portions of an application that need the additional capacity and being able to update/upgrade portions of the application without impacting users. Also, the design should be loosely-coupled to reduce dependency and such architecture makes it easier to use continuous integration and continuous delivery approaches;
- Improve fault isolation: larger applications can remain largely unaffected by the failure of a single module;
- Cloud native VNFs are able to support software modification process without impacts to service continuity, see ETSI GS NFV-REL 003 [i.4] and ETSI GS NFV-REL 006 [1].

Balancing the benefits and drawbacks of the micro-services-oriented architectures, 5G system has been architected as a framework in which network functions are componentized with each component offering specific capability. This framework is designated as Service Based Architecture and has been applied to the control plane of the 3GPP Core Network for 5G [i.10] where the different components interact with each other over specified interfaces [i.11].

Considering the state of the art, the next clauses specify the requirements for cloud native VNFs using a service approach, for taking advantage of the beneficial characteristics of using services for building applications whilst limiting the drawbacks that can arise for telecom applications.

5.7.2 Requirements

Requirement 5.7.1: If a VNF is designed using a micro-services oriented architecture, the NFV micro-services in a VNF shall be deployable and testable independently of each other.

Requirement 5.7.2: If a VNF is designed using a micro-services oriented architecture, each VNFC in a VNF shall be managed independently of other VNFCs in the VNF, i.e. initiated, terminated, scaled out/in, upgraded, updated and healed.

Requirement 5.7.3: The VNFPCD shall document cloud native VNF's support for software modification especially with respect to service continuity, allowing old and new versions to run simultaneously (as explained in ETSI GS NFV-REL 006 [1]).

5.8 Use of containers

5.8.1 Introduction

In cloud native environments, OS-containers is a preferred technology for VNFCs, whole VNFs or also subunits of a VNFC. See ETSI GS NFV EVE 004 [i.7] for the description of different container technologies including OS-container and higher-level containers (ETSI GS NFV-EVE 004 [i.7]).

NFV micros-services would ideally run in containers due to the lightweight characteristics of containers. Depending on the functionality implemented some containers technologies are better suited than others. For example, data plane near functions might be better suited using OS-containers on the other hand control and management related functionalities might be better suited with application containers.

Some cloud native VNFs can be provided as single container images and that way be very easily and fast deployed. In other cases, the cloud native VNF can be structured as a group of containers via its VNFCs, where the cloud native VNF may use a container infrastructure service or manage the containers itself.

5.8.2 Requirements

Requirement 5.8.1: The VNFC of a VNF should be capable to run in an OS-container (as defined in ETSI GS NFV EVE 004 [i.7]).

5.9 Zero-touch Management

5.9.1 Introduction

The complementary goal of cloud native designs is zero-touch management, i.e. the cloud native VNF should be automatically configured or self-configured where possible. There are several approaches to do that and several components in scope of NFV might be affected through that. This includes the zero-touch installation/configuration and instantiation, self-healing, self-optimizing features including automated resource management.

For healing please refer to clause 5.1.5 on resiliency.

5.9.2 Automated configuration

When a new instance of a VNF or VNFC is instantiated, it needs a certain degree of configuration. Among the several ways to achieve that, two of the major options are:

1) The VNF/VNFC is configured from the outside by a management entity like an EM or MANO component.

2) The VNF/VNFC fetches the required configuration from a configuration store and has internal provisions to do all other types of configuration itself.

NOTE: Hybrid approaches mixing 1) and 2) are possible as well.

5.9.3 Automated resource management

ETSI GS NFV-IFA 010 [2] (e.g. clause 5.1) describes in detail how automated resource management is supported by the functions of NFV-MANO.

Resource management on the virtualization layer can also be done through:

1) detailed management and control of resources assigned to VNFs and eventually in the future to VNFCs/NFV micro-services; or

2) by a resource management relying on increasing the underlying resources early enough to basically have always enough resource available for consumption by the VNFs/VNFCs/Micro Components. The underlying resources can be on-premises physical or virtual or off-premises physical or virtual resources.

The information about the expected resource management model used by a VNF is exposed to the MANO functions.

For the separation of infrastructure resource management and VNF resource management, the two are logically decomposed and are handled in separate management components or through an infrastructure planning process. Since NFV infrastructure resource are bound by available physical or virtual reserved resources it is important to understand the growth rate of resource usage in the various resource categories. Therefore, VNF information about the growth rate and based on what unit is an important piece of information for the management and planning process of the physical resource deployments.

NOTE: The present document does not elaborate on the dependency between different resource management. The infrastructure planning process is outside the scope of the present document.

The information about the physical resources and the dynamic scaling capabilities required by a VNF (see also clause 5.2) are exposed in the VNFPCD and provided to the MANO functions as static information like a growth functions of different input parameters, or a dynamic prediction of future resource requirements exposed by the VNF.

As described in ETSI GS NFV-IFA 010 [2] clause 5.1, resource management will consider quota and permitted allowance, priorities, resource reservation, etc. and allow for licensing (see ETSI GR NFV-EVE 010 [i.8]) and charging (ETSI GR NFV-EVE 008 [i.9]) for the dynamic resource management.

5.9.4 Requirements

Requirement 5.9.1: The VNFPCD of a cloud native VNF shall support a description of supported options for automated configuration used by the VNF.

NOTE: For details about automated configuration, see clause 5.9.2.

Requirement 5.9.2: The VNFPCD of a cloud native VNF shall support a description of the supported types of configuration store including the access method in case the VNF uses a fetch approach from a configuration store.

Requirement 5.9.3: The VNFPCD of a cloud native VNF shall support a description of the resource management models (see clause 5.9.3) used by the VNF.

Requirement 5.9.4: The information about the dynamic scaling capabilities shall be provided in the VNFPCD and to the MANO functions.

Requirement 5.9.5: In case the VNF triggers resource management, quota management, permitted allowance, other resource management constraints (e.g. priorities, resource reservation), licensing and charging shall be addressed in the same way as when MANO triggers resource management.

5.10 Load-balancing

5.10.1 Introduction

Cloud native VNFs typically are working in a highly distributed and scaled system. So, there will be load balancer systems established for distributing the traffic between VNFs or between components (VNFCs) inside a VNF instance.

Typically load balancers need to be adjusted during LCM operations, failure handling/healing (see clause 5.1), scaling operations (see clause 5.2), etc.

Different types of load balancing are described in ETSI GS NFV-SWA 001 [i.2] in clause 5.1.4.

Load balancing needs to be managed on multiple levels/domains hence knowledge of the precise VNF needs is paramount for their optimal use:

- VNF-internal load balancer.
- VNF-external load balancer.
- End-to-end load balancing.
- Infrastructure network load balancer.

5.10.2 Requirements

Requirement 5.10.1: The VNFPCD of a cloud native VNF shall support a description of the external load balancer it needs.

Requirement 5.10.2: The VNFPCD of a cloud native VNF shall support a description whether the VNF requires access to NFVI switches (e.g. DOPFR-like approach) for the case of load balancer being external to the VNF.

6 Classification of cloud native VNF implementations

6.1 Introduction

Clause 5 described a representative set of functionalities and associated requirements that a cloud native VNF is expected to provide. Based on these functionalities, the classifications of the cloud native VNF characteristics are provided in clause 6.2.

These classifications are critical for network operators as they need to take informed decisions on the acquisition of cloud native VNFs and subsequent integration into their networks. The result is a "packaging" of the cloud native VNFs that contains these classifications as part of their life cycle management. The cloud native VNF packaging is described in clause 6.3.

6.2 Cloud native VNF characteristics

6.2.1 Cloud native VNF characteristics and their classifications

The cloud native VNF requirements developed in clause 5 generate a set of characteristics that is summarized in this clause along with their classifications.

Table 1: Cloud native VNF Characteristics and Their Classifications

VNF Classification	Characteristics	Requirement
Resiliency - Redundancy	Redundancy Model	5.1.1, 5.1.2
	Mode of Recovery	5.1.1, 5.1.2
	Performance Metrics	5.1.1, 5.1.2
	Recovery Time	5.1.3
	Single Point of Failure	5.1.5
	Risk Factor	5.1.4
Resiliency - Fault Monitoring and Failure Detection	Fault Monitoring Metrics	5.1.6, 5.1.7
	Fault Logging Configuration	5.1.8, 5.1.9
	Failure Detection Scheme	5.1.7, 5.1.8
	Mean Time to Identify Failure	5.1.6, 5.1.7
Scaling - In/Out	Scaling Triggers	5.2.1, 5.2.2
	Scaling Influencing Factor	5.2.3, 5.2.4
	Maximum/minimum Scale Level	5.2.5
	Scaling Frequency	5.2.6
	Granularity of Scaling	5.2.5, 5.2.7
	Limiting Resources	5.2.7
	Scale-In Resource Free-Up	5.2.1

VNF Classification	Characteristics	Requirement
Design - Decomposition	Composition of VNF	5.3.1, 5.3.3
	Independence of VNFC	5.3.1, 5.3.3, 5.4.3
	State Management	5.5.1, 5.5.2, 5.5.3, 5.5.4, 5.5.5
	API Usage	5.3.2, 5.6.1, 5.6.2, 5.6.3, 5.6.4, 5.6.5
VNF Architecture - Use of Services	VNF Atomic Unit	5.7.1, 5.7.2
	VNF Atomic Unit Deployment and Testing	5.7.1, 5.7.2, 5.7.3
	VNF Atomic Unit Independence	5.4.1, 5.7.2, 5.7.3
	VNF Automation	5.9.1, 5.7.2, 5.9.3
	VNF Communication - Use of Lightweight APIs	5.3.2
Zero-Touch Management - Automated Instantiation and Configuration	VNF Configuration Method	5.9.1, 5.9.2, 5.9.3, 5.9.4, 5.9.5
Zero-Touch Management - Load Balancing	Internal Load Balancer	5.10.1, 5.10.2
	External Load Balancer	5.10.1, 5.10.2
	End-to-End Load Balancer	5.10.1, 5.10.2
	Infrastructure Network Load Balancer	5.10.1, 5.10.2
Zero-Touch Management - Automated Resource Management	Detailed Management and Control	5.9.3, 5.9.5

The resulting classification of VNFs into a specified set of classes based on values for the particular parameters listed is out of scope of the present document.

6.3 Cloud native VNF Product Characteristic Descriptor

The VNF requirements and characteristics of cloud native VNF described in the present document serve as a guide by which operators can determine how best to select them for their networks. The characteristics for the cloud native VNFs are captured in the VNFPCD.

Requirement 6.3.1: Cloud native VNF characteristics developed in the present document shall be included in the VNFPCD relating all software elements, configuration information and transition profiles (VNF traffic load limits necessary to trigger scaling up or down) which enable the VNF functionality.

6.4 Cloud native VNF Package

6.4.1 Cloud native VNF capacity profile

Since the cloud infrastructure needs some information for planning capacity and eventually for scheduling the various VNFCs a capacity profile is envisioned to be part of the VNF Package. The VNF capacity profile instructs the cloud infrastructure on determining demand (expected input traffic load) and the (scaling operation) instructions it executes at specific demand (trigger) levels. The cloud native VNF capacity profile includes identification of the input traffic interface and the load metric (packets/transaction/sec, etc.) and example trigger policies on some standard infrastructure configuration. The capacity profile is given in terms of the overall VNF and in terms of the VNFCs that compose each VNF. See clause 5.9 on resource management and clause 5.2 on scaling for a variety of capacity-oriented parameters.

6.4.2 Cloud native VNF operational profile

The operational profile includes parameter for operation task like the initial provisioning, recovery parameters, and re-location oriented parameters.

6.4.3 Requirements

Requirement 6.4.1: A cloud native VNF capacity profile shall be part of the VNF Package.

Requirement 6.4.2: A cloud native VNF capacity profile shall contain demand in terms of expected traffic load, the scaling operation at specific demand thresholds.

Requirement 6.4.3: A cloud native VNF operational profile shall be part of the VNF package to support initial provisioning, elastic re-provisioning, relocation re-provisioning and recovery provisioning.

Requirement 6.4.4: The cloud native VNF operational profile shall provide the identification of the runtime parameters reconfigurable in the operations for initial provisioning, elastic re-provisioning, relocation re-provisioning and recovery provisioning, on a per VNFC basis.

Requirement 6.4.5: The VNFPCD shall be included in the VNF Package of a cloud native VNF.

Annex A (informative): Authors & contributors

The following people have contributed to the present document:

Rapporteur:
Marcus Brunner, Swisscom

Other contributors:
Percy Tarapore, AT&T

Steven Wright, AT&T

Susana Sabater, Vodafone

Cecilia Corbi, Telecom Italia

Lei Zhu, Huawei

Aijuan Feng, Huawei

Ulrich Kleber, Huawei

Lei Wang, Huawei

Jie Miao, China Unicom

Gang He, China Unicom

Gongying Gao, China Unicom

Cristina Badulescu, Ericsson

Olivier, Le Grand, Orange

Annex B (informative): Bibliography

- ETSI GS NFV 002 (V1.2.1): "Network Functions Virtualisation (NFV); Architectural Framework".
- ETSI GS NFV-IFA 007 (V3.1.1): "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification".

Annex C (informative): Change History

Date	Version	Information about changes
March 2017	0.0.1	Skeleton
May 2017	0.0.2	Addition of contributions NFV EVE(17)000090r3, NFVEVE(17)000103r4, FVEVE(17)000119r1
June 6, 2017	0.0.3	Addition of contribution NFVEVE(17)000123r2
June 15, 2017	0.0.4	Addition of contribution NFVEVE(17)000111 and NFVEVE(17)000115r2
Sept 14, 2017	0.0.5	Fixed paragraph numbering for some paragraphs, Addition of NFVEVE(17)000234r1_EVE011___Initial_Draft_Text_for_clause_6.docx, Addition of NFVEVE(17)000227r1 and the according reference Addition of NFVEVE(17)000179r2
Nov 30, 2017	0.0.6	Addition of NFVEVE(17)000275_EVE011_Services_on-demand_deployment Plenary NFV#20 Approved Title Change ETSI GS EVE011 Network Functions Virtualisation (NFV) Release 3; Software ArchitectureVirtualised Network Function; Specification of the Classification of Cloud Native VNF implementations
Dec 15, 2017	0.0.7	Implementation of contributions: NFVEVE(17)000312r4_EVE011_Clause_5_2_Improvements NFVEVE(17)000313r2_EVE011_Clause_5_1_Improvements NFVEVE(17)000314r2_EVE011_Clause_5_3_Improvements NFVEVE(17)000316r2_EVE011_Clause_5_4_Improvements NFVEVE(17)000320r2_EVE011_Clause_5_5_Improvements NFVEVE(17)000321r2_EVE011_Add_Self_Healing_to_clause_5_1 NFVEVE(17)000322r1_EVE011_Clause_6_3_Cloud_Native_VNF_Product_Characteristic_De NFVEVE(17)000327r2_EVE011_Changes_to_clauses_1-4 editor action: fixing the reference i.11
Dec 22, 2017	0.0.8	Implementation of contributions NFVEVE(17)000333_EVE011_Clause_5_1_Introduction_merged_proposals NFVEVE(17)000334_EVE011_Further_improvements_clause_5_1 Removed reference i.12 (same document references twice (updated the reference in the text)
Feb 14, 2018	0.0.9	Implementation of contributions from EVE Meeting #80 NFVEVE(18)000009r1_EVE011_Changes_to_clause_5_1_2_AND_5_1_3 NFVEVE(18)000011r1_EVE011_Changes_to_clause_5_7 NFVEVE(18)000012_EVE011_Changes_to_clause_5_8
March 15, 2018	0.0.10	Implementation of contributions from EVE Meeting #81 (f2f) NFVEVE(18)000018_EVE011_some_small_editorial_fixes NFVEVE(18)000022r3_EVE011_Handling_Editors_Notes NFVEVE(18)000023r1_EVE011_Changes_to_clause_6_2 NFVEVE(18)000025r2_EVE011_Clause_5_4_VNF_design_for_location_independence NFVEVE(18)000026r2_EVE011_Clause_5_1_Downtime_of_MANO_services_ NFVEVE(18)000027r1_EVE011_Clause_5_7
April 13, 2018	0.0.11	Implementation of contributions approved from EVE Meeting #84 NFVEVE(18)000038r3_EVE011_MS_support_for_software_modification NFVEVE(18)000039r1_EVE011_support_graceful_termination NFVEVE(18)000040r1_EVE011_SLA_declaration_of_Open_API NFVEVE(18)000043_EVE011_Clause_5_9_Resource_Management_Clarifications
April 27, 2018	0.0.12	Implementation of contributions approved from EVE Meeting #85 NFVEVE(18)000049_EVE011_Clause_5_2_3_changes NFVEVE(18)000050r1_EVE011_Clause_5_2_5_changes NFVEVE(18)000051_EVE011_Clause_5_6_changes NFVEVE(18)000052_EVE011_Changes_to_clause_5_7 NFVEVE(18)000053_EVE011_Changes_to_clause_5_8 NFVEVE(18)000054r1_EVE011_Changes_to_clause_5_9_3
May 24, 2018	0.0.13	Implementation of contributions approved from EVE Meeting #86 NFVEVE(18)000065_EVE011_Proposed_raporteurs_actions NFVEVE(18)000059_EVE011_Changes_to_clause_5_5 NFVEVE(18)000060r1_EVE011_Changes_to_clause_5_3_and_5_7 NFVEVE(18)000061_EVE011_Clause_3_3_Abbreviations NFVEVE(18)000067r2_EVE011_proposed_normative_references

Date	Version	Information about changes
		NFVEVE(18)000068r1_EVE011_Add_a_resiliency_requirement_to_clause_5_1_6 NFVEVE(18)000069r1_EVE011_Add_a_deployment_constraint_requirement_to_clause_5_4 NFVEVE(18)000070_EVE011_changes_to_clause_5_2_5_and_5_3_1 NFVEVE(18)000074r2_EVE011_Changes_to_clause_5_6
Jun 1, 2018	0.0.14	Changes as agreed with NFVEVE(18)000079r1_EVE011_Proposed_Draft_0_0_14_cm
June 22, 2018	0.0.15	Implementation of contributions approved from EVE Meeting #87 NFVEVE(18)000078_EVE011_Open_API_definition NFVEVE(18)000083r1_EVE011_clause_6_3_the_creation_of_a_VNFPCD Implementation of contributions approved from EVE Meeting #88 NFVEVE(18)000086_EVE011_Changes_to_Requirements_5_3_4_and_5_7_1 NFVEVE(18)000092r1_EVE011_Add_requirements_to_state_handling
June 22, 2018	0.0.16	Implementation of changes agreed in EVE Meeting #88 of contribution NFVEVE(18)000088r1_EVE011_Proposed_updates_to_requirements Not implemented Requirement 5.9.5 (was not sure what the minutes meant
July 12, 2018	0.0.17	Implementation of changes agreed in EVE Meeting #90 of contribution NFVEVE(18)000098_EVE011_Clarify_Requirement_5_9_1 NFVEVE(18)000099_EVE011_Editorial_Changes NFVEVE(18)000100_EVE011_remote_storage_definition
August 24, 2018	0.0.18	Implementation of changes in the following contributions NFVEVE(18)000111r1_EVE011_review___use_of_micro-service NFVEVE(18)000112r2_EVE011_review___changes_related_to_VNFPCD • Changes as agreed and noted in EVE #91 Meeting Report • Editor's note in 5.5.2 should have been resolved. • In 5.9.6 there is an unresolved reference to 5.2.x • The requirement table needs to be updated
August 31, 2018	0.0.19	Implementation of NFVEVE(18)000110r2_Company_review_EVE011_Ericsson_cmmts And minutes of EVE#93 Meeting Report • Clause 5.1 should be reviewed by REL. • 5.2.2 , 5.2.5 and 5.5.3 should be revised to address comments. Need to add an editor's note pointing to this contribution. • Need to replace microservices by "NFV Micro-services", replace "remote storage" with "data repository". • Check that 6.3.1 is not already present in another form (shall in 6.3.2 from the new draft). Already approved as a shall. • Approved with editor's notes for 5.2.2 , 5.2.5 and 5.5.3. • Possible conflicts with EVE(18)0000113 should be identified by the rapporteur. • Revisions of 5.2.2 , 5.2.5 and 5.5.3 should pay attention to EVE(18)0000113 Implementation of NFVEVE(18)000113_EVE011_review___Orange_comments-2 And minutes of EVE#93 Meeting Report • "Single point of failure" in 5.1.2 may need a corresponding requirement. • 5.1.3 "appropriate location" is unclear and should be precised. • Last sentence in 5.2.1 and 5.2.2 should be deleted. • 5.2.5 will be revised, but editorial comments stand. • Last sentence in 5.3.2 "Lightweight APIs" should be clarified. • Need to delete Requirement 5.3.5, as it comes from IFA029. • Req 5.4.2 should be either precised or deleted. • 5.9.1 should be reworded. • First sentence of 6.3 should be reworded. • Should fix reference in last sentence of 6.4.1
Sept 20, 2018	0.0.20	1. NFVEVE(18)000119r1_EVE 011 Proposed Draft 0.0.19 (notes) a. References, need to remove IFA 029 which is not approved. (MB: i.3 is named "void") b. 2.1 remove editor's note, do not mention version numbers in the reference. Add "Note: it is recommended to use the latest versions of the following documents". Need to switch NFV003 from normative to informative references. (MB: done, ref 2 is void) c. Data repository is not defined in NFV 003 nor in the definitions clause, need a contribution mentioning persistence and the option of being remote. In 5.5.2 remote storage is still mentioned, mentioning data repository would be more appropriate. Remote storage definition is no longer required. (address) (MB: addressed through implementation of NFVEVE(18)000127 EVE011_Cloud_native_VNF_definition)NFVEVE(18)000129_EVE011_Data_repository_VNF_definition d. 3.1 issues addressed, remove the editor's notes. (MB: done)

Date	Version	Information about changes
		e. Still missing a definition for cloud native VNF. (see next document), MB: addressed through implementation of f. Need to copy the "NFV micro-services" definition from IFA029. (MB: that is already in the definition clause, no action taken) g. 5.1 was addressed, remove editor's note h. 5.2.2 remove editor's note. Replace all numbered list by "The VNFPCD includes a description of the scaling information otherwise present in the VNFD". (MB: ???) i. 5.2.5 in requirements change "'The VNF provider shall convey" to "The VNFPCD shall support a description of". (MB: done) j. 5.3.2 4th paragraph, remove "will", remove "e.g. VNF Common…". (MB: done) k. Req 5.3.3 replace with a requirement on composition of component, "… composed of NFV Micro-Services". Remove editor's note. (MB: done) l. 5.5 remove editor's note and put a forward reference "VNF state handling is addressed in clause 5.5" in the resiliency clause. (MB: done) m. 5.5.3 remove editor's note as it was already addressed. (MB: done) n. 5.6 title is misleading, change to "Published APIs", remove heading 5.6.2 placing its content at the end of the 5.6.1 introduction. (MB: done) o. 5.6.1 NOTE, remove text after "not in scope" (MB: done) 5.7 editor's note need to study the redundancy between 5.3 and 5.7 and remove redundancies. No overlap was found and editor's note can be removed. Rename 5.7 p. "Management aspects of Cloud Native VNFs". Remove req 5.7.1 as it is redundant. (MB: done) q. 5.9 change "ultimate" to "complementary", replace "e.g." with "i.e.", remove editor's note. (MB: done) r. 6.2 remove the requirement clause and the "requirements" in the title (MB: done, ??? was it meant to have everything removed?) s. 6.2.1 check numbering and remove editor's note. t. Delete clause 6.2.2 including the editor's note. u. 6.3 Remove first sentence which is unclear. Replace "developed". with "described". Replaced "blueprint" with "guide", replace last sentence (which is in the requirements) with "The characteristics for the cloud native VNFs are captured in the VNFPCD". Remove editor's note. Second editor's note already addressed. (MB: done) v. EVE+REL Review: – 5.1.1 replace first "VNFC" with "VNF/VNFC". Remove "as well as the use of accelerators". Add "such as" after "or parts of it". (MB: done) – 5.1.2 remove first editor's note, replace "failover time" with "recovery time". Replace "vnfc service" with "service provided by the VNFC" Remove second editor's note. (MB: done) – 5.1.3 replace "failover time" with "recovery time". remove two editor's notes(MB: done) – 5.1.4 remove "and faults", and "mean time to detect failures" bullet. (MB: done) – 5.1.5 remove "NFV micro-service" – 5.1.6 req 5.1.3 replace "failover" with "recovery", Note 1 add "mean time for detection". Req 5.1.4 replace ":" with "."(MB: done) – 5.5.2 replace "loading… from a remote storage" by "Loading … from a data repository". Replace "deparse" with "unparse", add "marshall/unmarshall". (MB: done) – 5.5.3 req 5.5.3 add "data repository" to "storage of state information" (MB: done) 2. Implementation of NFVEVE(18)000127 EVE011_Cloud_native_VNF_definition a. Where cloud native principles can be found would be helpful (cncf ?). NFV's view of cloud native can be different from others, i.e. zero-touch management is expected to be part of it. (MB: There is contribution NFVIFA(18)000811r1_IFA029_annex_A_4_Add_Information_of_CNCF_projects which has the sort of a CNCF definition with properties like computing paradigm that is optimized for modern distributed systems environments capable of scaling to tens of thousands of self-healing multi-tenant nodes, Container packaged, Dynamically managed, Micro-services oriented) b. Approve r1 but that cloud native principles need to be clarified. Use uncapitalized letters for cloud native. (MB: the only definition from CNCF is available, and to a certain degree cloud native is defined) 3. Implementation of NFVEVE(18)000129 EVE011_Data_repository_definition 4. From f2f meeting discussion a. 5.2.2 on scaling, keep the numbered list, but change bottom sentence to "The VNFPCD includes a description of the above scaling information, part of which is the necessary scaling data defined in the VNFD and consumable by MANO."

Date	Version	Information about changes
		b. all text black, check the requirement number in table 1 for "zero touch management" a comma is missing. Remove empty lines in table.

History

Document history		
V3.1.1	October 2018	Publication