ETSI GS NFV-IFA 010 V4.2.1 (2021-05)

Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Functional requirements specification

Disclaimer

The present document has been produced and approved by the Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
RGS/NFV-IFA010ed421

Keywords
functional, management, MANO, NFV,
orchestration, requirements, virtualisation

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

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 functional requirements for NFV management and orchestration, and general guidelines and requirements for NFV management and orchestration interface design.

The scope of the present document does not cover the functional requirements on interfaces.

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.

Not applicable.

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 GS NFV 002: "Network Functions Virtualisation (NFV); Architectural Framework".

[i.2] ETSI GR NFV 003: "Network Functions Virtualisation (NFV); Terminology for main concepts in NFV".

[i.3] ETSI GS NFV 004: "Network Functions Virtualisation (NFV); Virtualisation Requirements".

[i.4] Void.

[i.5] Void.

[i.6] Void.

[i.7] Void.

[i.8] ETSI GS NFV-PER 001: "Network Functions Virtualisation (NFV); NFV Performance & Portability Best Practises".

[i.9] ETSI GR NFV-IFA 023: "Network Functions Virtualisation (NFV); Management and Orchestration; Report on Policy Management in Mano; Release 3".

[i.10] ETSI GR NFV-TST 005: "Network Functions Virtualisation (NFV); Continuous Development and Integration; Report on use cases and recommendations for VNF Snapshot".

[i.11] ETSI GR NFV-IFA 022: "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on Management and Connectivity for Multi-Site Services".

[i.12] ETSI GR NFV-EVE 012 (V3.1.1): "Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework".

[i.13] ETSI GS NFV-IFA 013: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Os-Ma-nfvo reference point - Interface and Information Model Specification".

[i.14] ETSI GS NFV-IFA 005: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification".

[i.15] ETSI GS NFV-IFA 007: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification".

[i.16] ETSI GS NFV-IFA 008: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specification".

[i.17] ETSI GS NFV-IFA 014: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Network Service Templates Specification".

[i.18] ETSI GR NFV 001: "Network Functions Virtualisation (NFV); Use Cases".

[i.19] ETSI GS NFV-IFA 011: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; VNF Descriptor and Packaging Specification".

[i.20] ETSI GS NFV-REL 006: "Network Functions Virtualisation (NFV) Release 3; Reliability; Maintaining Service Availability and Continuity Upon Software Modification".

[i.21] ETSI GS NFV 006: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Architectural Framework Specification".

[i.22] ETSI GS NFV-IFA 040: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Requirements for service interfaces and object model for OS container management and orchestration specification".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GR NFV 003 [i.2] and the following apply:

NOTE: A term defined in the present document takes precedence over the definition of the same term, if any, in ETSI GR NFV 003 [i.2].

compute host: whole server entity, part of an NFVI, composed of a HW platform (processor, memory, I/O devices, internal disk) and a hypervisor running on it

NOTE: This definition is from ETSI GS NFV-PER 001 [i.8].

containerized workload: VNF or VNF component designed to be deployed and managed on Container Infrastructure Service (CIS) instances

NOTE: This definition is from ETSI GS NFV-IFA 040 [i.22].

Network Service (NS) healing: procedure that includes all virtualisation related corrective actions to repair a faulty Network Service (NS) instance including components/functionalities which make up the instance, and have been associated with this fault situation

NOTE 1: In a virtualised environment network service healing focuses only on the virtualised components/functionalities. In case of an NS consisting of virtualised and non-virtualised parts, a procedure able to handle both parts is needed. This will be done in connection with components/functionalities that are located outside the virtualised environment.

NOTE 2: "Virtualisation related corrective actions" refers to action(s) toward virtualised resource(s) and associated NS instance.

Operating System (OS) Container: virtualisation container utilizing a shared Operating System (OS) kernel of its host

NOTE 1: The host providing the shared OS kernel can be a physical compute node or another virtualisation container.

NOTE 2: This definition is from ETSI GS NFV-IFA 040 [i.22].

service availability level: information provided to assist in the selection of virtualised resources to be allocated for the NS constituents in terms of availability

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GR NFV 003 [i.2] and the following apply:

CIR	Container Image Registry
FPGA	Field Programmable Gate Array
NUMA	Non Uniform Memory Access
PCIe	Peripheral Component Interface express
URI	Uniform Resource Identifier

4 General Description

4.1 Introduction

Network Functions Virtualisation (NFV) adds new capabilities to communications networks and requires a new set of management and orchestration functions to be added to the current model of operations, administration, maintenance and provisioning. The NFV Management and Orchestration (NFV-MANO) architectural framework has the role to manage the infrastructure and orchestrate the resources needed by the Network Services (NSs) and Virtualised Network Functions (VNFs).

In order to guide the development of the specification of the interfaces exposed between the NFV-MANO Functional Blocks (FBs), it is important to have a clear and consolidated set of functional requirements to be addressed by the NFV-MANO. The present document is providing functional requirements on NFV-MANO e.g. VNF Lifecycle Management (LCM), NS LCM, virtualised resource management, etc.

4.2 Overview

In order to provide systematic functional requirements, the present document arranges the functional requirements by categorizing the requirements according to key operational functions of NFV-MANO, which are documented in ETSI GS NFV 006 [i.21].

Key operational function categories which are used to organize the requirements on NFV Orchestrator (NFVO), VNF Manager (VNFM) and Virtualised Infrastructure Manager (VIM) in the present document are listed below:

- Virtualised resource management.
- VNF LCM.
- NS LCM.

- VNF information management.
- NS information management.
- NFV performance management.
- NFV fault management.
- Security considerations.
- Software image management.
- NFV acceleration management.
- Multi-tenancy.

NOTE: This categorization groups related functional requirements together. Actual interface requirements derived from the functional requirements may be grouped differently, and/or individual interface requirements may be placed into a group that is different from the category of the related functional requirement.

5 General functional requirements

5.1 General functional requirements for virtualised resource management

The NFV-MANO architecture shall provide support to permit service providers to partially or fully virtualise the Network Functions (NFs) needed to create, deploy and operate the services they provide. In case of partial virtualisation, performance, management and operations of the non-virtualised NFs shall not be impacted.

The NFV-MANO architecture shall enable support for network slicing according to operator policies and SLAs, see clause 5.5.

The NFV-MANO architecture shall be able to support an NS composed of Physical Network Functions (PNFs) and VNFs implemented across multivendor environments.

The NFV-MANO architecture shall be able to manage NFV Infrastructure (NFVI) resources, in order to provide NSs and related VNFs and PNFs with the resources needed. Management of resources for PNFs shall be restricted to provisioning connectivity, e.g. necessary when an NS instance includes a PNF that needs to connect to a VNF.

The NFV-MANO architecture shall enable the NFVO and the VNFM to manage the virtualised resources needed for LCM of the VNFs. The NFV-MANO architecture shall enable deployments and implementations where:

- the NFVO is the only FB to manage the virtualised resources needed for the LCM of the VNF (**VNF-related Resource Management in indirect mode)**;
- the VNFM is the only FB to manage the virtualised resources needed for the LCM of the VNF **(VNF-related Resource Management in direct mode)**;
- the NFVO and the VNFM, both, manage the virtualised resources needed for the LCM of the VNF.

NOTE: This is a decision per VNFM whether it is the NFVO or the VNFM that manages the virtualised resources.

It is a deployment and implementation decision whether one option or both are deployed and implemented. All VNFs managed by one VNFM shall use the same option for virtualised resource management. The detailed requirements on the NFVO and the VNFM for each case are depicted in clauses 6.1 and 7.1.

In addition to managing the VNF-related virtualised resources as explained above, the NFV-MANO architecture shall enable the NFVO to manage the virtualised resources (i.e. network resources) that are needed for LCM of the NS(s).

Additionally, the NFV-MANO shall enable different models, per resource type, to facilitate availability of resources and to avoid resource contention. It shall be possible for the network operator, on a per NS basis, tenant basis or VNF basis, to select one of the following resource commitment models, or a combination of them:

- **Reservation** model, where resources are committed, but not allocated, to a particular consumer or consumer type. A reservation can have one of the following types (see details in clause A.2.8):

  1) reserving a set of resources considering particular virtualised resource configurations, i.e. reserving a number of virtualised containers, virtual networks, network ports and/or storage volumes;

  2) reserving virtualised resource capacity without considering particular resource configurations, i.e. reserving virtualised resource capacity of compute, storage and network resource types.

- **Quota/Allowance based** model, where the number of resources to be consumed by a particular consumer is limited to a defined amount or a percentage of resources; in this model, resources are committed upon demand from the consumer when a VNF or an NS is instantiated or scaled out, as long as those are within the limits established by the quota/allowance for that consumer or consumer type.

- **On demand**, where resources are committed when a VNF or an NS is instantiated or scaled out, as long as there are available resources for consumption.

NFV-MANO shall be able to manage resources (service resources and infrastructure resources) taking in account priorities based on operator policies and SLAs.

The permitted allowance concept should be distinguished from the quota concept:

- Quota: enforced by the VIM. Quotas are usually used to prevent excessive resource consumption in the VIM by a given consumer.

- Permitted allowance: maintained at NFVO level. Permitted allowances might vary in granularity (VNFM, VNF, group of VNFs, NS, etc.) and are used to control resource consumption by VNFMs in relation to the granularity associated with the permitted allowance.

The detailed requirements on the affected FBs are depicted in clauses 6.1, 7.1 and 8.2.

5.2 General functional requirements for multi-tenancy

Multi-tenancy can be applied to all infrastructure and service resources which can be consumed from an NFV system and managed by NFV-MANO. NFV provides isolation between the infrastructure resources and/or isolation between the service resources allocated to different tenants. As described in ETSI GR NFV 001 [i.18], clause 6.6, the NFV infrastructure is responsible for providing appropriate isolation. NFV-MANO shall provide the necessary information to the NFVI to allow the appropriate isolation.

NOTE 1: The term "resource" as used in the present clause goes beyond the definition of NFV-Resource as specified in the NFV Terminology document (ETSI GR NFV 003 [i.2]).

NOTE 2: NFV-MANO provides some capabilities to achieve such isolation, e.g. anti-affinity rules, resource-zones, etc. It is up to the Consumer to make proper use of these capabilities.

Figure 5.2-1 shows the entities relevant to multi-tenancy for any kind of resources.

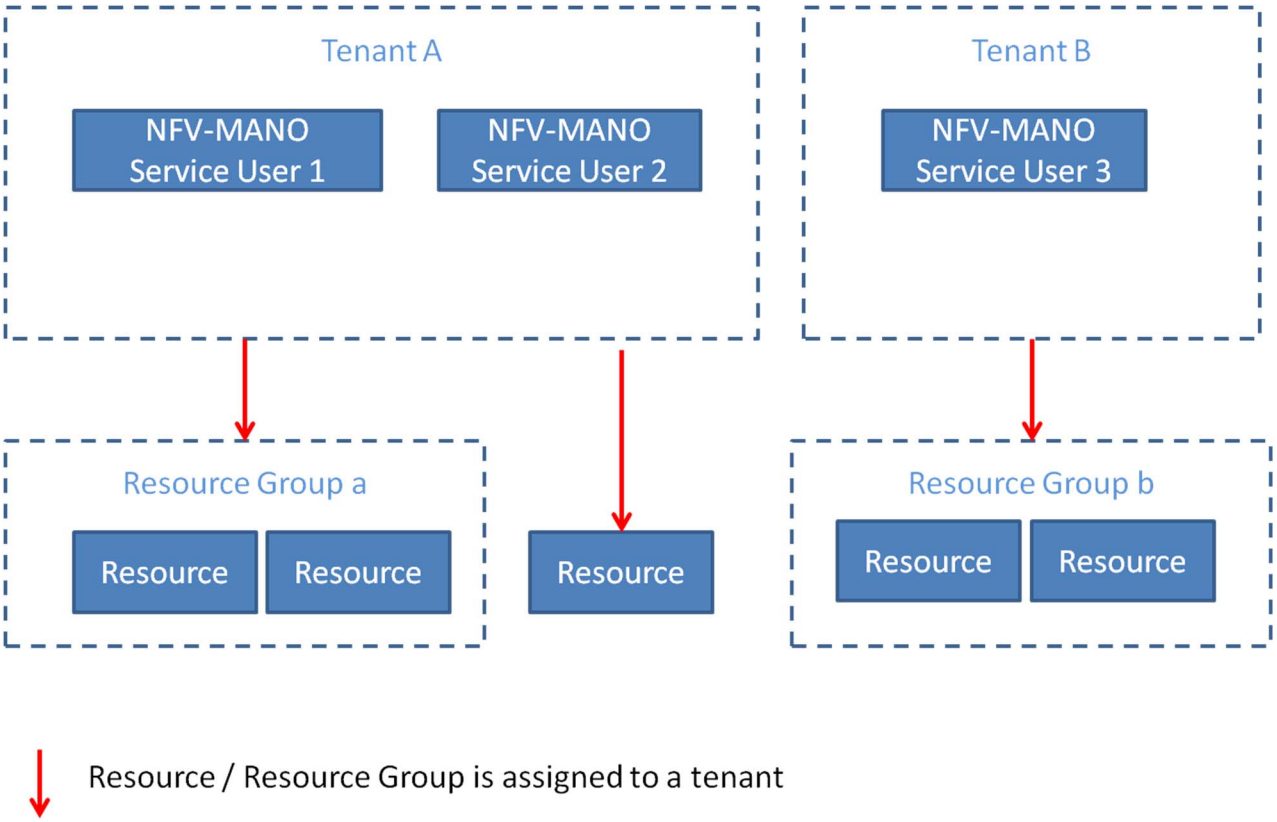

Figure 5.2-1: Entities relevant to multi-tenancy

Each FB may act as multiple tenants on the FBs from which it uses service or infrastructure resources. A service resource e.g. a VNF can be composed from multiple virtual resources from different tenants. Figure 5.2-2 shows an example how a VNFM may use tenants on the VIM.

EXAMPLE: The VNF (Resource Group a) is composed out of virtual resources from Resource Group c. The virtual resources in Resource Group c are assigned to Tenant C. Thus, the VNFM has to identify as Tenant C to modify the virtual resources for VNF (Resource Group a). The VNF (Resource Group b) uses virtual resources assigned to Tenant D (Resource Group d) and Tenant E. Therefore, the VNFM has to identify as Tenant D or Tenant E or both to modify the virtual resources for VNF (Resource Group b).

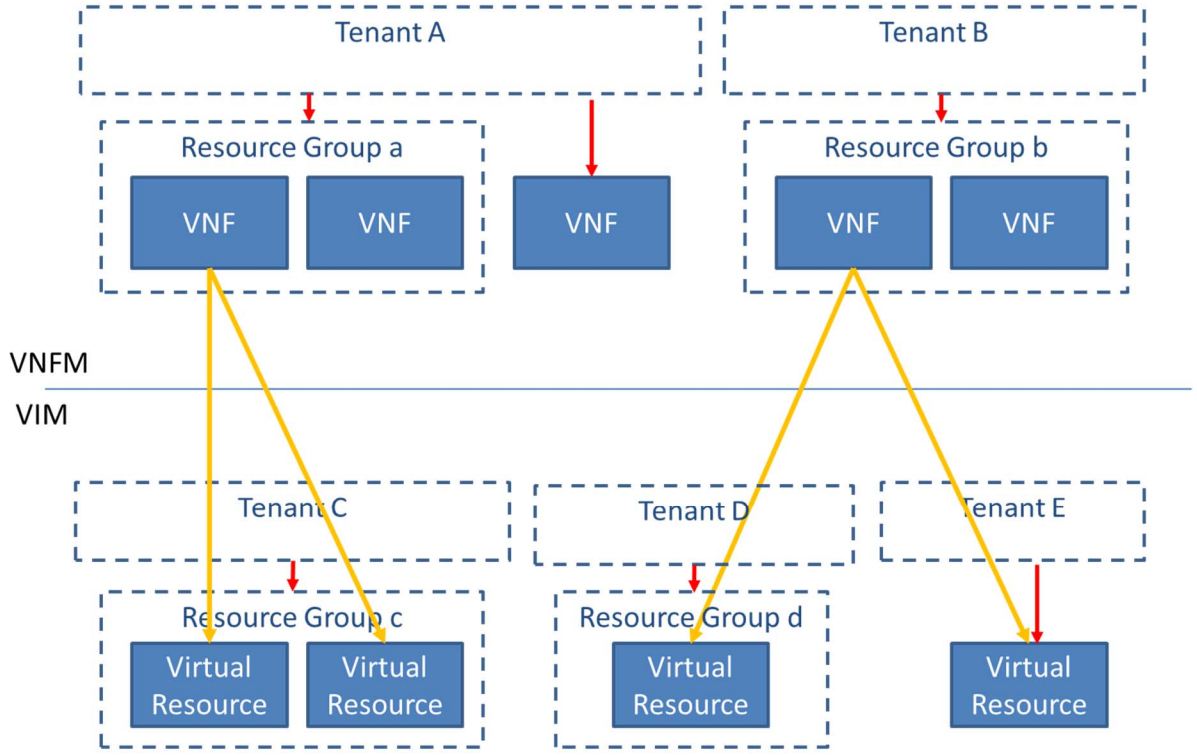

Figure 5.2-2: Example of how a VNFM may use tenants on a VIM

Since multi-tenancy exists for all kinds of service and infrastructure resources which can be used from an NFV-MANO service, tenants can be grouped based in the resources they use:

- A tenant to which virtual resources are assigned is referred to as an infrastructure tenant (Tenant C, D, E).
- A tenant to which VNFs are assigned is referred to as a VNF tenant (Tenant A, B).
- A tenant to which NSs are assigned is referred to as an NS tenant.

A resource group has different meaning for different resources which are being used:

- A resource group can be a "service resource group" containing VNFs, PNFs or NSs instances.
- A resource group can be an "infrastructure resource group" containing a set of virtual resources under the control of a VIM and belonging to a tenant.

The concepts of multi-tenancy and isolation between the tenants are important for support of network slices in NFV. The external systems managing network slices will act as NFV-MANO consumers. The resource groups can be assigned to single or multiple tenants, from the perspective of network slicing.

5.3 General requirements for the management of NFV-MANO functional entities

The NFV Architectural Framework shall support the management of NFV-MANO functional entities (i.e. NFVO, VNFM, VIM or the WIM when the WIM is integrated as part of the NFV-MANO framework). To fulfil this functionality, the NFV-MANO functional entity shall support and produce standard interfaces enabling consumers to perform the necessary management tasks such as configuration, performance and fault monitoring, retrieval of configuration and other information, state management and log management of a target NFV-MANO functional entity.

General requirements for the software modification of NFV-MANO functional entities are specified in clause 5.6.2.

5.4 General functional requirements for management of connectivity for Multi-Site services

In NFV-based network deployments, service providers deploy network services according to diverse business and operational requirements. In some cases, there will be network services deployed across multiple sites, wherein the endpoints and network functions will reside in two or more sites, which may be customer premises, N-PoPs or NFVI-PoPs. To fulfil these multi-site deployments, connectivity needs to be established among the service components, e.g. VNF, VNFC, PNF, possibly across Wide Area Networks (WAN) and/or access networks (collectively called WANs here), both legacy and SDN-enabled and their hybrid.

ETSI GR NFV-IFA 022 [i.11] introduces and analyses use cases related to multi-site connectivity. Clause 6.2 of ETSI GR NFV-IFA 022 [i.11] introduces the concept of the WIM that manages network resources across multiple NFVI-PoPs and, it is used to establish connectivity between different NFVI-PoPs, or between a PNF and an NFVI-PoP.

The NFV Architectural Framework shall support the management of connectivity across multiple sites to permit service providers to deploy and operate network services and VNFs on multiple sites.

The NFV Architectural Framework shall support the integration of WAN infrastructure management deployed as:

- Part of the NFV-MANO framework.
- External to the NFV-MANO framework (e.g. under control of other OSS/BSS systems).

Annex G illustrates and describes further these two integration variants.

In order to make network services deployable and operable across multiple sites, the NFV Information Model and descriptors shall contain the required information elements related to multi-site connectivity service.

5.5 General requirements to support network slicing

As described in ETSI GR NFV-EVE 012 [i.12], external systems managing network slices will use NFV-MANO and its capability to manage Network Services and their resources used for the network slices. Some principles of how this maps to specific requirements can be found in Annex D.

Network slice management functions will consume NS LCM when managing the constituents that are forming the network slices. NFV-MANO shall support priorities for network services to support network slicing based on operator policies and SLAs. It shall support the isolation of NSs assigned to different tenants. The NS instances supporting a network slice may span over multiple sites and multiple administrative domains.

NFV-MANO is not aware of the purpose for which the instantiation of an NS has been requested (i.e. the context of network slicing is invisible/transparent to NFV-MANO). The use of NS priority values (as introduced with network slicing in mind) allows NFV-MANO to resolve potential conflicts in LCM operations and resource allocations.

In case consumers expect conflicts be handled as "first come first served", the priority can be set to the same value.

The NFVO shall use the NS instance priority while resolving resource allocation conflicts during resource shortage situations in the following way:

- If multiple LCM operations are handled at the same time, resources shall be allocated in order of priority, starting with the highest priority NS instance. If necessary, operations on lower priority instances shall be pre-empted. See the use case in clause D.2.4.
- If a higher priority NS instance cannot be instantiated because of lack of resources and resources are allocated to lower priority NS instances, NFVO shall, while coordinating with the consumer and based on operator policies, terminate or scale in lower priority NS instance(s) to allow for the instantiation of the higher priority NS instance. See the use case in clause D.2.5. See note.
- If a higher priority NS instance cannot be scaled because of lack of resources and resources are allocated to lower priority NS instances, NFVO shall, while coordinating with the consumer and based on operator policies, terminate or scale in lower priority NS instance(s) to allow for the scaling of the higher priority NS instance. See note.

- If a higher priority NS instance cannot be healed because of lack of resources and the resources are allocated to lower priority NS instances, NFVO shall, while coordinating with the consumer and based on operator policies, terminate or scale in lower priority NS instance(s) to allow for the healing of the higher priority NS instance. See notes 1, 2 and 3.

- In case of a capacity shortage or performance limitation in NFV-MANO, NFVO shall use the NS instance priority to decide which LCM operations or healing will be executed first, be delayed or rejected.

NOTE 1: If not enough resources can be made available by terminating or scaling in lower priority NS instances, the instantiation/scaling/healing may fail for lack of resources.

NOTE 2: In case of NS instances with the same priority, NFVO cannot pre-empt, unless explicitly directed by the consumer.

NOTE 3: Annex D illustrates various cases of using the priority.

The NFVO shall notify consumers when resolving resource allocation conflicts using the NS instance priority during resource shortage situations.

The NFVO shall notify affected consumers e.g. after rejected LCM operations, when a resource or capacity shortage situation has ended and it can be expected that such LCM operations could now successfully executed if the consumer re-tries.

NFV-MANO shall support the isolation between network slices by isolating the infrastructure resources and/or isolating the service resources assigned to different tenants.

5.6 General requirements to support software modification

5.6.1 General requirements for software modification

An entire software modification task including preparation work (e.g. backup or snapshot) and closing actions (e.g. verify the normal service is resumed) shall be completed within a maximum maintenance period or, in case of a long-running maintenance, be completed as a set of separate phases, whereby each phase (including its preparation and closing work) can be completed within the specified maximum maintenance period.

NOTE 1: The functionality to support splitting the entire software modification into multiple phases could be realized by allowing to "pause" the software modification process at certain provider defined steps of the modification process.

NOTE 2: The provider of the software entity may specify an estimated minimum required time for the modification task which then can be used as lower bound for the maintenance period assuming a "default" performance expected for the software entity.

EXAMPLE: To complete the software modification of several entities 9 hours are estimated. However, due to operator settings, the maximum maintenance window is set to 6 hours. In order to support this use case, the software modification of the entities will be split into 2 phases of each maximal 6 hours that are executed in consecutive maintenance windows. Since, the provider of an entity can specify the minimum required time for the operation for the phase where the provider's entity is involved, e.g. 3 hours, i.e. the smallest granularity in time the software modification can be split into, it is used as the lower bound of the maintenance window.

Moreover, it shall be possible to monitor the progress of the software modification and it shall be possible to notify the progress (including start and end) of the software modification. In case of a failure at any stage of the entire software modification (i.e. preparation, actual software modification, and verification), the software modification shall be suspended, and it shall be possible to do a rollback of the software modification based on the operator's decision to roll back or to keep/re-try the software modification.

NOTE 3: Keeping or re-trying a software modification task will usually involve some manual steps/repair.

5.6.2 General requirements for the software modification of NFV-MANO functional entities

The management of NFV-MANO functional entities (i.e. NFVO, VNFM, or VIM) shall support the modification (i.e. upgrade/update) of their software. Thereby, it shall be ensured that any network service they are managing is kept running. In particular, the software modification shall not require the re-instantiation or the termination of any VNF or NS instance, it shall not impact the accessibility of the VNF and NS instances, and it shall not impact the virtualised resources used by the NS and VNF instances. Furthermore, it shall be ensured that after the software modification all NFV-MANO functional entities are connected amongst each other and towards external entities in the same way as prior to the software modification whenever the performed software modification relates to compatible changes such as bug fixing, backward compatible version changes to API, or support for new API versions in addition to old ones, etc.

During the software modification, it shall be possible to process an incoming NS LCM operation, VNF LCM operation, or virtualised resource management request. During the software modification, it shall also be possible to handle an ongoing NS LCM operation occurrence, VNF LCM operations occurrence or virtualised resource management. Depending on the type of the NS LCM, VNF LCM operation and/or virtualised resource management, the operator's configuration, and the capabilities of the software modification for the NFV-MANO functional block to be modified, an appropriate action can be taken. Possible actions include, e.g. to execute the NS and/or VNF LCM operation in parallel with the software modification of the NFV-MANO functional entities, delay the execution of the VNF LCM operation until the NFV-MANO software modification has completed, or notify the pending VNF LCM operation to consumer such that the consumer can trigger the VNF LCM operation again after the completion of the software modification.

NOTE: ETSI GS NFV-REL 006 [i.20] has specified requirements for the purpose of software modifications, such that NFV service availability and continuity is maintained. The present document version in Release 3 does not specify concrete functional requirements leading to potential interface requirements for handling the specific NFV-MANO software modification.

5.6.3 General requirements for changing the current VNF Package

Functional requirements for changing the current VNF Package, a.k.a VNF software modification, are specified in clause 7.2.5 of the present document.

5.6.4 General requirements for the software modification of NFVI components

5.6.4.1 Description

An initial set of requirements for the software modification of NFVI hardware and software components has been specified in ETSI GS NFV-REL 006 [i.20] with a focus on maintaining the VNF service and NS availability and continuity during the software modification. Software modifications at the NFVI will typically affect more than a single NFVI component and, thus, multiple virtualised resources of a VNF and NS may be impacted even simultaneously, which would result in service outage. To avoid such side effects a controlled deployment of software modifications across an entire resource pool is required which is also coordinated with the hosted VNFs. Therefore, a set of NFVI operation and maintenance constraints is specified in clause 5.6.4.2. The constraints can be provided in the VNFD or can be set at runtime. The constraints specify conditions for the NFVI operation and maintenance workflow from the VNF perspective that need to be fulfilled in order to enable the VNF to mitigate the disturbances caused by the NFVI operation and maintenance. For example, the impactNotificationLeadTime specifies the minimum "notice period" that a VNF requests to prepare for the upcoming impact.

The policy management interfaces are used to propagate any NFVI operation and maintenance constraints from the EM and/or VNFM to the VIM. Therefore, the constraints will be converted into NFVI operation and maintenance policies. Any policy conflicts detected by any of the NFV-MANO entities involved results in the rejection of the policy and shall be notified to the source(s) of the conflicting policies such that the conflict can be resolved.

The VIM shall notify any upcoming changes that will have an impact on the virtualised resources in order to allow the VNF to prepare for the impact, which can include updating some of the constraints, e.g. requesting an extension of the lead time due to an ongoing VNF LCM operation.

Annex H of the present document describes end-to-end examples of policy transfer and coordination procedures enabling the mitigation of NFVI operation and maintenance impacts.

5.6.4.2 NFVI operation and maintenance constraints

The following constraints shall be supported in NFV-MANO for the maintenance of the NFVI:

- impactNotificationLeadTime: Specifies the minimum notification lead time (relative time) requested for upcoming impact of the virtualised resource or their group (i.e. between the notification from the VIM and the action causing the impact).
- earliestTimeOfTermination: Specifies the earliest time (absolute time) when the virtualised resource instance(s) can be impacted or terminated. See note 3.
- supportedMigrationType: Specifies the allowed migration types (and their priorities) for virtual compute and virtual storage resources. Possible values:
  - NO_MIGRATION;
  - OFFLINE_MIGRATION;
  - LIVE_MIGRATION.

  See note 1.
- maxUndetectableInterruptionTime: Specifies the maximum interruption time that can be tolerated by the VNF. See note 1.
- minRecoveryTimeBetweenImpacts: Specifies the minimum time duration between consecutive impacts on members of a virtualised resource group to allow the consumer to recover from an impact on the virtualised resource instance(s) before the next impact on members of the group. For example, this allows a VNF to restore its redundancy after an impact. See note 2.
- impactSize: Specifies the number of members of a virtualised resource group that can be impacted at the same time. It shall be possible to provide different values depending on the size of the group. See note 2. The impactSize can be expressed either as:
  - maxNumberOfImpactedInstances: Specifies the maximum number of members of the group that can be impacted simultaneously without losing functionality; or
  - minNumberOfPreservedInstances: Specifies the minimum number of members of the group which need to be preserved simultaneously within a virtualised resource group.

NOTE 1: Live migration can be constrained by the maxUndetectableInterruptionTime. When the maximum undetectable interruption time is specified, it constrains the applicability of live migration.

NOTE 2: Impacts to instances of the group happening within the minimum recovery time are considered simultaneous impacts.

NOTE 3: The VIM notifies the time of the upcoming impact (e.g. based on the configuration or the constraint of impactNotificationLeadTime), which is provided to the EM. In response, the EM can provide via the VNFM the earliestTimeOfTermination constraint to the VIM. The earliestTimeOfTermination delays the start time of the upcoming impact. It is assumed that the time is synchronized between all involved entities.

In addition to the above list, the following constraint(s) should be supported in NFV-MANO for the maintenance of the NFVI:

- isImpactMitigationRequested: Specifies whether virtualised resources compensating for an upcoming impact are requested to be provided for the VNF in the notification of the upcoming impact.

5.7 General requirements to support service availability level

Service Availability Level (SAL) is information used to assist in the selection of appropriate virtualised resources to be allocated to or reserved for constituents of a Network Service (NS) to meet the availability expectation of the service provider towards the NS the constituents belong to. The use of SAL is optional for service providers.

SAL information may be either assigned to the NS or to the constituents of type "VNF" and "VL" of this NS, but not both. When SAL information is assigned to an NS, this information does not apply to any constituents of type "nested NS". The NFVO uses this information to grant the allocation or reservation of virtualised resources to the NS constituents, which eventually supports the NS with the required level of availability. The details of the method of selection of virtualised resources considering SAL information are out of scope of the present document version.

The NFVO shall support the capability to consider the provided SAL information at the allocation or reservation of virtualised resources, under all circumstance including situations of resource shortage.

6 Functional requirements for NFVO

6.1 Functional requirements for virtualised resource management

6.1.1 Functional requirements for general virtualised resource management

Table 6.1.1-1: Functional requirements for general virtualised resource management

Numbering	Functional requirements description
Nfvo.Gvrm.001	The NFVO shall support orchestration of actions related to virtualised resources managed by one or more VIMs.
Nfvo.Gvrm.002	The NFVO shall support the capability to mitigate conflicts in resource allocation in case of conflicting resource requests.
Nfvo.Gvrm.003	The NFVO shall support the capability to provide deployment-specific configuration information for virtualised resources related to NS.
Nfvo.Gvrm.004	The NFVO shall support the capability to consider priority information in actions related to virtualised resources.
Nfvo.Gvrm.005	The NFVO shall support the capability to consider priority information while mitigating conflicts in resource allocation.
Nfvo.Gvrm.006	The NFVO should support the capability to consider priority information while providing deployment-specific configurations information for virtualised resources related to NS.
Nfvo.Gvrm.007	The NFVO shall support the capability to consider the service availability level in the orchestration of actions related to virtualised resources managed by one or more VIMs.
Nfvo.Gvrm.008	The NFVO shall support the capability to consider the service availability level for mitigating conflicts in resource allocation in case of conflicting resource requests (see note).
NOTE: The service availability level is used for resolving conflicts when no priority or equal priority is assigned to the deployment flavour of NS instances with conflicting resource requests.	

6.1.2 Functional requirements for VNF-related resource management in indirect mode

Table 6.1.2-1: Functional requirements for VNF-related resource management in indirect mode

Numbering	Functional requirements description
Nfvo.VnfRmpbNfvo.001	When VNF-related Resource Management in indirect mode is applicable, the NFVO shall support the capability to request to the VIM the management of virtualised resources needed for VNFs instantiation, scaling and termination (see notes 1 and 4).
Nfvo.VnfRmpbNfvo.002	When VNF-related Resource Management in indirect mode is applicable, the NFVO shall support the capability to invoke resource management operations toward the VIM as requested by the VNFM.
Nfvo.VnfRmpbNfvo.003	When VNF-related Resource Management in indirect mode is applicable, the NFVO shall support the capability to receive notifications regarding the resources being allocated to or released from specific VNF instances, as well as regarding events and relevant fault reports related to those resources (see notes 1 and 3).
Nfvo.VnfRmpbNfvo.004	When VNF-related Resource Management in indirect mode is applicable, the NFVO shall support the capability to request allocation and update of resources in the different resource commitment models (see note 2).
Nfvo.VnfRmpbNfvo.005	When VNF-related Resource Management in indirect mode is applicable, the NFVO shall support the capability to request to the VIM affinity and anti-affinity policies for the VNF's virtualised resources (see note 1).
Nfvo.VnfRmpbNfvo.006	When VNF-related Resource Management in indirect mode is applicable, the NFVO shall support the capability of providing the VIM with constraints and policies applicable to specific virtualised resources and groups of virtualised resources related to specific VNF instances (see note 5).
Nfvo.VnfRmpbNfvo.007	When VNF-related Resource Management in indirect mode is applicable, the NFVO shall support the capability to update constraints and policies applicable to the virtualised resources and groups of virtualised resources related to specific VNF instances (see notes 5 and 6).
Nfvo.VnfRmpbNfvo.008	When VNF-related Resource Management in indirect mode is applicable, the NFVO shall support the capability to receive from VNFM the constraints and policies applicable to virtualised resources allocated to a specific VNF instance (see note 5).
NOTE 1: Virtual resources managed for the LCM of VNFs include compute and storage resources needed for VNF components as well as networking resources needed to ensure intra-VNF connectivity. NOTE 2: Resource commitment models are: reservation model, quota model and on-demand. NOTE 3: Events include NFVI outage, NFVI software modification and performance related events. NOTE 4: The management of virtualised resources includes allocation, update, scaling, termination, etc. of virtualised resources. NOTE 5: Constraints and policies related to virtualised resource(s) that can be impacted by NFVI maintenance activities or other operations. NOTE 6: Updates shall be done in accordance with the constraints provided in the VNFD, if any.	

6.1.3 Functional requirements for VNF-related resource management in direct mode

Table 6.1.3-1: Functional requirements for VNF-related resource management in direct mode

Numbering	Functional requirements description
Nfvo.VnfRmpbVnfm.001	When VNF-related Resource Management in direct mode is applicable, the NFVO shall support the capability to provide appropriate information about VIM to enable the VNFM to access the VIM.

6.1.4 Functional requirements for NS-related resource management performed by the NFVO

Table 6.1.4-1: Functional requirements for NS-related resource management performed by NFVO

Numbering	Functional requirements description
Nfvo.NsRmpbNfvo.001	The NFVO shall support the capability to issue requests to the VIM in order to allocate resources needed for the connectivity of NSs, identify current resource allocations associated with a particular NS instance, update current resources allocated to the NS instance or release resources that had been allocated to an NS instance (see note 1).
Nfvo.NsRmpbNfvo.002	The NFVO shall support the capability to query to the VIM about the resources that are allocated for the connectivity of the VNF Forwarding Graphs (VNFFGs) of specific NS instances.
Nfvo.NsRmpbNfvo.003	The NFVO shall support the capability to receive notifications of the resources that are allocated to or released from specific NS instances as well as events and relevant fault reports related to those resources (see notes 1 and 2).
Nfvo.NsRmpbNfvo.004	The NFVO shall support the capability to consider the priority information when dealing with the resources.
Nfvo.NsRmpbNfvo.005	As part of a feasibility check, the NFVO shall support the capability to reserve the resources needed for an NS LCM operation.
Nfvo.NsRmpbNfvo.006	The NFVO shall support the capability to release resources, that had previously been reserved as part of a feasibility check of an NS LCM operation.
Nfvo.NsRmpbNfvo.007	The NFVO shall support the capability to consider the affinity/anti-affinity rules when dealing with the resources needed for the connectivity of NSs (see note 3).
Nfvo.NsRmpbNfvo.008	The NFVO shall support the capability to determine the required virtualized network resources to interconnect the constituents (e.g. VNFs) in a NS instance to meet the requirements for the NS Virtual links based on the information provided in the NSD.
NOTE 1: Resources needed for the connectivity of NSs include networks, subnets, ports, routing resources, addresses, links and forwarding rules, and are used for the purpose of ensuring inter-VNF connectivity. NOTE 2: Events include NFVI outage and performance related events. NOTE 3: An example of an affinity/anti-affinity rule applicable to the connectivity of NS constituents to an NS VL can be whether to share or not share the same layer 2 NFVI resources.	

6.1.5 Functional requirements for resource reservation management

Table 6.1.5-1: Functional requirements for resource reservation management

Numbering	Functional requirements description
Nfvo.Rrm.001	The NFVO shall support the capability to request creation, query, update and termination of virtualised resource reservation to corresponding VIM(s) as part of NS LCM, VNF LCM, and VNF lifecycle granting procedures, and during configuration/reconfiguration of resources in the NFVI Point of Presence(s) (NFVI-PoPs).
Nfvo.Rrm.002	The NFVO shall support the capability to consider affinity/anti-affinity rules for resource reservation management.
Nfvo.Rrm.003	The NFVO shall support the capability to receive change notification regarding to virtualised resource reservation.
Nfvo.Rrm.004	When a resource reservation model is used, the NFVO shall support the capability to provide to VNFM resource reservation identification information.
Nfvo.Rrm.005	The NFVO shall support the capability to consider NS instance priorities for virtualised resource reservation.
Nfvo.Rrm.006	The NFVO shall support the capability to use resources reservation management requests to perform the feasibility check and/or the resource reservation for an NS LCM operation.

6.1.6 Functional requirements for virtualised resource and NFVI capacity management

Table 6.1.6-1: Functional requirements for virtualised resource capacity management

Numbering	Functional requirements description
Nfvo.Vrcm.001	The NFVO shall support the capability to maintain information regarding the virtualised resources capacity and its usage at different granularities, including usage per VNFM or per NS (see note 1).
Nfvo.Vrcm.002	The NFVO shall support the capability to query information about resource zones managed by the VIM and about NFVI-PoP(s) administered by the VIM.
Nfvo.Vrcm.003	The NFVO shall support the capability to maintain information regarding the resource zones available on the connected VIMs.
Nfvo.Vrcm.004	The NFVO shall support the capability to retrieve information regarding the virtualised resources capacity and its usage at different granularities and levels, including (not limited to) total per NFVI-PoP and per resource zone (see note 2).
Nfvo.Vrcm.005	The NFVO shall support the capability to synchronize periodically and automatically, or on demand, the virtualised resource capacity information maintained in the NFVO with the information managed by the VIM(s).
Nfvo.Vrcm.006	The NFVO shall support the capability to configure thresholds for setting virtualised resource capacity shortage alarms at different granularities and levels, including (not limited to) per NFVI-PoP and per resource zone.
Nfvo.Vrcm.007	The NFVO shall support the capability to notify about virtualised resource capacity shortage.
Nfvo.Vrcm.008	The NFVO shall support the capability to receive the notification from VIM related to the changes to NFVI capacity information.
NOTE 1: This information can be maintained for multiple uses, including statistics, analytics, granting VNF requests, management of NS, determining placement for VNFs on certain NFVI-PoPs and resource zones, for general network planning, etc. Refer to Annex B for further information. NOTE 2: The capacity information can include information related to available, allocated, reserved and total virtualised resource capacity.	

Table 6.1.6-2: Functional requirements for NFVI capacity management

Numbering	Functional requirements description
Nfvo.Ncm.001	The NFVO shall support the capability to maintain information regarding the NFVI capacity (including compute hosts) and its usage at different granularities, including usage per VNFM, per NS, per NFVI-PoP, or for the whole NFVI (see note 1).
Nfvo.Ncm.002	The NFVO shall support the capability to retrieve information regarding the NFVI capacity (including compute hosts) and its usage at different granularities and levels, including (not limited to) total per NFVI-PoP and per resource zone (see note 2).
Nfvo.Ncm.003	The NFVO shall support the capability to synchronize periodically and automatically, or on demand, the NFVI capacity information maintained in the NFVO with the information managed by the VIM(s).
Nfvo.Ncm.004	The NFVO shall support the capability to configure thresholds for setting NFVI capacity shortage alarms at different granularities and levels, including (not limited to) per NFVI-PoP and per resource zone.
Nfvo.Ncm.005	The NFVO shall support the capability to notify about NFVI resource capacity shortage.
NOTE 1: This information can be maintained for multiple uses, including statistics, analytics, granting VNF requests, management of NS, determining placement for VNFs on certain NFVI-PoPs and resource zones, for general network planning, etc. Refer to Annex B for further information. NOTE 2: The capacity information includes information related to available, allocated, reserved and total NFVI capacity.	

6.1.7 Functional requirements for virtualised resource performance management

Table 6.1.7-1: Functional requirements for virtualised resource performance management

Numbering	Functional requirements description
Nfvo.Vrpm.001	The NFVO shall support the capability to invoke the virtualised resource performance management operations on the virtualised resources for the NS(s) it manages (see notes 1 and 2).
Nfvo.Vrpm.002	The NFVO shall support the capability to receive performance information related to virtualised resources for the NS(s) it manages (see note 2).
Nfvo.Vrpm.003	The NFVO shall support the capability to map to the NS(s) the received performance information related to virtualised resources (see note 2).
NOTE 1: The virtualised resource performance management can include: setting threshold conditions on the performance information collected by the VIM for specific virtualised resource(s), creating Performance Management (PM) jobs by specifying different limitations and conditions for collecting and reporting of performance information from specified virtualised resource(s), etc. NOTE 2: The performance management operations mentioned in this requirement apply to performance measurements defined for measured object types applicable to the Or-Vi reference point.	

6.1.8 Functional requirements for virtualised resource fault management

Table 6.1.8-1: Functional requirements for virtualised resource fault management

Numbering	Functional requirements description
Nfvo.Vrfm.001	The NFVO shall support the capability to collect fault information related to the virtualised resources allocated to NS(s) that it manages.
Nfvo.Vrfm.002	The NFVO shall support the capability to correlate the virtualised network resource fault information with the impacted NS(s) that it is managing.
Nfvo.Vrfm.003	The NFVO shall support the capability to request corrective operations on virtualised network resources to VIM in order to perform NS healing (see note).
NOTE: The virtualised network resources refer to the virtualised resources supporting the connectivity of the NS instance(s).	

6.1.9 Functional requirements for virtualised resource information management

Table 6.1.9-1: Functional requirements for virtualised resource information management

Numbering	Functional requirements description
Nfvo.Vrim.001	The NFVO shall support collection of information on virtualised resource that can be consumed in a VIM or across multiple VIMs.
Nfvo.Vrim.002	The NFVO shall support the capability to forward the information about resource shortage to the Operation Support System (OSS) as soon as it becomes available in the NFVO.
Nfvo.Vrim.003	The NFVO shall support the capability to receive the notifications regarding the changes of the information on consumable virtualised resources that can be provided by the VIM(s).

6.1.10 Functional requirements for Network Forwarding Path (NFP) management

Table 6.1.10-1: Functional requirements for NFP management

Numbering	Functional requirements description
Nfvo.Nfpm.001	The NFVO shall support the capability of requesting management of NFPs.
Nfvo.Nfpm.002	The NFVO should support the capability to provide or update the classification and selection rules applied to a specific NFP instance (see note 1).
Nfvo.Nfpm.003	The NFVO shall support the capability to receive the classification and selection rules applied to NFP(s) from an authorized entity (see note 2).
NOTE 1: The classification and selection rules applied to NFPs can be rules to classify and select NFPs. An NFP is allocated as the default path for specific types of traffic or packets. The rules are provided to VIM by NFVO, and VIM configures those rules in the Network Controllers to enable the Network Controllers to configure corresponding forwarding tables in NFVI network resources. NOTE 2: The classification and selection rules applied to NFPs are optionally included in the NS Descriptor (NSD). In cases when they are not included, they can be provided to NFVO later to be assigned to an existing NFP. The authorized entity sending NFP rule to NFVO may include OSS/Business Support System (BSS).	

6.1.11 Functional requirements for quota management

Table 6.1.11-1: Functional requirements for quota management

Numbering	Functional requirements description
Nfvo.Qm.001	The NFVO shall support the capability to request the VIM to create the quota for a consumer of the virtualised resources.
Nfvo.Qm.002	The NFVO shall support the capability to request the VIM to change the quota for a consumer of the virtualised resources.
Nfvo.Qm.003	The NFVO shall support the capability to request the VIM to delete the quota for a consumer of the virtualised resources.
Nfvo.Qm.004	The NFVO shall support the capability to query to the VIM the information of the quota for a consumer of the virtualised resources.
Nfvo.Qm.005	The NFVO shall support the capability to receive change notification regarding virtualised resource quota.
Nfvo.Qm.006	The NFVO may support the capability to provide to the VNFM the information on available quota(s) applicable to this VNFM (see notes 1 and 2).
NOTE 1: The information on available quota(s) allows the VNFM to interact with the VIM to receive information regarding the quota(s) applied to the VNFM or the VNF(s) which the VNFM manages, when VNF-related Resource Management in direct mode is applicable. NOTE 2: The information on available quota(s) allows the VNFM to interact with the NFVO to receive information regarding the quota(s) applied to the VNFM or the VNF(s) which the VNFM manages, when VNF-related Resource Management in indirect mode is applicable.	

6.1.12 Functional requirements related to permitted allowance management

Table 6.1.12-1: Functional requirements related to permitted allowance management

Numbering	Functional requirements description
Nfvo.Pam.001	When an allowance model is used, it shall be possible for the NFVO to maintain and enforce permitted allowance at various granularity levels (VNFM, VNF, NS, etc.).
Nfvo.Pam.002	A permitted allowance shall be expressed either as a defined amount of resources or as percentages of the total available resources per type of resources.
Nfvo.Pam.003	When an allowance model is used, the NFVO shall support the capability to reject any granting requests from VNFM that would cause the corresponding permitted allowance to be exceeded (see note).
Nfvo.Pam.004	When an allowance model is used, the NFVO shall support the capability to manage the overall consumption of resources across all permitted allowances.
Nfvo.Pam.005	When an allowance model is used, the NFVO shall support the capability to provide notification when the permitted allowance reaches its limit.
Nfvo.Pam.006	When an allowance model is used, the NFVO shall support the capability to process a request for permitted allowance extension or permitted allowance reduction.
Nfvo.Pam.007	When an allowance model is used, the NFVO shall support the capability to arbitrate conflict in permitted allowance consumption (see example 1).
Nfvo.Pam.008	When an allowance model is used, the NFVO shall support the capability to consider NS priority information in conflict arbitration in permitted allowance consumption (see example 2).
NOTE: NFVO might decide, based on policy, to extend a given allowance reaching its limit.	
EXAMPLE 1: An example of conflict can be in case when multiple concurrent resource allocations can be foreseen to exceed the allowance. EXAMPLE 2: An example of conflict can be multiple concurrent resource allocation requests related to different NS instances, together exceeding the allowance. The priorities of each NS instance involved in the conflict may be used for the conflict arbitration.	

6.2 Functional requirements for VNF lifecycle management

6.2.1 Functional requirements for VNF lifecycle management

Table 6.2.1-1: Functional requirements for VNF lifecycle management

Numbering	Functional requirements description
Nfvo.VnfLcm.001	The NFVO shall support the capability to process notifications about VNF lifecycle change.
Nfvo.VnfLcm.002	The NFVO shall support the capability of granting of the LCM requests.
Nfvo.VnfLcm.003	The NFVO shall support the capability to validate the lifecycle operation requests submitted to it, using information specified in the VNF Package.
Nfvo.VnfLcm.004	The NFVO shall support the capability to request changing the state of a VNF instance (see note 1).
Nfvo.VnfLcm.005	When NFVO is the consumer of the VNF LCM operation, the NFVO shall support the capability to query the status of the ongoing LCM operation.
Nfvo.VnfLcm.006	The NFVO shall support the capability to query information about a VNF instance.
Nfvo.VnfLcm.007	The NFVO shall support the capability to request the creation and deletion of the identifier of a VNF instance.
Nfvo.VnfLcm.008	The NFVO shall support the capability to request VNFM to conduct error handling operation(s) after the VNF life cycle operation occurrence fails (see notes 2 and 3).
Nfvo.VnfLcm.009	The NFVO shall support the capability to consider NS instance priorities while granting of the VNF LCM requests.
Nfvo.VnfLcm.010	The NFVO shall support the capability to check the availability of the requested amount of OS container extended resources during granting when the VNF is realised by a set of OS containers. See note 4.
NOTE 1: Change state refers to start and stop a VNF instance/VNF Component (VNFC) instances(s). These operations are complementary to instantiate/create a VNF or terminate a VNF. NOTE 2: It is up to the protocol design stage to design the detail error handling operation(s). NOTE 3: It depends on the VNF capabilities and is declared in the VNFD whether and how the operation(s) are supported by a particular VNF. NOTE 4: The actual type of the OS container extended resources may be transparent for the NFVO.	

6.2.2 Functional requirements for VNF instantiation

Table 6.2.2-1: Functional requirements for VNF instantiation

Numbering	Functional requirements description
Nfvo.VnfI.001	The NFVO shall support the capability to request the instantiation of a VNF instance.
Nfvo.VnfI.002	The NFVO shall support the capability to send to the VNFM, as part of the VNF instantiation request, input parameters specific for the VNF instance being instantiated.

6.2.3 Functional requirements for VNF scaling

NOTE: The LCM operations that expand or contract a VNF instance include scale in, scale out, scale up, scale down. Not all VNFs support all these operations, which implies that the set of operations that a VNFM will be able to perform on a VNF instance will depend on the VNF capabilities.

Table 6.2.3-1: Functional requirements for VNF scaling

Numbering	Functional requirements description
Nfvo.VnfS.001	The NFVO shall support the capability to request expanding the capacity of a VNF instance (see note 1).
Nfvo.VnfS.002	The NFVO shall support the capability to request contracting the capacity of a VNF instance (see note 2).
NOTE 1: Expansion can either be performed by scaling out or scaling up. NOTE 2: Contraction can either be performed by scaling in or scaling down.	

6.2.4 Functional requirements for VNF termination

Table 6.2.4-1: Functional requirements for VNF termination

Numbering	Functional requirements description
Nfvo.VnfT.001	The NFVO shall support the capability to request the termination of a VNF instance.
Nfvo.VnfT.002	The NFVO shall support the capability to check the dependencies between VNF instances before granting the termination of a particular VNF instance.

6.2.5 Functional requirements for VNF/VNFC Snapshots

Table 6.2.5-1: Functional requirements for VNF/VNFC Snapshots

Numbering	Functional requirements description
Nfvo.VnfSnap.001	The NFVO shall support the capability of granting VNF/VNFC Snapshot operation requests according to operator policies (see note).
NOTE: VNF/VNFC Snapshot operations include VNF/VNFC Snapshot creation and reversion.	

6.2.6 Functional requirements for changing the current VNF Package

Table 6.2.6-1: Functional requirements for changing the current VNF Package

Numbering	Functional requirements description
Nfvo.VnfSwm.001	The NFVO shall have the capability to support changing the current VNF Package.
NOTE: The capability includes updates and upgrades of the software of VNFs.	

6.2.7 Functional requirements for change of the external VNF connectivity

Table 6.2.7-1: Functional requirements for change of the external VNF connectivity

Numbering	Functional requirements description
Nfvo.VnfCC.001	The NFVO shall support the capability to request the change of external connectivity of a VNF instance. See note.
NOTE: Changing the external connectivity may imply connecting the VNF to an additional network or network segment to disconnect the VNF from a network or network segment to which it was previously connected.	

6.3 Functional requirements for NS lifecycle management

6.3.1 Functional requirements for NS lifecycle management

Table 6.3.1-1: Functional requirements for NS lifecycle management

Numbering	Functional requirements description
Nfvo.NsLcm.001	The NFVO shall ensure the integrity of data related to the NS instances (e.g. descriptors, software images, records, etc.) against loss and corruption from hardware/software failures and against tampering with such data by unauthorized parties.
Nfvo.NsLcm.002	The NFVO shall support the capability to use the deployment information from the NSD for the NS LCM.
Nfvo.NsLcm.003	The NFVO shall support the capability to notify about the following events related to NS lifecycle changes: • The start of the lifecycle procedure. • The result of a feasibility check related to an NS LCM operation. • The end and the result of the lifecycle procedure.
Nfvo.NsLcm.004	The NFVO shall support the capability to execute scheduled NS lifecycle operations.
Nfvo.NsLcm.005	The NFVO shall support the capability to manage the connectivity between the VNFs, nested NS(s) and PNF(s) that are part of the NS.
Nfvo.NsLcm.006	The NFVO shall support the capability to provide the status of an NS LCM operation in response to a request.
Nfvo.NsLcm.007	The NFVO shall support the capability to consider priority information while executing scheduled NS lifecycle operations.
Nfvo.NsLcm.008	The NFVO shall support the capability to use descriptor identifiers of NSD constituents provided at the interface with precedence over those descriptor identifiers included in the NSD when executing NS LCM operations that create instances of those constituents or add them to the NS instance (see note).
NOTE: The purpose of this requirement is to avoid a change in the NSD if a different descriptor version of one or some of the NSD constituents (VNFDs, nested NSDs or PNFDs) is used for an NS instance.	

6.3.2 Functional requirements for NS instantiation

Table 6.3.2-1: Functional requirements for NS instantiation

Numbering	Functional requirements description
Nfvo.NsI.001	The NFVO shall support the capability to manage the instantiation of an NS instance (see notes 3 and 4).
Nfvo.NsI.002	The NFVO shall support the capability to invoke the instantiation of the constituent VNFs for an NS (see note 5).
Nfvo.NsI.003	The NFVO shall support the capability to invoke the creation of the constituent VLs for an NS.
Nfvo.NsI.004	The NFVO shall support the capability to create VNFFG(s) for an NS (see note 1).
Nfvo.NsI.005	The NFVO shall support the capability to instantiate an NS which includes existing VNF instances (see note 2).
Nfvo.NsI.006	The NFVO shall support the capability to perform a feasibility check of an NS instantiation operation in terms of availability of the resources.
Nfvo.NsI.007	The NFVO shall support the capability to reserve resources needed for an NS instantiation operation during the feasibility check for that NS instantiation operation.
Nfvo.Nsl.008	The NFVO shall support the capability to instantiate a composite NS which includes existing NS instances as nested NSs (see note 6).
NOTE 1: The VNFFG(s) of an NS can include PNF(s). NOTE 2: The VNF descriptors (VNFDs) of the existing VNF instances shall be referenced from the NSD of the NS instance being instantiated or from new values provided at the interface when the NS instantiation is invoked. The existing VNF instances may need to be modified as part of NS instantiation. NOTE 3: The NFVO shall use the nested NSDs referenced from the composite NSD or from new values provided at the interface when the composite NS instantiation is invoked to instantiate the constituent nested NSs. See also Nfvo.NsLcm.008. NOTE 4: The NFVO shall use the PNFDs referenced from the NSD or from new values provided at the interface when the NS instantiation is invoked to add the PNFs to the NS. See also Nfvo.NsLcm.008. NOTE 5: The NFVO shall use the VNFDs referenced from the NSD or from new values provided at the interface when the NS instantiation is invoked to request the instantiation of the constituent VNFs. See also Nfvo.NsLcm.008. NOTE 6: The NS descriptors (NSDs) of the existing NS instances shall be referenced from the NSD of the composite NS instance being instantiated or from new values provided at the interface when the NS instantiation is invoked. See also Nfvo.NsLcm.008.	

6.3.3 Functional requirements for NS scaling

Table 6.3.3-1: Functional requirements for NS scaling

Numbering	Functional requirements description
Nfvo.NsS.001	The NFVO shall support the capability to manage the expansion of an NS instance (see note 1).
Nfvo.NsS.002	The NFVO shall support the capability to manage the contraction of an NS instance (see note 2).
Nfvo.NsS.003	The NFVO shall support the capability to request to scale a VNF instance as part of the expansion/contraction of an NS instance.
Nfvo.NsS.004	The NFVO shall support the capability to evaluate the impact on NS instance(s) it manages when scaling needs to be performed on a component instance (i.e. a VNF or nested NS) shared or not.
Nfvo.NsS.005	The NFVO shall support the capability to consider NS instance priorities while evaluating NS expansion.
Nfvo.NsS.006	The NFVO may support the capability to consider NS instance priorities while evaluating NS contraction.
NOTE 1: Expansion can either be performed by increasing the number of the existing VNF instance(s) or expansion of the existing VNF instance(s). NOTE 2: Contraction can either be performed by decreasing the number of the existing VNF instance(s) or contraction of the existing VNF instance(s).	

6.3.4 Functional requirements for NS updating

Table 6.3.4-1: Functional requirements for NS updating

Numbering	Functional requirements description
Nfvo.NsU.001	The NFVO shall support the capability to manage the update of an NS instance.
Nfvo.NsU.002	The NFVO shall support the capability to add new VNF(s)/VL(s)/VNFFG(s)/PNF(s)/Nested NS(s)/Service Access Point(s) (SAPs) to an existing NS in order to perform NS update (see notes 3 and 4).
Nfvo.NsU.003	The NFVO shall support the capability to remove the VNF(s)/VL(s)/VNFFG(s)/PNF(s)/Nested NS(s)/SAP(s) from an existing NS in order to perform NS update.
Nfvo.NsU.004	The NFVO shall support the capability to update the existing VNF(s)/VL(s)/VNFFG(s) involved in an existing NS (see note 1).
Nfvo.NsU.005	The NFVO shall support the capability to add existing VNF instance(s) to an existing NS (see note 2).
Nfvo.NsU.006	The NFVO shall support the capability to perform a feasibility check of an update NS operation in terms of availability of the resources.
Nfvo.NsU.007	The NFVO shall support the capability to reserve resources needed for the update NS operation during the feasibility check for that update NS operation.
Nfvo.NsU.008	The NFVO shall support the capability to add existing NS instance(s) as nested NS(s) to an existing NS (see note 5).
NOTE 1: The operation of updating the existing VNF(s) involved in an existing NS is embedded in the fine grained NS LCM operation, and can include: changing the Deployment Flavour (DF) of VNF instances, changing the operational state of a VNF instance, modifying VNF information data, modifying VNF configuration data.	
NOTE 2: The VNFDs of the existing VNF instances shall be referenced from the NSD of the NS instance being updated or from new values provided at the interface when the NS updating is invoked. The existing VNF instance(s) may need to be modified as part of NS update. See also Nfvo.NsLcm.008.	
NOTE 3: The NFVO shall use the VNFDs referenced from the NSD or from new values provided at the interface when the NS update is invoked to request the instantiation of new constituent VNFs. See also Nfvo.NsLcm.008.	
NOTE 4: The NFVO shall use the PNFDs referenced from the NSD or from new values provided at the interface when the NS update is invoked to add PNFs to the NS instance. See also Nfvo.NsLcm.008.	
NOTE 5: The NSDs of the existing NS instances shall be referenced from the NSD of the composite NS instance being updated or from new values provided at the interface when the NS updating is invoked. See also Nfvo.NsLcm.008.	

6.3.5 Functional requirements for NS termination

Table 6.3.5-1: Functional requirements for NS termination

Numbering	Functional requirements description
Nfvo.NsT.001	The NFVO shall support the capability to terminate an NS instance.
Nfvo.NsT.002	The NFVO shall support the capability to request the termination of VNF instance(s) in order to perform NS termination.
Nfvo.NsT.003	The NFVO shall support the capability to retain a VNF instance currently used by another NS instance (i.e. other than the NS being terminated) when performing NS termination.
Nfvo.NsT.004	The NFVO shall support the capability to return information about retained VNF instance(s) used by another NS instance (i.e. other than the NS being terminated) when performing NS termination.

6.4 Functional requirements for VNF configuration management

Configuration parameters referred in this clause include those set at initial configuration and any other configurable parameter declared in the VNFD.

Table 6.4-1: Functional requirements for VNF configuration management

Numbering	Functional requirements description
Nfvo.VnfCm.001	The NFVO shall support the capability to invoke a request to set initial configuration parameters for a VNF instance.
Nfvo.VnfCm.002	The NFVO shall support the capability to invoke a request to update configuration parameters for a VNF instance.

6.5 Functional requirements for VNF information management

6.5.1 Functional requirements for VNF Package management

Table 6.5.1-1: Functional requirements for VNF Package management

Numbering	Functional requirements description
Nfvo.VnfPkgm.001	The NFVO shall support the capability of management of VNF Packages (see note 1).
Nfvo.VnfPkgm.002	The NFVO shall support the capability to verify the integrity and authenticity of the VNF Package.
Nfvo.VnfPkgm.003	The NFVO shall support the capability to verify that all mandatory information in the VNF Package is present and complies with the standard for this information.
Nfvo.VnfPkgm.004	The NFVO shall support the capability to notify about changes of the VNF Package.
Nfvo.VnfPkgm.005	The NFVO shall support the capability to validate the integrity and authenticity of the VNFD in the VNF Package.
Nfvo.VnfPkgm.006	The NFVO shall support the capability to notify about the on-boarding of the VNF Package.
Nfvo.VnfPkgm.007	The NFVO shall support the capability to request modifying the VNF instance information in the VNFM to refer to a different VNF Package when no conflicts exist between the previous and the newly referred VNF Package (see note 2).
Nfvo.VnfPkgm.008	The NFVO shall support the capability to allow on-boarding of different VNF Packages of a VNF.
NOTE 1: The VNF Packages management can include on-boarding, enable/disable, query, fetch and delete of VNF Packages.	
NOTE 2: A related use case is to keep NFV-MANO in sync about a VNF application software modification (see clause 5.7 of ETSI GS NFV-IFA 011 [i.19]).	

6.5.2 Functional requirements for VNF instance information management

Table 6.5.2-1: Functional requirements for VNF instance information management

Numbering	Functional requirements description
Nfvo.VnfIIm.001	The NFVO shall support the capability to query information on the mapping relationship between the VNF instance(s) and associated virtualised resource.

6.6 Functional requirements for NS information management

6.6.1 Functional requirements for NSD management

Table 6.6.1-1: Functional requirements for NSD management

Numbering	Functional requirements description
Nfvo.NsDtm.001	The NFVO shall support the capability of management of NSD (see note).
Nfvo.NsDtm.002	The NFVO shall support the capability to verify the integrity of the provided NSD.
Nfvo.NsDtm.003	The NFVO shall support the capability to verify that all mandatory information in the NSD is present and complies with the standard for this information.
Nfvo.NsDtm.004	The NFVO shall support the capability to report information related to the operation result of NSD.
Nfvo.NsDtm.005	The NFVO shall support the capability to perform version control of on-boarded NSDs.
Nfvo.NsDtm.006	The NFVO shall support the capability to notify about the on-boarding of the NSD.
Nfvo.NsDtm.007	The NFVO shall support the capability to notify about the changes of the NSD.
Nfvo.NsDtm.008	The NFVO shall support the capability to notify about the deletion of the NSD.
NOTE: The NSD management can include on-boarding, update, enable/disable, query, fetch and delete of NSD.	

6.6.2 Functional requirements for NS instance information management

Table 6.6.2-1: Functional requirements for NS instance information management

Numbering	Functional requirements description
Nfvo.NsIim.001	The NFVO shall support the capability to receive run-time data related to NS instances that it has created (see note).
NOTE: Run-time data of NS instance can be information related to the run-time virtualised resource allocated to an NS instance, such as performance measurements related to resources of this instance or the VNF instance within this NS instance, resource reservation information for NFVI resources reserved for this NS instance, etc.	

6.6.3 Functional requirements for PNF Descriptor (PNFD) archive management

Table 6.6.3-1: Functional requirements for PNFD archive management

Numbering	Functional requirements description
Nfvo.PnfDtm.001	The NFVO shall support the capability of management of PNFD archives (see note).
Nfvo.PnfDtm.002	The NFVO shall support the capability to verify the integrity of the provided PNFD and the archive.
Nfvo.PnfDtm.003	The NFVO shall support the capability to verify that all mandatory information in the PNFD is present and complies with the standard for this information.
Nfvo.PnfDtm.004	The NFVO shall support the capability to report information related to the operation result of PNFD archive management.
Nfvo.PnfDtm.005	The NFVO shall support the capability to perform version control of on-boarded PNFD archive(s).
NOTE: The PNFD management can include on-boarding, update, query, fetch and delete of PNFD archive(s).	

6.7 Functional requirements for NS performance management

Table 6.7-1: Functional requirements for NS performance management

Numbering	Functional requirements description
Nfvo.NsPm.001	The NFVO shall support the capability of performance management of NSs.
Nfvo.NsPm.002	The NFVO shall support the capability to notify availability of performance information related to the NSs it manages (see note).
Nfvo.NsPm.003	In response to a query, the NFVO shall support the capability to provide the information about active PM jobs which match the filter criteria.
NOTE: Performance information related to a given NS results from either collected performance information of the virtualised resources impacting the connectivity of this NS instance or VNF related performance information issued by the VNFM for the VNFs that are part of this NS instance. The latter performance information also results from collected performance information of the virtualised resources that are mapped to this VNF instance.	

6.8 Functional requirements for VNF fault management

6.8.1 Functional requirements for virtualisation-related fault management

Table 6.8.1-1: Functional requirements for virtualisation-related fault management

Numbering	Functional requirements description
Nfvo.VirFm.001	The NFVO shall support the capability to request VNF healing to VNFM.
Nfvo.VirFm.002	The NFVO shall support the capability to collect notifications about alarms on a VNF instance as a consequence of state change in the virtualised resources used by the VNF.

6.9 Functional requirements for NS fault management

Table 6.9-1: Functional requirements for NS fault management

Numbering	Functional requirements description
Nfvo.NsFm.001	The NFVO shall support the capability to provide notifications of fault information related to the NSs it manages (see notes 1 and 2).
Nfvo.NsFm.002	The NFVO shall support the capability to provide fault information on the NSs it manages (see notes 1 and 2).
Nfvo.NsFm.003	The NFVO shall support the capability to provide notifications of changes in fault information related to the NSs it manages (see notes 1 and 2).
Nfvo.NsFm.004	The NFVO shall support the capability to perform automated or on-demand healing on the NSs it manages.
Nfvo.NsFm.005	The NFVO shall support the capability to notify the errors during NS lifecycle procedure.
Nfvo.NsFm.006	The NFVO shall support the capability to evaluate the impact on NS instance(s) it manages when NS healing needs to be performed on a component instance (i.e. a VNF or nested NS) shared or not.
NOTE 1: Fault information on a given NS results from either a collected virtualised resource fault impacting the connectivity of the NS instance or a VNF alarm (see clause 7.6) issued by the VNFM for a VNF that is part of this NS instance. NOTE 2: Fault information on a given NS instance can include the information related to the alarm (e.g. alarm created, alarm cleared, etc.), alarm causes and identification of this NS instance and fault information concerning the virtualised resources supporting the constituent VNFs for this NS instance and the virtualised resources supporting the connectivity of this NS instance.	

6.10 Functional requirements for infrastructure resource management

Table 6.10-1: Functional requirements for infrastructure resource management

Numbering	Functional requirements description
Nfvo.Irm.001	The NFVO shall support the capability to collect the information about NFVI-PoPs, such as network connectivity endpoints and geographical locations (see note).
NOTE: This information may be used by the NFVO for building and keeping NFVI-PoP topology information.	

6.11 Functional requirements for security consideration

Table 6.11-1: Functional requirements for security consideration

Numbering	Functional requirements description
Nfvo.Sc.001	The NFVO shall support the capability to validate that the received message is from an authenticated and authorized consumer.
Nfvo.Sc.002	The NFVO shall support the capability to verify the integrity of the received message.
Nfvo.Sc.003	The NFVO shall support the capability to encrypt the sent message or decrypt the received message using negotiated key and algorithm to or from an authenticated and authorized consumer or producer.

6.12 Functional requirements for software image management

NOTE: The software image(s) is/are at virtualisation container level, e.g. Virtual Machine (VM) or OS container images.

Table 6.12-1: Functional requirements for software image management

Numbering	Functional requirements description
Nfvo.Sim.001	The NFVO shall support the capability to distribute the VM software image(s) to one or more VIMs, as well as associated signatures and certificates. See note 2.
Nfvo.Sim.002	The NFVO shall support the capability to query the VIM for information on the VM software images.
Nfvo.Sim.003	The NFVO shall support the capability to invoke VM software image deletion request to VIM on those VM software image(s) which were distributed by the NFVO and managed by VIM.
Nfvo.Sim.004	The NFVO shall support the capability to invoke updating the user-defined metadata for the selected VM software images which were distributed by the NFVO and managed by VIM (see note 1).
Nfvo.Sim.005	The NFVO shall support the capability to ensure software image(s) isolation between the tenants.
Nfvo.Sim.006	The NFVO shall support the capability to distribute the OS container software image(s) to one or more CIRs.
Nfvo.Sim.007	The NFVO shall support the capability to query the CIR for information on the OS container software images.
Nfvo.Sim.008	The NFVO shall support the capability to invoke OS container software image deletion requests to the CIR on those OS container software image(s) which were distributed by the NFVO and managed by the CIR.
NOTE 1: The metadata may, but need not come from VNF Package. NOTE 2: Signatures of VM software images and their signing certificates shall be distributed in a format supported by the VIM. For some VIM implementations this can require the NFVO to extract the raw signature value from the signature file provided in the VNF Package and upload this to the VIM.	

6.13 Functional requirements for NFV acceleration management

Table 6.13-1: Functional requirements for NFV acceleration management

Numbering	Functional requirements description
Nfvo.NfvAm.001	When VNF-related Resource Management in indirect mode is applicable, the NFVO shall support the capability to request to the VIM the allocation and release of necessary acceleration resources to meet acceleration capability requirement(s) of the VNFs (see note).
Nfvo.NfvAm.002	The NFVO shall support the capability to retrieve acceleration capability requirement(s) of the VNF from the VNFD (see note).
Nfvo.NfvAm.003	The NFVO shall support the capability to receive acceleration capability information from VIM (see note).
Nfvo.NfvAm.004	The NFVO shall support the capability to query acceleration capability information from VIM (see note).
Nfvo.NfvAm.005	The NFVO shall support the capability to select a VIM that has enough available acceleration capabilities to support acceleration capability requirement(s) of the VNF (see note).
NOTE: The acceleration capabilities can include type, capacity, Non-Uniform Memory Architecture (NUMA) support, etc.	

6.14 Functional requirements for multi-tenancy

Table 6.14-1: Functional requirements for multi-tenancy

Numbering	Functional requirements description
Nfvo.Mtm.001	The NFVO shall support the capability of management of NS tenants (see note 1).
Nfvo.Mtm.002	The NFVO may support the capability of management of infrastructure tenants (see note 1) and mapping of such infrastructure tenants to the VIM managed infrastructure tenants in case VNF-related resource management in indirect mode is applicable.
Nfvo.Mtm.003	The NFVO shall support the capability to assign on-boarded VNF Packages and NSDs to one or more NS tenants (see note 2).
Nfvo.Mtm.004	The NFVO shall support the capability to on-board VNF Packages and NSDs for a tenant.
Nfvo.Mtm.005	The NFVO shall allow a tenant to instantiate VNFs and NSs using VNF Packages and NSD s assigned to this tenant or shared VNF Packages and NSDs.
Nfvo.Mtm.006	The NFVO shall support the capability to limit the scope of operations only to the service and infrastructure resource groups assigned to the requesting tenant.
Nfvo.Mtm.007	The NFVO shall support the capability to enable isolation between resources assigned to different tenants. See note 3.
NOTE 1: The management of tenants include: • create, read, update, delete tenants; • associate a tenant with a single or multiple consumers of the Os-Ma interface, defining also the role; • associate a tenant to a "service resource group", i.e. to a collection of NSs; • associate a tenant to "infrastructure resource group" managed by a VIM or to multiple "infrastructure resource groups" managed by different VIMs; • manage the association of a tenant and a VNFM if a VNF specific VNFM is used. NOTE 2: A VNF Package or NSD which is assigned to a single tenant is commonly referred to as a private VNF Package or NSD of this tenant. A VNF Package or NSD which is assigned to all tenants is commonly referred to as a public VNF Package or NSD. A VNF Package or NSD which is assigned to more than one tenant is commonly referred to as a shared VNF Package or NSD. NOTE 3: Isolation needs to be provided by the NFVI layer, and will be enabled by the multi-tenancy concept, see clause 5.2. For the isolation requirements see ETSI GS NFV 004 [i.3], requirements [Per.2], [Sec.1] and [Mod.6].	

6.15 Functional requirements for compute host reservation management

Table 6.15-1: Functional requirements for compute host reservation management

Numbering	Functional requirements description
Nfvo.Chrm.001	The NFVO shall support the capability to request the management of reservations at the compute host level to corresponding VIM(s) (see note).
Nfvo.Chrm.002	The NFVO shall support the capability to receive notifications related to changes of the state and/or capabilities of the reserved compute host(s).
NOTE: The management includes the creation, update, query and termination of compute host reservation(s).	

6.16 Functional requirements for policy management

Table 6.16-1: Functional requirements for policy management

Numbering	Functional requirements description
Nfvo.Plcm.001	The NFVO shall support the capability to manage NFV-MANO policies (see notes 1 and 2).
Nfvo.Plcm.002	The NFVO shall support the capability to report the conflicted NFV-MANO policies it detects (see note 3).
Nfvo.Plcm.003	The NFVO shall support the capability to resolve conflicts in the NFV-MANO polices it creates (see note 2).
Nfvo.Plcm.004	The NFVO shall support the capability to enforce NFV-MANO policies.
NOTE 1: This includes consuming operations to transfer, delete, update, query, activate, deactivate, associate and disassociate NFV-MANO policies. NOTE 2: NFV-MANO polices managed by the NFVO include policies applied in NS lifecycle management (instantiation, scaling, healing and termination), policies applied in VNF lifecycle management (instantiation, scaling, healing and termination) and polices applied in virtualised resource management (resource allocation, reservation, quota management and capacity management). The procedures also include associating and disassociating the policy with/from corresponding VNF instances, NS instances or resources. NOTE 3: The conflicted NFV-MANO policies include policies applied in NS lifecycle management (instantiation, scaling, update, healing and termination).	

6.17 Functional requirements for management of network services in a multiple administrative domain environment

Table 6.17-1: Functional requirements for management of network services in a multiple administrative domain environment

Numbering	Functional requirements description
Nfvo.Madm.001	The NFVO shall support the capability to invoke NS lifecycle operation granting towards the NFVO in another administrative domain.
Nfvo.Madm.002	The NFVO shall support the capability to receive invocations of NS lifecycle operation granting from the NFVO in another administrative domain.
Nfvo.Madm.003	The NFVO shall support the capability to invoke the instantiation of a nested NS towards the NFVO in another administrative domain.
Nfvo.Madm.004	The NFVO shall support the capability to invoke the scaling of a nested NS towards the NFVO in another administrative domain.
Nfvo.Madm.005	The NFVO shall support the capability to invoke the healing of a nested NS towards the NFVO in another administrative domain.
Nfvo.Madm.006	The NFVO shall support the capability to query information related to a nested NS from the NFVO in another administrative domain.
Nfvo.Madm.007	The NFVO shall support the capability to request the creation and deletion of the identifier of a nested NS from the NFVO in another administrative domain.
Nfvo.Madm.008	The NFVO shall support the capability to invoke the termination of a nested NS towards the NFVO in another administrative domain.
Nfvo.Madm.009	The NFVO shall support the capability to reject a request from an NFVO in another administrative domain to terminate an NS if this NS is in use or if determined by network service provider's policies.
Nfvo.Madm.010	The NFVO shall support the capability to identify that an instance of an NS that it manages is no longer used as a constituent nested NS of a composite NS managed by itself or by other NFVO in other administrative domains. See note.
Nfvo.Madm.011	The NFVO shall support the capability to receive NS lifecycle change notifications related to a nested NS from the NFVO in another administrative domain.
Nfvo.Madm.012	The NFVO shall support the capability to query information related to an NSD from the NFVO in another administrative domain.
Nfvo.Madm.013	The NFVO shall support the capability to receive notifications about alarms and fault information related to a nested NS from the NFVO in another administrative domain.
Nfvo.Madm.014	The NFVO shall support the capability to request PM jobs operations and receive performance management information related to a nested NS from the NFVO in another administrative domain.
Nfvo.Madm.015	The NFVO shall support the capability to consider NS instance priorities while granting NS lifecycle operations from the NFVO in another administrative domain.
Nfvo.Madm.016	The NFVO shall support the capability to use NS instance priority information while invoking the instantiation or scaling of a nested NS towards the NFVO in another administrative domain.
NOTE: By knowing whether an instance of an NS is in use, the NFVO can determine whether to terminate the NS instance and delete its NS identifier according to the network service provider's policies.	

6.18 Functional requirements for management of connectivity for Multi-Site services

Table 6.18-1: Functional requirements for management and connectivity for Multi-Site services

Numbering	Functional requirements description
Nfvo.Mss.001	The NFVO shall support the capability to manage the lifecycle of NS across multiple NFVI-PoPs.
Nfvo.Mss.002	The NFVO shall support the capability to manage NS Virtual Link aggregation over a determined WAN virtualised network resource based on diverse operational policies. See note 1.
Nfvo.Mss.003	The NFVO shall support the capability to query and acquire information about the virtual network resources in the WAN.
Nfvo.Mss.004	The NFVO shall support the capability to handle alarm notifications about faulty virtualised network resources in the WAN. See notes 3 and 4.
Nfvo.Mss.005	The NFVO shall support the capability to prepare and request the allocation of virtualised network resources in the WAN in advance of their usage. See notes 3 and 4.
Nfvo.Mss.006	The NFVO shall support the capability to determine the required virtualised network resources in the WAN to meet the requirements for the NS Virtual Links based on the information provided in the NSD. See note 2.
Nfvo.Mss.007	The NFVO shall support the capability to orchestrate actions related to virtualised network resources among multiple NFVI-PoPs managed by one or more VIMs and/or WIMs.
Nfvo.Mss.008	The NFVO shall support the capability to update NS Virtual Links to be assigned a specific virtualised network resource in the WAN.
Nfvo.Mss.009	The NFVO shall support the capability to inform/notify the VNFM about changes/failures of connectivity on an NS Virtual Link impacting the connectivity of a VNF constituent of the NS and managed by the VNFM. See note 3.
Nfvo.Mss.010	The NFVO shall support the capability to request the VNFM to connect/disconnect a specific external connection point of a VNF. See note 3.
Nfvo.Mss.011	The NFVO shall support the capability to determine the required virtualised network resources in the WAN to meet the requirements for the multi-site deployment of a VNF based on the information provided in the VNFD and/or received via interfaces.
Nfvo.Mss.012	The NFVO shall support the capability to query and acquire information about the connectivity support between NFVI-PoPs. See note 5.
Nfvo.Mss.013	The NFVO shall support the capability to manage VNF internal VL when the VL spans virtualised network resources of different NFVI-PoPs and across WAN.
Nfvo.Mss.014	The NFVO shall support the capability to orchestrate the acquisition and provisioning of information produced by a VIM/WIM about managed virtualised network resources for connecting to the virtualised network resource managed by other VIMs/WIMs.
NOTE 1: Operational policies can take different rules or criteria to determine reusing an existing virtualised network resource for the aggregation, such as: - tenancy of the new VL with respect to the already assigned VL; - the (group of) Network Service(s) or VNF(s) to which the new VL to be instantiated belongs to; - the (group of) connectivity types of the VL; - the QoS class of the VL; - the throughput requirements of the VL; and - affinity/anti-affinity rules specified in the NSD. NOTE 2: An example of a requirement is an affinity/anti-affinity constraint to ensure that NS VL are anti-affine in terms of physical WAN resources to fulfil certain redundancy requirements. NOTE 3: This is in support of management of Virtual Link redundancy among NFVI-PoPs. NOTE 4: This is in support of management of Virtual Link healing among NFVI-PoPs.	

NOTE 5: The information that can be collected includes (not an exhaustive list):
- Identification of the WAN(s)
- Identification of the WIM(s)
- List of NFVI-PoP connectivity endpoints that can be used
- Network types and connectivity types
- Network segments
- Network layering capabilities
- QoS, bitrate and capacity parameters for each connection
- Support of differentiation of data flows from different VL
- Support for traffic/data flows differentiation
- Geographical information
- Path distances, to estimate latencies
- Topology information

This information may be used by the NFVO for building and keeping topology information.

6.19 Functional requirements related to the support for network slicing

Table 6.19-1: Functional requirements related to the support for network slicing

Numbering	Functional requirements description
Nfvo.Slice.001	The NFVO shall support the capability to manage NS instances, taking into account priorities.
Nfvo.Slice.002	The NFVO shall support the capability to take in account NS instance priorities during all operations of resource management.
Nfvo.Slice.003	The NFVO shall support the capability to take in account NS instance priorities during all lifecycle management operations.
Nfvo.Slice.004	The NFVO shall support the capability to enable the isolation between the tenants. See note.
NOTE: Isolation needs to be provided by the NFVI layer, and will be enabled by the multi-tenancy concept, see clause 5.2. For the isolation requirements see ETSI GS NFV 004 [i.3], requirements [Per.2], [Sec.1] and [Mod.6].	

6.20 Functional requirements for VNF Snapshot Packages

Table 6.20-1: Functional requirements for VNF Snapshot Packages

Numbering	Functional requirements description
Nfvo.VnfSnap.001	The NFVO shall support the creation, building, uploading, extraction, deletion, fetching and update of a VNF Snapshot Package.
Nfvo.VnfSnap.003	The NFVO shall support the fetching of selected artifacts of a VNF Snapshot Package.
Nfvo.VnfSnap.003	The NFVO shall support the query of VNF/VNFC Snapshot Package Information.
Nfvo.VnfSnap.004	The VNFM shall have the capability to provide information about the VNF Snapshot Packages (see note).
NOTE: Information about a VNF Snapshot Package includes the location, content, and availability of the VNF Snapshot Packages.	

6.21 Functional requirements for OS container configuration management

Table 6.21-1: Functional requirements for OS container configuration management

Numbering	Functional requirements description
Nfvo.Osccm.001	The NFVO shall support the capability to request management operations from the CISM service interface for OS container configuration management.
Nfvo.Osccm.002	The NFVO shall support the capability to request the CISM to create a namespace.
Nfvo.Osccm.003	The NFVO shall support the capability to request the CISM to delete a namespace.
Nfvo.Osccm.004	The NFVO shall support the capability to query the CISM for information on namespaces managed by the CISM.
Nfvo.Osccm.005	The NFVO shall support the capability to request the CISM to create a namespace quota.
Nfvo.Osccm.006	The NFVO shall support the capability to request the CISM to modify a namespace quota.
Nfvo.Osccm.007	The NFVO shall support the capability to request the CISM to delete a namespace quota.
Nfvo.Osccm.008	The NFVO shall support the capability to query the CISM for information on namespace quota managed by the CISM.
Nfvo.Osccm.009	The NFVO shall support the capability to request the CISM to create policies for MCIOs.
Nfvo.Osccm.010	The NFVO shall support the capability to request the CISM to modify policies for MCIOs.
Nfvo.Osccm.011	The NFVO shall support the capability to request the CISM to replace policies for MCIOs.
Nfvo.Osccm.012	The NFVO shall support the capability to request the CISM to delete policies for MCIOs.
Nfvo.Osccm.013	The NFVO shall support the capability to query the CISM for information on policies for MCIOs managed by the CISM.
Nfvo.Osccm.014	The NFVO shall support the capability to request the CISM to create MCIO configurations. See note.
Nfvo.Osccm.015	The NFVO shall support the capability to request the CISM to modify MCIO configurations. See note.
Nfvo.Osccm.016	The NFVO shall support the capability to request the CISM to replace MCIO configurations. See note.
Nfvo.Osccm.017	The NFVO shall support the capability to request the CISM to delete MCIO configurations. See note.
Nfvo.Osccm.018	The NFVO shall support the capability to query the CISM for information on MCIO configurations managed by the CISM. See note.
NOTE: MCIO configurations referred in this requirement concern MCIOs within the scope of a container cluster.	

6.22 Functional requirements for MCIOP management

Table 6.22-1: Functional requirements for MCIOP management

Numbering	Functional requirements description
Nfvo.Mciopm.001	The NFVO shall support the capability to distribute MCIOP(s) to one or more MCIOP repositories.
Nfvo.Mciopm.002	The NFVO shall support the capability to query MCIOP repositories for information on MCIOP(s).
Nfvo.Mciopm.003	The NFVO shall support the capability to invoke MCIOP deletion requests to MCIOP repositories on those MCIOP(s) which were distributed by the NFVO and managed by the MCIOP repositories.

7 Functional requirements for VNFM

7.1 Functional requirements for virtualised resource management

7.1.1 Functional requirements for virtualised resource management

Table 7.1.1-1: Functional requirements for virtualised resource management

Numbering	Functional requirements description
Vnfm.Vrm.001	The VNFM shall support providing deployment-specific configuration information for virtualised resource related to VNF instance(s).
Vnfm.Vrm.002	The VNFM shall support the capability to maintain the mapping between a VNF instance and the virtualised resources of the VNF instance (see note 1).
Vnfm.Vrm.003	The VNFM shall support the capability to request resource allocation for VNF instance that meet the requirements specified by the VNF provider.
Vnfm.Vrm.004	The VNFM should support the capability to determine, based on the VNFD information, the resources needed to overcome the impact on currently allocated virtualised resources in response to an event received about upcoming NFVI impacts.
Vnfm.Vrm.005	VNFM shall support the capability to receive the constraints and policies applicable to virtualised resources allocated to the VNF (see note 2).
Vnfm.Vrm.006	The VNFM shall support the capability to provide notifications of changes in virtualised resources allocated to the VNFs it manages due to NFVI operation and maintenance.
NOTE 1: The VNFM maintains the mapping between virtualised resources and the VNF in order to, for example: - Map virtualised resources fault information, performance information and change notifications to corresponding VNFCs. - Request management of virtualised resources to support current instantiated VNFCs, instantiate new VNFCs, terminate existing instantiated VNFCs, and internal connectivity in the VNF (VLs and Connection Points (CPs)). NOTE 2: Constraints and policies related to virtualised resource(s) that can be impacted by NFVI maintenance activities or other operations.	

7.1.2 Functional requirements for VNF-related resource management in indirect mode

Table 7.1.2-1: Functional requirements for VNF-related resource management in indirect mode

Numbering	Functional requirements description
Vnfm.VnfRmpbNfvo.001	When VNF-related Resource Management in indirect mode is applicable, the VNFM shall support the capability to request to NFVO the management of virtualised resources needed for VNFs instantiation, scaling and termination (see note).
Vnfm.VnfRmpbNfvo.002	When VNF-related Resource Management in indirect mode is applicable, the VNFM shall support the capability to invoke resource management requests towards the NFVO to allocate resources that meet the requirements specified by the VNF provider.
NOTE: The management of virtualised resources includes allocation, update, scaling, termination, etc. of virtualised resources.	

7.1.3 Functional requirements for VNF-related resource management in direct mode

Table 7.1.3-1: Functional requirements for VNF-related resource management in direct mode

Numbering	Functional requirements description
Vnfm.VnfRmpbVnfm.001	When VNF-related Resource Management in direct mode is applicable, the VNFM shall support the capability to request to the VIM the management of virtualised resources needed for VNFs instantiation, scaling and termination (see notes 1 and 4).
Vnfm.VnfRmpbVnfm.002	When VNF-related Resource Management in direct mode is applicable, the VNFM shall support the capability to query to the VIM about the resources being allocated to VNF instances it manages (see note 1).
Vnfm.VnfRmpbVnfm.003	When VNF-related Resource Management in direct mode is applicable, the VNFM shall support the capability to receive notifications regarding the resources being allocated to or released from specific VNF instances, as well as regarding events and relevant fault reports related to those resources (see notes 1 and 3).
Vnfm.VnfRmpbVnfm.004	When VNF-related Resource Management in direct mode is applicable, the VNFM shall support the capability to request allocation and update of resources in the different resource commitment models (see notes 2 and 5).
Vnfm.VnfRmpbVnfm.005	When VNF-related Resource Management in direct mode is applicable, the VNFM shall support the capability to request to the VIM affinity and anti-affinity policies for the VNF's virtualised resources (see note 1).
Vnfm.VnfRmpbVnfm.006	When VNF-related Resource Management in direct mode is applicable and a resource reservation model is used, the VNFM shall support the capability to use resource reservation identification information obtained from the NFVO to request allocation of virtualised resources for a VNF.
Vnfm.VnfRmpbVnfm.007	When VNF-related Resource Management in direct mode is applicable, the VNFM shall support the capability to obtain appropriate information to enable the VNFM to access the VIM.
Vnfm.VnfRmpbVnfm.008	When VNF-related Resource Management in direct mode is applicable, the VNFM shall support the capability of providing the VIM with NFVI operation and maintenance constraints (expressed as policies) concerning the virtualised resources for the VNF.
Vnfm.VnfRmpbVnfm.009	When VNF-related Resource Management in direct mode is applicable, the VNFM shall support the capability to update the NFVI operation and maintenance constraints (expressed as policies) (see note 6) concerning the virtualised resources for the VNF.
NOTE 1: Virtual resources managed for the LCM of VNFs include compute and storage resources needed for VNF components as well as networking resources needed to ensure intra-VNF connectivity. NOTE 2: Resource commitment models are: reservation model, quota model and on-demand. NOTE 3: Events include NFVI outage, NFVI software modification and performance related events. NOTE 4: The management of virtualised resources includes allocation, update, scaling, termination, etc. of virtualised resources. NOTE 5: This does not imply that the VNFM can manage resource reservations and quotas, which are NFVO's prerogatives. NOTE 6: Updates shall be done in accordance with the constraints provided in the VNFD, if any.	

7.1.4 Functional requirements for resource reservation management

Table 7.1.4-1: Functional requirements for resource reservation management

Numbering	Functional requirements description
Vnfm.Rrm.001	The VNFM shall support the capability to receive change notification regarding virtualised resource reservation.
Vnfm.Rrm.002	The VNFM shall support the capability to query information regarding virtualised resource reservation.

7.1.5 Functional requirements for virtualised resource performance management

Table 7.1.5-1: Functional requirements for virtualised resource performance management

Numbering	Functional requirements description
Vnfm.Vrpm.001	The VNFM shall support the capability to invoke the virtualised resource performance management operations on the virtualised resources for the VNF/VNFC instance(s) it manages (see note 1).
Vnfm.Vrpm.002	The VNFM shall support the capability to receive performance information related to virtualised resources for the VNF/VNFC instance(s) it manages (see note 2).
Vnfm.Vrpm.003	The VNFM shall support the capability to map to the VNF instances the received performance information related to virtualised resources (see note 2).
NOTE 1: The virtualised resource performance management can include setting threshold conditions on the performance information collected by the VIM for specific virtualised resource(s), creating PM jobs by specifying different limitations and conditions for collecting and reporting of performance information from specified virtualised resource(s), etc.	
NOTE 2: The performance management operations mentioned in this requirement apply to performance measurements defined for measured object types applicable to the Vi-Vnfm reference point.	

7.1.6 Functional requirements for virtualised resource fault management

Table 7.1.6-1: Functional requirements for virtualised resource fault management

Numbering	Functional requirements description
Vnfm.Vrfm.001	The VNFM shall support the capability to collect fault information related to the virtualised resources allocated to VNF instance(s) that it manages.
Vnfm.Vrfm.002	The VNFM shall support the capability to correlate virtualised resource fault information with the impacted VNF(C) instance(s) that it manages.

7.1.7 Functional requirements for virtualised resource information management

Table 7.1.7-1: Functional requirements for virtualised resource information management

Numbering	Functional requirements description
Vnfm.Vrim.001	The VNFM should support the capability to query information regarding consumable virtualised resources that can be provided by the VIM.
Vnfm.Vrim.002	The VNFM shall support the capability to receive the notifications regarding the changes of the information on consumable virtualised resources that can be provided by the VIM.

7.1.8 Functional requirements for quota management

Table 7.1.8-1: Functional requirements for quota management

Numbering	Functional requirements description
Vnfm.Qm.001	The VNFM should support the capability to query the information on the quota(s) that apply to this VNFM or to the VNF(s) that this VNFM manages.
Vnfm.Qm.002	The VNFM should support the capability to receive change notification regarding the quota constraint(s) that apply to this VNFM or to the VNF that this VNFM manages.
Vnfm.Qm.003	The VNFM may support the capability to receive information from NFVO on available quota(s) applicable to this VNFM (see notes 1 and 2).
NOTE 1: The information on available quota(s) allows the VNFM to interact with the VIM to receive information regarding the quota(s) applied to the VNFM or the VNF(s) which the VNFM manages, when VNF-related Resource Management in direct mode is applicable.	
NOTE 2: The information on available quota(s) allows the VNFM to interact with the NFVO to receive information regarding the quota(s) applied to the VNFM or the VNF(s) which the VNFM manages, when VNF-related Resource Management in indirect mode is applicable.	

7.1.9 Functional requirements related to permitted allowance management

Table 7.1.9-1: Functional requirements related to permitted allowance management

Numbering	Functional requirements description
Vnfm.Pam.001	When an allowance model is used, the VNFM shall support the capability to notify its resource consumption.

7.2 Functional requirements for VNF lifecycle management

7.2.1 Functional requirements for VNF lifecycle management

NOTE: Not all VNFs support all the VNF lifecycle operations which associate with the capabilities defined in the present document. For any given VNF, the VNFM will only be able to perform those operations that are supported by that VNF.

Table 7.2.1-1: Functional requirements for VNF lifecycle management

Numbering	Functional requirements description
Vnfm.VnfLcm.001	The VNFM shall support the capability to notify about the following events related to VNF lifecycle changes: • the start of the lifecycle procedure; • the end and the result of the lifecycle procedure, including errors during the procedure, if any.
Vnfm.VnfLcm.002	The VNFM shall support the capability to notify about the type of VNF lifecycle change, the addition/deletion of VNFCs, and about the changes on virtualised resources associated to VNFC(s) as result of the VNF lifecycle change.
Vnfm.VnfLcm.003	The VNFM shall support the capability to notify about virtual networks and CPs that are added/deleted as part of the VNF lifecycle operation.
Vnfm.VnfLcm.004	The VNFM shall support the capability to validate the lifecycle operation requests it processes, using information specified in the VNF Package.
Vnfm.VnfLcm.005	The VNFM shall support the capability to change the state of a VNF instance/VNFC instance(s) (see note 1).
Vnfm.VnfLcm.006	The VNFM shall support the capability to use the deployment information from the VNFD for the VNF LCM.
Vnfm.VnfLcm.007	The VNFM shall support the capability to provide the status of a VNF LCM operation in response to a query.
Vnfm.VnfLcm.008	The VNFM shall support the capability to request an operation granting before executing the VNF lifecycle operation procedure, in procedures that can require changes in terms of virtualised resources usage, or require changes in terms of namespace quota (when the VDUs of the VNF are realized by a set of OS containers) or impact NS management (see note 2).
Vnfm.VnfLcm.009	The VNFM shall support the capability to switch the DF of a VNF instance.
Vnfm.VnfLcm.010	The VNFM shall support the capability to create and delete the identifier of the VNF instance which it manages.
Vnfm.VnfLcm.011	The VNFM shall support the capability to conduct VNF error handling operation(s) after the VNF life cycle operation occurrence fails (see notes 3 and 4).
NOTE 1: Change state refers to start and stop a VNF instance/VNFC instance(s). These operations are complementary to instantiate/create a VNF, or terminate a VNF. NOTE 2: This includes procedures related to instantiation, scaling, healing, and termination of VNF instances. NOTE 3: It is up to the protocol design stage to design the detail error handling operation(s). NOTE 4: It depends on the VNF capabilities and is declared in the VNFD whether and how the operation(s) are supported by a particular VNF.	

7.2.2 Functional requirements for VNF instantiation

Table 7.2.2-1: Functional requirements for VNF instantiation

Numbering	Functional requirements description
Vnfm.VnfI.001	The VNFM shall support the capability to manage the instantiation of a VNF instance.
Vnfm.VnfI.002	The VNFM shall support the capability to request VIM to allocate resources for the VNF instance being instantiated.
Vnfm.VnfI.003	The VNFM shall support the capability to configure deployment specific parameters for the VNF instance being instantiated.
Vnfm.VnfI.004	The VNFM shall support the capability to store the information of the allocated resources and configured deployment specific parameters for the instantiated VNF.

7.2.3 Functional requirements for VNF scaling

NOTE: The LCM operations that expand or contract a VNF instance include scale in, scale out, scale up, scale down. Not all VNFs support all these operations, which implies that the set of operations that a VNFM will be able to perform on a VNF instance will depend on the VNF capabilities.

Table 7.2.3-1: Functional requirements for VNF scaling

Numbering	Functional requirements description
Vnfm.VnfS.001	The VNFM shall support the capability to manage the expansion of the capacity of a VNF instance (see note 1).
Vnfm.VnfS.002	The VNFM shall support the capability to manage the contraction of the capacity of a VNF instance (see note 2).
Vnfm.VnfS.003	The VNFM shall support the capability to manage the scaling out/in of a VNF instance in order to perform expansion/contraction.
Vnfm.VnfS.004	The VNFM shall support the capability to expand/contract a VNF instance based on a request from the VNF instance or its EM if it exists.
Vnfm.VnfS.005	The VNFM shall support the capability to expand/contract a VNF instance based on a request from NFVO.
Vnfm.VnfS.006	The VNFM should support the capability to monitor the state of a VNF instance and trigger its expansion/contraction when certain conditions are met.
NOTE 1: Expansion can either be performed by scaling out or scaling up, but only the former is required in the present release. NOTE 2: Contraction can either be performed by scaling in or scaling down, but only the former is required in the present release.	

7.2.4 Functional requirements for VNF termination

Table 7.2.4-1: Functional requirements for VNF termination

Numbering	Functional requirements description
Vnfm.VnfT.001	The VNFM shall support the capability to terminate a VNF instance.

7.2.5 Functional requirements for changing the current VNF Package

Table 7.2.5-1: Functional requirements for changing the current VNF Package

Numbering	Functional requirements description
Vnfm.VnfSwm.001	The VNFM shall have the capability to support changing the current VNF Package. See notes 1 and 2.
Vnfm.VnfSwm.002	The VNFM shall support the capability to manage the instantiation of VNFC instances of a particular software version within the VNF instance being upgraded.
NOTE 1: The support of changing the current VNF Package includes handling the software images and the required resource related aspects. NOTE 2: The "change current VNF Package" includes updates and upgrades of the software of VNFs and VNFCs.	

7.2.6 Functional requirements for change of the external VNF connectivity

Table 7.2.6-1: Functional requirements for change of the external VNF connectivity

Numbering	Functional requirements description
Vnfm.VnfCC.001	The VNFM shall support the capability to change the external connectivity of a VNF instance. See note.
NOTE: Changing the external connectivity may imply connecting the VNF to an additional network or network segment or to disconnect the VNF from a network or network segment to which it was previously connected.	

7.3 Functional requirements for VNF configuration management

Configuration parameters referred in this clause include those set at initial configuration and any other configurable parameter declared in the VNFD.

Table 7.3-1: Functional requirements for VNF configuration management

Numbering	Functional requirements description
Vnfm.VnfCm.001	The VNFM shall support the capability to set initial configuration parameters for a VNF/VNFC instance.
Vnfm.VnfCm.002	The VNFM shall support the capability to update configuration parameters for a VNF/VNFC instance.

7.4 Functional requirements for VNF information management

7.4.1 Functional requirements for VNF Package management

Table 7.4.1-1: Functional requirements for VNF Package management

Numbering	Functional requirements description
Vnfm.VnfPkgm.001	The VNFM shall support the capability to obtain details of available VNF Packages of the VNFs which it manages (see note).
Vnfm.VnfPkgm.002	The VNFM should support the capability to receive notifications as a result of on-boarding of VNF Packages (see note).
Vnfm.VnfPkgm.003	The VNFM should support the capability to receive notifications as a result of changes on VNF Package states (see note).
NOTE: Information about the availability, content and current state of VNF Packages is needed for the VNFM to validate the lifecycle operation requests (refer to requirement Vnfm.VnfLcm.004), and to perform the lifecycle management operations.	

7.4.2 Functional requirements for VNF instance information management

Table 7.4.2-1: Functional requirements for VNF instance information management

Numbering	Functional requirements description
Vnfm.VnfIim.001	The VNFM shall support the capability to receive run-time data related to VNF instances that it has created (see note 1).
Vnfm.VnfIim.002	The VNFM shall support the capability to provide information on the mapping relationship between the VNF instance(s) and associated virtualised resource in response to the query.
Vnfm.VnfIim.003	The VNFM shall support the capability to modify the VNF instance information to refer to a different VNF Package (see note 2).
Vnfm.VnfIim.004	The VNFM shall support the capability to modify information about a VNF instance.
NOTE 1: Run-time data of VNF instance can be information from VIM related to the virtualised resource allocated to a run-time VNF instance, such as VNF instance address, record of significant VNF lifecycle event, etc. NOTE 2: A related use case is to keep NFV-MANO in sync about a VNF application software modification (see clause 5.7 of ETSI GS NFV-IFA 011 [i.19]).	

7.5 Functional requirements for VNF performance management

Table 7.5-1: Functional requirements for VNF performance management

Numbering	Functional requirements description
Vnfm.VnfPm.001	The VNFM shall support the capability to notify the availability of VNF/VNFC performance information, resulting from virtualised resources performance information, related to the VNFs/VNFCs it manages (see note).
NOTE: Performance information on a given VNF/VNFC results from collected performance information of the virtualised resources that are mapped to this VNF/VNFC instance.	

7.6 Functional requirements for VNF fault management

7.6.1 Functional requirements for virtualised resource-related VNF fault management

Table 7.6.1-1: Functional requirements for virtualised resource-related VNF fault management

Numbering	Functional requirements description
Vnfm.VrVnfFm.001	The VNFM shall support the capability to provide notifications of virtualised resource-related fault information on the VNFs it manages (see notes 1 and 2).
Vnfm.VrVnfFm.002	The VNFM shall support the capability to provide virtualised resource-related fault information on the VNFs it manages (see notes 1 and 2).
Vnfm.VrVnfFm.003	The VNFM shall support the capability to provide notifications of changes in virtualised resource-related fault information related to the VNFs it manages (see notes 1 and 2).
Vnfm.VrVnfFm.004	The VNFM shall support the capability to notify about alarms on VNF and any of its VNFCs as a consequence of state changes in the virtualised resources used by the VNF and its VNFCs.
Vnfm.VrVnfFm.005	The VNFM shall support the capability to request corrective operations on virtualised resources to VIM in order to perform VNF healing.
Vnfm.VrVnfFm.006	The VNFM shall support the capability to assign unique identifiers to the virtualised resource-related fault information on the VNFs it manages (see note 3).
Vnfm.VrVnfFm.007	The VNFM shall support the capability to keep the alarm record(s) in the alarm list unless the criteria (see note 4) is met.
NOTE 1: Virtualised resource-related fault information on a given VNF results from a collected virtualised resource fault impacting the corresponding VNF/VNFC instance. NOTE 2: Virtualised resource-related fault information on a given VNF instance can includes the information related to the alarm (e.g. alarm created, alarm cleared, etc.), alarm causes and identification of this VNF instance and fault details related to the virtualised resources allocated to this VNF/VNFC instance. NOTE 3: Two alarms that are produced by the same VNFM cannot have the same identifier. NOTE 4: The criteria to be met before an alarm record can be removed from alarm list is: the alarm acknowledgement state is "acknowledged" and the perceived severity is "cleared".	

7.6.2 Functional requirements for virtualisation-related fault management

Table 7.6.2-1: Functional requirements for virtualisation-related fault management

Numbering	Functional requirements description
Vnfm.VirFm.001	The VNFM shall support the capability to perform on-demand VNF healing on the VNF(s) it manages.
Vnfm.VirFm.002	The VNFM shall support the capability to perform automated VNF healing on the VNF(s) it manages.

7.7 Functional requirements for security consideration

Table 7.7-1: Functional requirements for security consideration

Numbering	Functional requirements description
Vnfm.Sc.001	The VNFM shall support the capability to validate that the received message is from an authenticated and authorized consumer.
Vnfm.Sc.002	The VNFM shall support the capability to verify the integrity of the received message.
Vnfm.Sc.003	The VNFM shall support the capability to encrypt the sent message or decrypt the received message using negotiated key and algorithm to or from an authenticated and authorized consumer or producer.

7.8 Functional requirements for software image management

NOTE: The software image(s) is/are at virtualisation container level, e.g. Virtual Machine (VM) or OS container images.

Table 7.8-1: Functional requirements for software image management

Numbering	Functional requirements description
Vnfm.Sim.001	The VNFM shall support the capability to query the VIM for information of the VM software images.
Vnfm.Sim.002	The VNFM shall support the capability to query the CIR for information of the OS container software images.

7.9 Functional requirements for NFV acceleration management

Table 7.9-1: Functional requirements for NFV acceleration management

Numbering	Functional requirements description
Vnfm.NfvAm.001	When VNF-related Resource Management in direct mode is applicable, the VNFM shall support the capability to request to the VIM the allocation and release of necessary acceleration resources to meet the acceleration capability requirement(s) of the VNFs (see note).
Vnfm.NfvAm.002	When VNF-related Resource Management in indirect mode is applicable, the VNFM shall support the capability to request to the NFVO the allocation and release of necessary acceleration resources to meet the acceleration capability requirement(s) of the VNFs (see note).
Vnfm.NfvAm.003	The VNFM shall support the capability to retrieve acceleration capability requirement(s) of the VNF from the VNFD (see note).
NOTE: The acceleration capabilities can include type, capacity, NUMA support, etc.	

7.10 Functional requirements for multi-tenancy

Table 7.10-1: Functional requirements for multi-tenancy

Numbering	Functional requirements description
Vnfm.Mtm.001	When a VNFM supports multi-tenancy it shall support the capability of management of VNF tenants (see note).
Vnfm.Mtm.002	The VNFM shall support the capability to limit the scope of operations only to the service resource groups assigned to the requesting VNF tenant.
NOTE: A VNFM may be private for a tenant or it can support multi-tenancy. The management of tenants for a VNFM supporting multi-tenancy include: - create, read, update, delete tenants; - associate a tenant to a "service resource group", i.e. to a collection of VNFs; - associate a tenant to "infrastructure resource groups" managed by a VIM or to multiple "infrastructure resource groups" managed by different VIMs.	

7.11 Functional requirements for VNF indicator management

Table 7.11-1: Functional requirements for VNF indicator management

Numbering	Functional requirements description
VNFM_NFV_IND.001	The VNFM shall support the capability to receive notifications of VNF indicator value changes for the VNFs it manages (see note).
VNFM_NFV_IND.002	The VNFM shall support the capability to retrieve VNF indicator values, for the VNFs it manages, from the corresponding VNF/EM (see note).
NOTE: Indicators are information supplied by the VNF or the EM to provide some indication on the VNF behaviour. VNFM can use these indicators in conjunction with virtualised resource data to perform auto-scaling decisions.	

7.12 Functional requirements for policy management

Table 7.12-1: Functional requirements for policy management

Numbering	Functional requirements description
Vnfm.Plcm.001	The VNFM shall support the capability to manage NFV-MANO policies (see notes 1 and 2).
Vnfm.Plcm.002	The VNFM shall support the capability to report the conflicted NFV-MANO policies it detects (see note 3).
Vnfm.Plcm.003	The VNFM shall support the capability to enforce NFV-MANO policies.
NOTE 1: This includes consuming operations to transfer, delete, update, query, activate, deactivate, associate and disassociate NFV-MANO policies. NOTE 2: NFV-MANO polices managed by the VNFM include policies applied in virtualised resource management (resource allocation). NOTE 3: The conflicted NFV-MANO policies include policies applied in VNF lifecycle management (instantiation, scaling, healing and termination).	

7.13 Functional requirements for VNF/VNFC Snapshots

Table 7.13-1: Functional requirements for VNF/VNFC Snapshots

Numbering	Functional requirements description
Vnfm.VnfSnap.001	The VNFM shall support the identification of parameters for VNF/VNFC Snapshot operations (see notes 1 and 2).
Vnfm.VnfSnap.002	The VNFM shall support resolving VNFC Snapshots for VNF Snapshot creation and reversion (see note 3).
Vnfm.VnfSnap.003	The VNFM shall support the creation of and reversion to a VNFC Snapshot.
Vnfm.VnfSnap.004	The VNFM shall support the creation of and reversion to a VNF Snapshot.
Vnfm.VnfSnap.005	Void.
Vnfm.VnfSnap.006	Void.
Vnfm.VnfSnap.007	Void.
Vnfm.VnfSnap.008	The VNFM shall have the capability to provide information about the VNF/VNFC Snapshots (see note 4).
Vnfm.VnfSnap.009	Void.
NOTE 1: VNF/VNFC Snapshot operations include VNF/VNFC Snapshot creation and reversion. NOTE 2: The parameters may be included in incoming requests or provided by means of VNF/VNFC Snapshot descriptors. NOTE 3: Resolving VNFC Snapshots denotes the identification of those VNFCs and their elements to be included in/reverted for the VNF Snapshot. NOTE 4: Information about a VNF/VNFC Snapshot includes the location, content, and availability of the VNF/VNFC Snapshots.	

7.14 Functional requirements for management of connectivity for Multi-Site services

Table 7.14-1: Functional requirements for management and connectivity for Multi-Site services

Numbering	Functional requirements description
Vnfm_Mss.001	The VNFM shall support the capability to perform multi-site VNF deployment.
Vnfm_Mss.002	The VNFM shall support the capability to connect/disconnect a specific external connection point of a VNF.

7.15 Functional requirements for containerized workload management

7.15.1 Functional requirements for management of containerized workloads based on MCIOPs

Table 7.15.1-1: Functional requirements for management of containerized workloads based on MCIOPs

Numbering	Functional requirements description
Vnfm.Mciop.001	The VNFM shall support the capability to request management operations from the CISM service interface for containerized workloads based on MCIOPs.
Vnfm.Mciop.002	The VNFM shall support the capability to utilize information obtained from VNF descriptors and information received via interface requests as input to requests for the management of containerized workloads based on MCIOPs towards the CISM.
Vnfm.Mciop.003	The VNFM shall support the capability to request the CISM to instantiate containerized workloads based on MCIOPs.
Vnfm.Mciop.004	The VNFM shall support the capability to query the CISM for information on containerized workloads based on MCIOPs managed by the CISM.
Vnfm.Mciop.005	The VNFM shall support the capability to request the CISM to modify containerized workloads based on MCIOPs.
Vnfm.Mciop.006	The VNFM shall support the capability to request the CISM to terminate containerized workloads based on MCIOPs.

7.15.2 Functional requirements for MCIO management

Table 7.15.2-1: Functional requirements for MCIO management

Numbering	Functional requirements description
Vnfm.Mcio.001	The VNFM shall support the capability to request MCIO management operations from the CISM service interfaces for compute/storage/network management.
Vnfm.Mcio.002	The VNFM shall support the capability to request the CISM to create MCIOs whose declarative descriptors specify compute/storage/network infrastructure resource requests.
Vnfm.Mcio.003	The VNFM shall support the capability to query the CISM for information about the actual and desired state of MCIOs whose declarative descriptors specify compute/storage/network resource requests.
Vnfm.Mcio.004	The VNFM shall support the capability to request the CISM to modify the desired state of MCIOs whose declarative descriptors specify compute/storage/network infrastructure resource requests.
Vnfm.Mcio.005	The VNFM shall support the capability to request the CISM to modify the actual state of MCIOs whose declarative descriptors specify compute/storage/network infrastructure resource requests.
Vnfm.Mcio.006	The VNFM shall support the capability to request the CISM to replace MCIOs whose declarative descriptors specify compute/storage/network infrastructure resource requests.
Vnfm.Mcio.007	The VNFM shall support the capability to request the CISM to delete MCIOs whose declarative descriptors specify compute/storage/network infrastructure resource requests.
Vnfm.Mcio.008	The VNFM shall support the capability to request the CISM to list MCIOs whose declarative descriptors specify compute/storage/network resource requests.
Vnfm.Mcio.009	The VNFM shall support the capability to receive notifications from the CISM on events of changes of the desired or actual state of MCIOs whose declarative descriptors specify compute/storage/network resource requests.

7.15.3 Functional requirements for OS container configuration management

Table 7.15.3-1: Functional requirements for OS container configuration management

Numbering	Functional requirements description
Vnfm.Osccm.001	The VNFM shall support the capability to request management operations from the CISM service interface for OS container configuration management.
Vnfm.Osccm.002	The VNFM shall support the capability to request the CISM to create MCIO configurations. See note.
Vnfm.Osccm.003	The VNFM shall support the capability to request the CISM to modify MCIO configurations. See note.
Vnfm.Osccm.004	The VNFM shall support the capability to request the CISM to replace MCIO configurations. See note.
Vnfm.Osccm.005	The VNFM shall support the capability to request the CISM to delete MCIO configurations. See note.
Vnfm.Osccm.006	The VNFM shall support the capability to query the CISM for information on MCIO configurations managed by the CISM. See note.
NOTE: MCIO configurations referred in this requirement concern MCIOs within the scope of a containerized VNF.	

8 Functional requirements for VIM

8.1 General considerations

The following statement on the scope of VIM applies to all VIM related requirements:

- The VIM is responsible for controlling and managing the NFVI compute, storage and network resources of an operator's NFVI-PoP or a subset thereof (see note).

NOTE: This does not limit the possibility of VIM implementations capable of managing multiple NFVI-PoPs in any way.

8.2 Functional requirements for virtualised resource management

8.2.1 Functional requirements for virtualised resource management

Table 8.2.1-1: Functional requirements for virtualised resource management

Numbering	Functional requirements description
Vim.Vrm.001	The VIM shall support NFVI resource management within its area of responsibility (see note 1).
Vim.Vrm.002	The VIM shall support the capability of resource reservation management (see note 2).
Vim.Vrm.003	The VIM shall support the capability of quota based resource management.
Vim.Vrm.004	The VIM shall support the capability to correlate allocated and reserved virtualised resources with changes on underlying hardware/software resources due to maintenance, operation and management of the NFVI, and change the state of the allocated and reserved virtualised resources accordingly.
Vim.Vrm.005	The VIM shall support the capability to notify changes about allocated and reserved virtualised resources (see note 3).
Vim.Vrm.006	The VIM shall support the capability to enforce affinity and anti-affinity policies for NFVI resource management (see note 4).
Vim.Vrm.007	VIM shall support the capability to receive the virtualised resource management requests from VNFM and/or NFVO, and conduct the corresponding resource management operations.
Vim.Vrm.008	The VIM should support the capability of providing virtualised resources in compensation for those being impacted by NFVI operation and maintenance.
Vim.Vrm.009	The VIM shall support the capability of receiving information about the constraints and policies applicable to virtualised resources and their groups (see note 5).
NOTE 1: NFVI resource management includes allocation, termination, update, etc. of virtualised resources. NOTE 2: The management can include the creation, update, query and termination of resource reservation(s). NOTE 3: Notifications refer to individual resources and to groups of resources. For planned events they shall be sent in advance before the operation. NOTE 4: The policies include constraints on the number of group members that may be impacted simultaneously by planned events and the minimum time between consecutive impacts. NOTE 5: Constraints and policies related to virtualised resource(s) that can be impacted by NFVI maintenance activities or other operations.	

8.2.2 Functional requirements for resource reservation management

Table 8.2.2-1: Functional requirements for resource reservation management

Numbering	Functional requirements description
Vim.Rrm.001	The VIM shall support the capability to manage resources according to different resource commitment models, as follows: • Reservation model. • Quota model. • On demand.
Vim.Rrm.002	When a reservation model is used, the VIM shall support the capability to ensure that resources are allocated or updated from a resource reservation, when processing virtualised resource allocation or update requests.
Vim.Rrm.003	When a reservation model is used, the VIM shall support the capability to infer information about what reservation is applicable by using input information received with the allocation or update request.
Vim.Rrm.004	When a reservation model is used and explicit reservation identification is indicated, the VIM shall support the capability to use such information to map to the applicable resource reservation.
Vim.Rrm.005	When a reservation model is used and explicit reservation identification is not indicated, the VIM shall support the capability to map to the applicable reservation by using other information such as consumer/tenant identification (see note).
Vim.Rrm.006	The VIM shall support the capability to consider affinity/anti-affinity rules for resource reservation management.
Vim.Rrm.007	The VIM shall support the capability to notify the change regarding to virtualised resource reservation.
NOTE: In this case of so-called "implicit reservation identification", the reservation identified has been reserved by the NFVO as a single bulk of resources, and successive allocations consume from that bulk.	

8.2.3 Functional requirements for virtualised resource and NFVI capacity management

Table 8.2.3-1: Functional requirements for virtualised resource capacity management

Numbering	Functional requirements description
Vim.Vrcm.001	The VIM shall support the capability to collect and maintain information regarding the capacity of the NFVI it manages.
Vim.Vrcm.002	The VIM shall support the capability to provide information related to available, allocated, reserved and all virtualised resource capacity.
Vim.Vrcm.003	The VIM shall support the capability to provide the notification of the change(s) related to the capacity of the virtualised resource which are managed by it.
Vim.Vrcm.004	The VIM shall support the capability to provide information about NFVI-PoP(s) it administers, such as network connectivity endpoints and geographical location.
Vim.Vrcm.005	The VIM shall support the capability to provide information about resource zones in the NFVI that it manages.

Table 8.2.3-2: Functional requirements for NFVI capacity management

Numbering	Functional requirements description
Vim-Ncm.001	The VIM shall support the capability to provide information related to available, allocated, reserved and total NFVI capacity (including compute hosts).
Vim-Ncm.002	The VIM shall support the capability to provide the notification related to the changes of NFVI capacity information.

8.2.4 Functional requirements for virtualised resource performance management

Table 8.2.4-1: Functional requirements for virtualised resource performance management

Numbering	Functional requirements description
Vim.Vrpm.001	The VIM shall support the capability to collect performance information related to virtualised resources (see note 1).
Vim.Vrpm.002	The VIM shall support the capability to notify regarding the performance information on the virtualised resources that are allocated (see note 1).
Vim.Vrpm.003	The VIM shall support the capability of virtualised resource performance management in response to the request (see note 2).
NOTE 1: Virtualised resource performance information can include the virtualised resource consumption level, such as Central Processing Unit (CPU) utilization, memory usage and bandwidth consumption. NOTE 2: The performance management can include creation, update, query and deletion of PM job or thresholds.	

8.2.5 Functional requirements for virtualised resource fault management

Table 8.2.5-1: Functional requirements for virtualised resource fault management

Numbering	Functional requirements description
Vim.Vrfm.001	The VIM shall support the capability to collect fault information related to virtualised resources (see note 1).
Vim.Vrfm.002	The VIM shall support the capability to notify regarding the fault information on virtualised resources that are allocated (see note 2).
Vim.Vrfm.003	The VIM shall support the capability to notify changes in fault information on virtualised resources (see note 2).
Vim.Vrfm.004	The VIM shall support the capability to perform automated or on-demand corrective operations on virtualised resources failure.
Vim.Vrfm.005	The VIM shall support the capability to provide fault information on virtualised resources that are allocated in response to a query (see note 2).
NOTE 1: The virtualised resources fault can include virtualisation container crashes, virtual network ports errors, virtualisation containers to storage disconnection, etc. NOTE 2: The fault information related to virtualised resources can include the information related to the alarm (e.g. alarm created, alarm cleared, etc.), alarm causes and identification of the virtualised resources causing the alarm, and so on.	

8.2.6 Functional requirements for virtualised resource information management

Table 8.2.6-1: Functional requirements for virtualised resource information management

Numbering	Functional requirements description
Vim.Vrim.001	The VIM shall support the capability of providing information on virtualised resource that can be consumed within its area of responsibility (see note).
Vim.Vrim.002	The VIM shall support the capability to notify the change of information on virtualised resources that can be consumed within its area of responsibility.
NOTE: Virtualised resource Information provided by the VIM can include the description on the characteristic of the virtualised resource that can be consumed, such as virtualised resource configurations (virtual CPU configurations, types of network connectivity (e.g. L2, L3), size of virtual memory, types and size of virtualised storage resource, etc.), and/or templates (e.g. a virtual machine with 2 virtual CPUs and 2 GB of virtual memory), and so on.	

8.2.7 Functional requirements for virtualised resource configuration management

Table 8.2.7-1: Functional requirements for virtualised resource configuration management

Numbering	Functional requirements description
Vim.Vrcm.001	The VIM shall support the capability of configuration management of an individual virtualised resource using specific deployment configuration information received (see note).
Vim.Vrcm.002	The VIM should support the capability of configuration management of a set of related virtualised resources using specific deployment configuration information received (see note).
NOTE: The deployment of specific configuration information can include: Internet Protocol (IP) address types and range, subnet, ports, other guest Operating System (OS) configuration, so on.	

8.2.8 Functional requirements for NFP management

Table 8.2.8-1: Functional requirements for NFP management

Numbering	Functional requirements description
Vim.Nfpm.001	The VIM shall support the capability of management of NFPs, including creating, updating, and deleting an NFP.
Vim.Nfpm.002	The VIM shall support the capability to provide fault notification about the virtualised resources (e.g. CP, virtual network) associated with a specific NFP instance (see note).
Vim.Nfpm.003	The VIM shall validate that the classification and selection rule update does not impact the running classification and selection rules applied to the NFP instance.
NOTE: For example, when a CP instance of an NFP instance is failed, VIM notifies NFVO, and then NFVO disables an NFP or updates the rules applied to the NFP instances.	

8.2.9 Functional requirements for quota management

Table 8.2.9-1: Functional requirements for quota management

Numbering	Functional requirements description
Vim.Qm.001	The VIM shall support the capability to reject virtualised resource allocation requests causing a quota to be exceeded.
Vim.Qm.002	The VIM shall support the capability to create resource quota for the consumer of the virtualised resources (e.g. Tenant).
Vim.Qm.003	The VIM shall support the capability to update the resource quota for the consumer of the virtualised resources (e.g. Tenant) of the virtualised resource.
Vim.Qm.004	The VIM shall support the capability to delete the resource quota for the consumer of the virtualised resources (e.g. Tenant).
Vim.Qm.005	The VIM shall support the capability to provide information on the resource quota for the consumer of the virtualised resources.
Vim.Qm.006	The VIM shall support the capability to notify the changes of the information on the resource quota for the consumer of the virtualised resources.

8.3 Functional requirements for infrastructure resource management

8.3.1 Functional requirements for infrastructure resource performance management

Table 8.3.1-1: Functional requirements for infrastructure resource performance management

Numbering	Functional requirements description
Vim.Irpm.001	The VIM shall support the capability of collection of performance information related to software and hardware resources within the NFVI (see notes 1 and 2).
NOTE 1: Hardware resources within the NFVI refer to physical compute, storage, and networking resources. Software resources refer to software components within the NFVI (e.g. a hypervisor) but do not refer to the VNF's software.	
NOTE 2: Performance information related to software and hardware resource within the NFVI can include software and hardware resource consumption level, such as physical memory consumption, CPU power consumption, Peripheral Component Interface express (PCIe) bandwidth consumption.	

8.3.2 Functional requirements for infrastructure resource fault management

Table 8.3.2-1: Functional requirements for infrastructure resource fault management

Numbering	Functional requirements description
Vim.Irfm.001	The VIM shall support the capability to correlate fault information on virtualised resources with fault information related to underlying used software and hardware resources within the NFVI (see note 1).
Vim.Irfm.002	The VIM shall support the capability of collection of fault information related to software and hardware resources within the NFVI (see note 2).
NOTE 1: Hardware resources within the NFVI refer to physical compute, storage, and networking resources. Software resources refer to software components within the NFVI (e.g. a hypervisor) but do not refer to the VNF's software. NOTE 2: The software and hardware resources fault can include suspension of the underlying OS, physical network disconnection due to a Network Interface Controller (NIC) failure, etc.	

8.4 Functional requirements for security consideration

Table 8.4-1: Functional requirements for security consideration

Numbering	Functional requirements description
Vim.Sc.001	The VIM shall support the capability to validate that the received message is from an authenticated and authorized consumer.
Vim.Sc.002	The VIM shall support the capability to verify the integrity of the received message.
Vim.Sc.003	The VIM shall support the capability to encrypt the sent message or decrypt the received message using negotiated key and algorithm to or from an authenticated and authorized consumer or producer.

8.5 Functional requirements for software image management

NOTE: The software image(s) is/are at virtualisation container level, e.g. Virtual Machine (VM) or OS container images.

Table 8.5-1: Functional requirements for software image management

Numbering	Functional requirements description
Vim.Sim.001	The VIM shall support the capability of management of VM software images as requested.
Vim.Sim.002	The VIM shall support the capability to verify the integrity and authenticity of the VM software images.
Vim.Sim.003	The VIM should support the capability to manage multiple versions of VM software images.
Vim.Sim.004	The VIM shall support the capability to provide the information on the VM software images which it manages.

8.6 Functional requirements for NFV acceleration management

Table 8.6-1: Functional requirements for NFV acceleration management

Numbering	Functional requirements description
Vim.NfvAm.001	The VIM shall support the management of the NFV acceleration resources (see note 1).
Vim.NfvAm.002	The VIM shall support the capability to retrieve feature related information provided by the NFV acceleration resources.
Vim.NfvAm.003	The VIM shall support the capability to provide acceleration capability information to NFVO (see note 2).
Vim.NfvAm.004	The VIM shall support the capability to translate the acceleration capability requirement (e.g. bandwidth value) into acceleration resource context (e.g. number of Field Programmable Gate Array (FPGA) blocks).
NOTE 1: Acceleration resource management in VIM includes discovery, allocation, release, reprogram, etc. of acceleration resources in NFVI. NOTE 2: The information can include type, capacity, NUMA support, etc.	

8.7 Functional requirements for multi-tenancy

Table 8.7-1: Functional requirements for multi-tenancy

Numbering	Functional requirements description
Vim.Mtm.001	The VIM shall support the capability of management of infrastructure tenants (see note 1).
Vim.Mtm.002	The VIM shall support the capability to identify software images assigned to an infrastructure tenant and software images shared among infrastructure tenants.
Vim.Mtm.003	The VIM shall support the capability to allow an infrastructure tenant to instantiate virtual resources using its own private software images or shared software images.
Vim.Mtm.004	The VIM shall support the capability to limit the scope of operations only to the infrastructure resource groups assigned to the requesting infrastructure tenant.
NOTE 1: The management of tenants include: - create, read, update, delete tenants; - associate a tenant to one or more "infrastructure resource groups" managed by a VIM. NOTE 2: A software image which is assigned to a single tenant is commonly referred to as a private software image of this tenant. A software image which is assigned to all tenants is commonly referred to as a public software image. A software image which is assigned to more than one tenant is commonly referred to as a shared software image.	

8.8 Functional requirements for compute host reservation management

Table 8.8-1: Functional requirements for compute host reservation management

Numbering	Functional requirements description
Vim.Chrm.001	The VIM shall support the capability of compute host reservation management (see note).
Vim.Chrm.002	The VIM shall support the capability to notify changes of reserved compute host(s).
NOTE: The management includes the creation, update, query and termination of compute host reservation(s).	

8.9 Functional requirements for policy management

Table 8.9-1: Functional requirements for policy management

Numbering	Functional requirements description
Vim.Plcm.001	The VIM shall support the capability to report the conflicted NFV-MANO policies it detects (see note).
Vim.Plcm.002	The VIM shall support the capability to enforce NFV-MANO policies.
NOTE: The conflicted NFV-MANO policies include policies applied in virtualised resource management (resource allocation, reservation, quota management and capacity management).	

8.10 Functional requirements for virtualised resource Snapshots

Table 8.10-1: Functional requirements for virtualised resource Snapshots

Numbering	Functional requirements description
Vim.VrSnap.001	The VIM shall utilize the Snapshot capabilities supported by the virtualisation technologies (see note).
Vim.VrSnap.002	The VIM shall support the creation of and reversion to a Snapshot of virtualised resources.
NOTE: Snapshot capabilities include configuration of the snapshot operation such as specifying resources to be included, specifying if an instance shall be stopped/halted after snapshot creation, specifying the path for saving snapshots.	

8.11 Functional requirements for management of connectivity for Multi-Site services

Table 8.11-1: Functional requirements for management and connectivity for Multi-Site services

Numbering	Functional requirements description
Vim.Mss.001	The VIM shall support the capability to manage virtualised network resources for connectivity of the NFVI-PoP to/from WAN.
Vim.Mss.002	The VIM shall support the capability to update existing virtualised network resources within the NFVI-PoP to connect to a WAN virtualised network resource enabling connectivity to/from the WAN.
Vim.Mss.003	The VIM shall support the capability to update existing virtualised network resource within the NFVI-PoP to reconnect from a WAN virtualised network resource to another WAN virtualised network resource.
Vim.Mss.004	The VIM shall support the capability to manage virtualised network resources for overlay or inter-AS connections to/from other NFVI-PoPs.
Vim.Mss.005	The VIM shall support the capability to provide information about the connectivity of the NFVI-PoP to/from external networks (e.g. WAN).
Vim.Mss.006	The VIM shall support the capability to provide information about the association of the virtualised network resource within the NFVI-PoP with the internal-to-external NFVI-PoP interconnection.

9 Architectural level Requirements

9.1 General guidelines for NFV management and orchestration interface design

This clause defines general interface guidelines applicable to all NFV-MANO interfaces.

These guidelines are applicable for interface specifications.

Table 9.1-1: General guidelines for NFV management and orchestration interface design

Numbering	Guideline description
Inf.NfvMoidG.001	The interface should be self-contained enabling easy implementation and maintenance (see note).
Inf.NfvMoidG.002	The interfaces should be based on standardized specification, which does not allow room for interpretation.
NOTE: Self-contained implies that the specification should not refer or depend on the specifications of another one.	

9.2 General requirements to NFV management and orchestration interface design

This clause defines general interface requirements applicable to all NFV-MANO interfaces.

NOTE: The requirements for individual interfaces will not be covered in this clause.

These requirements are applicable for interface specifications.

Table 9.2-1: General requirements to NFV management and orchestration interface design

Numbering	Requirements description
Inf.NfvMoid.001	The interface shall provide an extension mechanism.
Inf.NfvMoid.002	The interface extension mechanism should support the addition of private extensions.
Inf.NfvMoid.003	The interface specification shall identify for each information element and attribute whether is mandatory or optional in the context where it is used (see note 4).
Inf.NfvMoid.004	The interface specification shall contain the complete specification of all mandatory information elements necessary for interoperability at the interface.
Inf.NfvMoid.005	Entity names (see note 5) shall be unique across all entity types and all reference points in a given naming domain (see note 1).
Inf.NfvMoid.006	Entity names (see note 5) shall not embed any information beyond the name itself (see note 2).
Inf.NfvMoid.007	An entity (see note 5) shall have the same name across all reference points that it appears.
Inf.NfvMoid.008	A common filtering description shall be used across all NFV interface operations having a filter input parameter (see note 3).
NOTE 1: The extent of a naming domain is a deployment decision which can potentially cover multiple instances of the various NFV reference architecture FBs.	
NOTE 2: For example, it is not recommended to embed location or containment hierarchy in an entity names (such information should be kept in separate attributes).	
NOTE 3: Only a subset of the filtering capability might be needed for a given operation. Typical filtering might be entity list or type matching, template matching or attribute value matching.	
NOTE 4: A context is either a set of input/output information elements for an operation or a set of attributes within a structured information element.	
NOTE 5: Depending on the actual communication solution, an entity may take different forms (e.g. a parameter in a message, a field in a URI, etc.). Consequently, its name may take different forms as well (e.g. a field or parameter tag).	

9.3 General requirements for NFV management and orchestration services

Table 9.3-1: General requirements for NFV management and orchestration services

Numbering	Guideline description
Mano.NfvMos.001	The NFV-MANO shall enable the discovery and retrieval of information regarding management and orchestration related interfaces, including all information necessary for their usage (e.g. interface endpoint address).

9.4 General requirements for multi-tenancy

Table 9.4-1: General requirements for multi-tenancy

Numbering	Functional requirements description
Nfv.Mtm.001	A consumer of an interface which supports multi-tenancy shall provide the identification of an appropriate tenant (infrastructure tenant, VNF tenant or NS tenant) when performing an operation.

10 Functional requirements for NFV-MANO as managed entities

10.1 Functional requirements for management of NFVO as a managed entity

Table 10.1-1: Functional requirements for management of NFVO as a managed entity

Identifier	Functional requirement description
Nfvo.Oam.001	The NFVO as a managed entity shall support NFV-MANO fault management by a managing entity.
Nfvo.Oam.002	The NFVO as a managed entity shall support NFV-MANO performance management by a managing entity.
Nfvo.Oam.003	The NFVO as a managed entity shall support NFV-MANO configuration and information management by a managing entity.
Nfvo.Oam.004	The NFVO as a managed entity shall support NFV-MANO state management by a managing entity.
Nfvo.Oam.005	The NFVO as a managed entity shall support NFV-MANO log management by a managing entity.

10.2 Functional requirements for management of VNFM as a managed entity

Table 10.2-1: Functional requirements for management of VNFM as a managed entity

Identifier	Functional requirement description
Vnfm.Oam.001	The VNFM as a managed entity shall support NFV-MANO fault management by a managing entity.
Vnfm.Oam.002	The VNFM as a managed entity shall support NFV-MANO performance management by a managing entity.
Vnfm.Oam.003	The VNFM as a managed entity shall support NFV-MANO configuration and information management by a managing entity.
Vnfm.Oam.004	The VNFM as a managed entity shall support NFV-MANO state management by a managing entity.
Vnfm.Oam.005	The VNFM as a managed entity shall support NFV-MANO log management by a managing entity.

10.3 Functional requirements for management of VIM as a managed entity

Table 10.3-1: Functional requirements for management of VIM as a managed entity

Identifier	Functional requirement description
Vim.Oam.001	The VIM as a managed entity shall support NFV-MANO fault management by a managing entity.
Vim.Oam.002	The VIM as a managed entity shall support NFV-MANO performance management by a managing entity.
Vim.Oam.003	The VIM as a managed entity shall support NFV-MANO configuration and information management by a managing entity.
Vim.Oam.004	The VIM as a managed entity shall support NFV-MANO state management by a managing entity.
Vim.Oam.005	The VIM as a managed entity shall support NFV-MANO log management by a managing entity.

10.4 Functional requirements for management of WIM as a managed entity

Table 10.4-1: Functional requirements for management of WIM as a managed entity

Identifier	Functional requirement description
Wim.Oam.001	The WIM as a managed entity shall support NFV-MANO fault management by a managing entity.
Wim.Oam.002	The WIM as a managed entity shall support NFV-MANO performance management by a managing entity.
Wim.Oam.003	The WIM as a managed entity shall support NFV-MANO configuration and information management by a managing entity.
Wim.Oam.004	The WIM as a managed entity shall support NFV-MANO state management by a managing entity.
Wim.Oam.005	The WIM as a managed entity shall support NFV-MANO log management by a managing entity.

11 Functional requirements for WIM

11.1 General considerations

The following statement on the scope of WIM applies to all WIM related requirements:

- The WIM provides and manages connectivity between NFVI-PoPs in support for multi-site services.

11.2 Functional requirements related to virtualised resource management

11.2.1 Functional requirements for virtualised resource management

Table 11.2.1-1: Functional requirements for virtualised resource management

Numbering	Functional requirements description
Wim.Vrm.001	The WIM shall support management of virtualised network resources for connectivity amongst NFVI-PoP over WAN infrastructure (see note).
NOTE: The resource management includes allocation, termination, update, etc. of virtualised resources in the WAN infrastructure.	

11.2.2 Functional requirements for resource reservation management

Table 11.2.2-1: Functional requirements for resource reservation management

Numbering	Functional requirements description
Wim.Rrm.001	The WIM shall support the capability to manage resources according to different resource commitment models (see clause 5.1 for the basic concept of the resource model), as follows: • Reservation model (see note 1); • Quota model (see note 2); • On demand (see note 3).
NOTE 1: The reservation model is used when virtualised network resources in the WAN are added in advance of their usage (e.g. redundancy, healing). NOTE 2: The quota model is used when a particular consumer is limited to a defined amount or a percentage of resources. NOTE 3: The resources are committed when a connectivity amongst NFVI-PoPs is instantiated or scaled out, as long as there are available resources for consumption.	

11.2.3 Functional requirements for virtualised resource fault management

Table 11.2.3-1: Functional requirements for virtualised resource fault management

Numbering	Functional requirements description
Wim.Vrfm.001	The WIM shall support the capability to report alarms about faulty virtualised network resources in the WAN.

11.2.4 Functional requirements for virtualised resource information management

Table 11.2.4-1: Functional requirements for virtualised resource information management

Numbering	Functional requirements description
Wim.Vrim.001	The WIM shall support the capability to provide information about the virtual network resources of the WAN (see note 1).
Wim.Vrim.002	The WIM shall support the capability to provide information about the connectivity support on the WAN.
Wim.Vrim.003	The WIM shall have the capability of QoS information management for virtualised network resources of the WAN (see note 2).
NOTE 1: The virtual network resources includes e.g. topology, bandwidth, etc. NOTE 2: The QoS information includes e.g. bitrate, latency, delay, jitter, etc.	

12 Functional requirements for CISM function

12.1 General considerations

The following statement on the scope of the CISM function, part of NFV-MANO, applies to all CISM related requirements:

- The CISM is responsible for the deployment, monitoring, and lifecycle management of containerized workloads as MCIOs running in OS containers.
- The CISM exposes corresponding APIs to its consumers and translates incoming requests into operations which are enforced towards the CIS.
- The CISM is further responsible to parse and interpret declarative descriptors and configuration files of the containerized workloads.

The CISM function is exposing OS container management service interfaces on different abstraction levels:

- The "OS container workload management service" exposes a management service interface on MCIOP abstraction level.
- The "OS container compute/storage/network management services" expose management service interfaces on MCIO abstraction level.

More detailed information about the CISM services and the OS container NFV object model is provided in ETSI GS NFV-IFA 040 [i.22].

The functional requirements on the CISM function are grouped into requirements per CISM management service.

12.2 Functional requirements for OS container infrastructure resource management

Table 12.2-1: Functional requirements for OS container infrastructure resource management

Numbering	Functional requirements description
Cism.Irm.001	The CISM function shall support the capability to request OS container infrastructure resource management from the CIS that meet the specified compute/storage/network resource requests of the MCIOs. (see note)
Cism.Irm.002	The CISM function shall support the capability of namespace quota based OS container infrastructure resource management. (see note)
Cism.Irm.003	The CISM function shall support the capability to enforce affinity and anti-affinity policies for OS container infrastructure resource management. (see note)
NOTE: OS container infrastructure resource management includes allocation, termination, update etc. of OS container infrastructure resources.	

12.3 Functional requirements for MCIO management

Table 12.3-1: Functional requirements for MCIO management

Numbering	Functional requirements description
Cism.Mciom.001	The CISM function shall support the capability to validate the lifecycle operation requests for MCIOs it processes, using information specified in declarative descriptors.
Cism.Mciom.002	The CISM function shall support the capability to manage the creation of MCIOs whose declarative descriptors specify compute/storage/network resource requests.
Cism.Mciom.003	The CISM function shall support the capability to manage the modification of the desired state of MCIOs whose declarative descriptors specify compute/storage/network resource requests.
Cism.Mciom.004	The CISM function shall support the capability to manage the modification of the actual state of MCIOs whose declarative descriptors specify compute/storage/network resource requests.
Cism.Mciom.005	The CISM function shall support the capability to manage the replacement of MCIOs whose declarative descriptors specify compute/storage/network resource requests.
Cism.Mciom.006	The CISM function shall support the capability to manage the deletion of MCIOs whose declarative descriptors specify compute/storage/network resource requests.
Cism.Mciom.007	The CISM function shall support the capability to provide information about the actual and desired state of MCIOs whose declarative descriptors specify compute/storage/network resource requests.
Cism.Mciom.008	The CISM function shall support the capability to list MCIOs whose declarative descriptors specify compute/storage/network resource requests.
Cism.Mciom.009	The CISM function shall support the capability to notify on events of changes of the desired or actual state of MCIOs whose declarative descriptors specify compute/storage/network resource requests.

12.4 Functional requirements for management of containerized workloads based on MCIOPs

Table 12.4-1: Functional requirements for management of containerized workloads based on MCIOPs

Numbering	Functional requirements description
Cism.Cwm.001	The CISM function shall support the capability to validate the lifecycle operation requests for containerized workloads it processes, using information specified in the MCIOP.
Cism.Cwm.002	The CISM function shall support the capability to manage the instantiation of containerized workloads based on a MCIOP.
Cism.Cwm.003	The CISM function shall support the capability to provide information about containerized workloads based on a MCIOP.
Cism.Cwm.004	The CISM function shall support the capability to manage the modification of containerized workloads based on a MCIOP.
Cism.Cwm.005	The CISM function shall support the capability to manage the termination of containerized workloads based on a MCIOP.
Cism.Cwm.006	The CISM function shall support the capability to extract and modify declarative descriptors of MCIOs from MCIOPs.
Cism.Cwm.007	The CISM function shall support the capability to request MCIO management operations from the CISM service interfaces for compute, storage and network management.
Cism.Cwm.008	The CISM function shall support the capability to receive run-time data related to containerized workloads that it has created.
Cism.Cwm.009	The CISM function shall support the capability to provide information on the mapping relationship between the containerized workloads based on a MCIOP and associated MCIOs in response to the query.

12.5 Functional requirements for OS container configuration management

Table 12.5-1: Functional requirements for OS container configuration management

Numbering	Functional requirements description
Cism.Cm.001	The CISM function shall support the capability to manage the creation of MCIO configurations.
Cism.Cm.002	The CISM function shall support the capability to manage the modification of MCIO configurations.
Cism.Cm.003	The CISM function shall support the capability to manage the replacement of MCIO configurations.
Cism.Cm.004	The CISM function shall support the capability to manage the deletion of MCIO configurations.
Cism.Cm.005	The CISM function shall support the capability to provide information about MCIO configurations.
Cism.Cm.006	The CISM function shall support the capability to list MCIO configurations.
Cism.Cm.007	The CISM function shall support the capability to notify on events of MCIO configuration changes.
Cism.Cm.008	The CISM function shall support the capability to manage the creation of policies for MCIOs.
Cism.Cm.009	The CISM function shall support the capability to manage the modification of policies for MCIOs.
Cism.Cm.010	The CISM function shall support the capability to manage the replacement of policies for MCIOs.
Cism.Cm.011	The CISM function shall support the capability to manage the deletion of policies for MCIOs.
Cism.Cm.012	The CISM function shall support the capability to provide information about MCIO policies.
Cism.Cm.013	The CISM function shall support the capability to list MCIO policies.
Cism.Cm.014	The CISM function shall support the capability to notify on events of MCIO policy changes.
Cism.Cm.015	The CISM function shall support the capability to manage the creation of namespace quota.
Cism.Cm.016	The CISM function shall support the capability to manage the modification of namespace quota.
Cism.Cm.017	The CISM function shall support the capability to manage the deletion of namespace quota.
Cism.Cm.018	The CISM function shall support the capability to provide information about namespace quota.
Cism.Cm.019	The CISM function shall support the capability to manage the creation of namespaces.
Cism.Cm.020	The CISM function shall support the capability to manage the deletion of namespaces.
Cism.Cm.021	The CISM function shall support the capability to provide information about namespaces.

12.6 Functional requirements for OS container image management

Table 12.6-1: Functional requirements for OS container image management

Numbering	Functional requirements description
Cism.Cim.001	The CISM function shall support the capability to query the CIR function for information of OS container images.

13 Functional requirements for CIR function

13.1 General considerations

The following statement on the scope of the CIR function applies to all CIR related requirements:

- The CIR is responsible for storing OS Container images and maintaining information of OS Container images.
- The CIR is able to manage OS Container images according to the different roles.
- An OS Container image contains all components required to deploy an OS Container on a CIS instance.

More detailed information about the CIR services and the OS container NFV object model is provided in ETSI GS NFV-IFA 040 [i.22].

13.2 Functional requirements for OS container image management

Table 13.2-1: Functional requirements for OS container image management

Numbering	Functional requirements description
Cir.Cim.001	The CIR function shall support the capability of management of OS container images as requested. (see note)
Cir.Cim.002	The CIR function shall support the capability to verify the integrity of the OS container images.
Cir.Cim.003	The CIR function should support the capability to manage multiple versions of OS container images.
Cir.Cim.004	The CIR function shall support the capability to provide the information on the OS container images which it manages.
Cir.Cim.005	The CIR function shall support the capability to manage OS container images according to the different roles.
NOTE: OS container image management includes adding, deleting etc. of OS container images.	

Annex A (informative): Resource management additional information

A.1 Quota based resource management

A.1.1 Overview

To ensure appropriate allocation of NFVI resources, resource quotas can be used in the VIM. These quotas can be used to constrain the NFVI resources which a consumer of these resources can obtain. A consumer identifier will be included in all resource requests to the VIM where quota based resource management is supported. The entities which the consumer identifier maps to are up to service provider configuration. A request for resources beyond a quota limit will be rejected by the VIM.

To ensure that the NFVO has visibility of actual resource utilization in the NFVI, resource consumption and availability information can be exchanged between the VIM and NFVO via processes of event notification, periodic update and query.

A.1.2 Summary of key aspects

Key aspects of the quota based resource management approach are:

- A consumer quota is associated with a consumer identifier.
- Service providers determine the appropriate level of resource quotas associated with consumer identifiers, and the mapping of consumer identifiers to entities.
- A consumer quota for NFVI resources is set in the VIM via interaction with the NFVO or via an alternative configuration mechanism.
- The VNFM may be informed of the resource quotas at the VIM which is imposed on it or the VNFs which it manages.
- The VNFM takes direction from the NFVO before taking any action relating to the instantiation and scaling of VNFs.
- A VIM that supports quota based resource management will validate that requests for resources are within the quota of the consumer identifier provided in the request prior to allocation.
- If a quota associated with a consumer identifier is exceeded the VIM will reject the request.

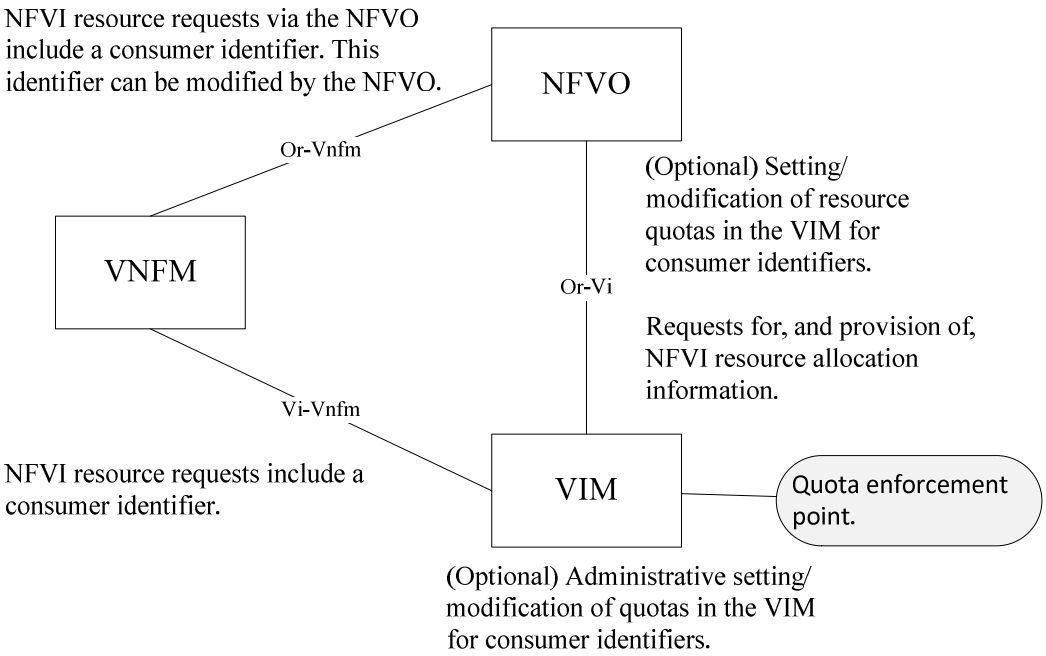

Figure A.1.2-1: Architectural outline of resource quotas

A.1.3 Assignment of consumer identifiers

Consumer identifiers will be assigned via local configuration or via instruction from the NFVO. The entities which the consumer identifier is associated with are determined by the service provider.

A.1.4 Setting of quotas

To avoid unexpected or inappropriate use of NFVI resources, defined quotas (limits) for consumers can be set in the VIM regarding the type and quantity of resources which can be requested from the VIM. The quota information which associates consumer identifiers with specific quotas is communicated to the VIM over the Or-Vi reference point or by some other configuration process. Quota can be modified after being set.

A.1.5 NFVO awareness of NFVI resource consumption

To enable the NFVO to intelligently manage resources, the NFVO can obtain information from the VIM regarding NFVI resource allocations and outstanding resource reservations. It can do this via notification of NFVI resource consumption change events, resource information change notifications from the VIM or a periodic resource information query to the VIM.

A.1.6 NFVI resource acquisition

A VNFM with granted permission for the instantiation or scaling of a VNF can send a resource request to the VIM containing a consumer identifier. If the resources are available in the NFVI, and the quota associated with consumer identifier is not exceeded, then the requested resources will be allocated. If allocation of the requested resources would breach the quota for the consumer identifier, then the request will be rejected. Additionally, a notification can be sent to the NFVO informing it of the action taken by the VIM.

The NFVO can use the notification of this event to determine a subsequent action to: free up NFVI resources; seek access to alternative NFVI resources; or take whatever action was felt to be appropriate.

A.1.7 Resource contention mitigation

The NFVO is expected to have the ability to monitor resource allocation in the NFVI via the VIM. Hence it is anticipated that any decision it takes which would require consumption of additional NFVI resources would take into account its understanding of resource availability in the NFVI. If the NFVO was aware of resource limitations in the NFVI, and hence that there was a probability of insufficient resources to complete a VNF lifecycle management task, then the NFVO might not grant this task and take alternative action instead.

A.1.8 Data centre resource utilization efficiency

Resource management without reservation maximizes the availability of NFVI resources by ensuring that resources are only removed from the pool of available resources when in active use.

A.1.9 Resource management evolution and interoperability

The resource quota enforcement approach could be commercially deployed in phases. For example, an initial deployment can involve very simple consumer resource limitations quotas administratively configured in the VIM. The deployed solution could then be enhanced over time as each entity became more capable. Further enhancement could be provided via a mechanism to enable reservation of NFVI resources from the NFVO. This capability might be used to assure resource availability for critical VNFs or where it was felt necessary in a data centre environment shared by different commercial entities.

A.1.10 Co-existence of resource quota enforcement and resource management with reservation

It is anticipated that the reservation of NFVI resources from the NFVO to the VIM would render the requested resources unavailable until they were released. Hence a resource request without a reservation and using the quota based resource management would have resources allocated to it from a pool of free resources not under active reservation. Additionally, local rules will determine the behaviour in the VIM if a reservation is received which is in excess of an applicable consumer quota.

A.2 Management of resource reservations

A.2.1 Introduction

Reservation enables securing resources to guarantee their availability without allocating them, i.e. resources are committed to a particular consumer or consumer type, but not necessarily all of them are allocated/instantiated yet.

Various use cases for reservation are introduced and the key aspects of reservation presented.

A.2.2 Use cases

A.2.2.1 Use case for securing resources for several tenants

The NFV-MANO framework enables tenants to request and make use of virtualised resources provided by the platform. VIM manages the NFVI and offers to consumers (tenants) operations for managing virtualised resources. In NFV deployments, several tenants can coexist, and in this scenario resource management race conditions can happen, ending in resource service denegation. In carrier telco environments, with stringent SLAs, reliability and performance requirements, resource service denegation can become an issue.

The NFVO plays a key role in the NFV-MANO, as central point for orchestrating the resource consumption by VNFs and NSs and granting the lifecycle operations. The NFVO cannot guarantee resource availability during the granting of a VNF lifecycle request if the resources needed to accommodate such lifecycle operation have not been secured (i.e. reserved) by the VIM, entity responsible for the NFVI resources management.

A.2.2.2 Use case for securing resources with detailed capabilities

The VIM, as end point for managing and controlling the NFVI resource holds more detailed information about the managed resources and their availability. At the NFVO, visibility of specific resources is not the same as the VIM. The NFVO holds information about the availability, reserved and allocated NFVI resources as abstracted by the VIM.

Examples of more detailed information are specific acceleration capabilities, CPU-pinning, etc. This information is visible at the VIM level in order to execute the right allocation of virtualised resources according to the resource capability requirements. If such capabilities are needed, and the NFVO has no visibility on the particular resources accommodating such capabilities, granting the VNF lifecycle operations can lead to undesired resource service denegation, in particular those that follow with subsequent virtualised resource management requests for detailed capabilities.

A.2.2.3 Use case for securing resources during NS instantiation

An NS can be composed of a number of VNFs, VLs to interconnect them, etc. In order to realize an NS, it is possible that a great quantity of NFVI resources will be needed. Thus, the instantiation of an NS will be possible as long as all the resources can be secured to be available at the time of the instantiation of the NS.

The instantiation of an NS can involve several transactions, with possibly a number of different VIMs managing the required NFVI resources, and VNFMs managing the lifecycle of the VNFs to instantiate. During the instantiation process, if resources cannot be secured to be available by the VIM(s) for the NS, the overall instantiation can fail. This can lead to inefficient processing and arrangement of NS instantiation.

A.2.2.4 Use case for securing resources during NS scaling

An NS can be composed of a number of VNFs, VLs to interconnect them, etc. In order to realize an NS, as well as for scaling purposes, it is possible that a great quantity of NFVI resources will be needed. Moreover, under certain scenarios, such as sport events or natural disasters, operators require that NSs can scale to accommodate the extra traffic to handle. Such NS scaling requires adding extra resources to be used by the VNFs part of the NS, or new ones to be instantiated. By reserving resources in advance against the VIM managing the resources, it is ensured that NS can scale properly.

A.2.2.5 Use case for securing resources related to a scheduled event

NSs or certain capacity may only be needed for a specified duration. For instance, the duration of a scheduled sport event is usually known in advance, i.e. with an expectation to be ended at some point in time.

To support the event, the operator may need to add extra NS capacity or instantiate a new NS. In this scenario, the service provider wishes to secure the instantiation of new VNF instances, or the expansion of existing instances for the NS by reserving underlying NFVI resources.

The present use case exemplifies the need for the NFVO and VIM to handle reservation time information.

As part of the NS instantiation/expansion, the NFVO requests to the appropriate VIM(s) the reservation of virtualised resources needed by the VNF instances. In addition, the NFVO provides information about the expected timespan where the virtualised resources will be used, i.e. it provides start and end time information. The time information may either be the same or have certain deviation from the scheduled event timing to allow for certain backup time. This information about start and end time helps the VIM to determine the best scheduling of resources and their availability in the NFVI-PoP(s). This is particularly applicable when scheduling resources for multiple future events, i.e. the VIM will know about reservations that have been scheduled but whose reserved resources are not being used yet or reservations that have been scheduled, but whose reserved resources will be freed prior to another reservation.

A.2.3 Summary of key aspects

Key aspects of the resource reservation are:

- NFVO decides if and when a resource reservation is needed.

- Resource reservation can be done:
  - before a VNF LCM operation as part of an NS LCM operation;
  - as part of granting procedure for a VNF LCM operation; and
  - during configuration/reconfiguration of resources in the NFVI-PoP(s).
- NFVO requests the reservation of the needed resources to the VIM.
- Reservations are identifiable. A reservation identifier establishes the identity of the arrangement for securing the future usage of resources by a consumer.
- When resource reservation is performed as indicated by policies, the reservation identifier is directly used by NFVO as part of managing the resource reservation. The identifier is provided to the VNFM, either as part of a VNF LCM operation request or in response to a granting request:
  - VNFM uses the reservation identifiers for requests related to the resources that are needed for the instantiation and lifecycle of VNF.

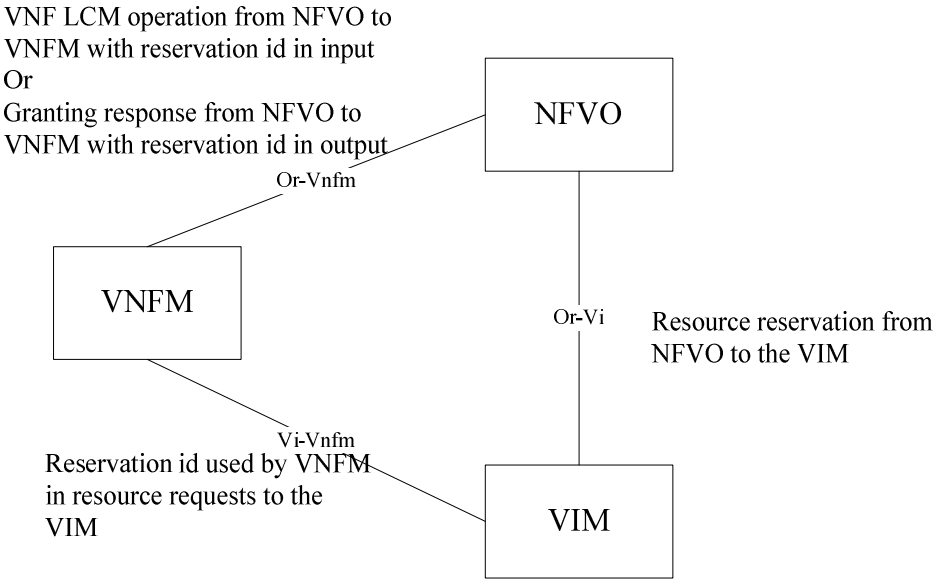

Figure A.2.3-1: Architectural outline of reservation

A.2.4 Resource reservation management by NFVO

Resource reservations are triggered by NFVO by calling the corresponding VIM to reserve the resources. It is anticipated that the reservation of NFVI resources from the NFVO to the VIM would render the requested resources unavailable until they were released.

The NFVO, based on the feasibility check and reservation option in the input LCM operation or based on operator policies, reserves virtualised resources and/or physical compute hosts.

In case of NS LCM operation where reservation is needed, NFVO will reserve the resources needed for each VNF LCM operation for all the impacted VNFs in the NS. Once the reservations are successfully secured, the NFVO will issue corresponding reservation identifier(s) to the VNFM.

In case of failure of one of the LCM operations, the NFVO will cancel any pending reservations associated with the LCM request.

In case of VNF LCM operation, not coming from an NS LCM operation, if reservation is needed, the NFVO will reserve the needed resources as part of the granting request. The corresponding reservation identifier(s) will be returned as part of the grant response.

A.2.5 Resource reservation handling by the VNFM

A VNFM can receive, either in part of the input parameters of a VNF LCM operation or in the response of a grant request, one or more than one reservation identifier.

A reservation identifier indicates that a reservation has been performed for this VNF. The VNFM makes use of this reservation identifier(s) in the subsequent resource requests for this VNF made to the VIM.

A.2.6 Resource reservation contention mitigation

The VIM handles the resource reservation contention mitigation as the VIM is responsible for the control of whether virtualised resources can be reserved or not based on the detailed internal capacity information that it maintains.

The VIM is expected to have the ability to monitor the availability of resources in the NFVI and how virtualised resources can be accommodated in the NFVI. To mitigate reservation contention, it is also expected the VIM will ensure that NFVI resources are reserved efficiently. For instance, performing by the VIM a uniform reservation in the physical NFVI resources may lead to a situation where certain virtualised resources demanding large amount of resources cannot be allocated when needed.

EXAMPLE: Consider 2 physical NFVI resource nodes (Node-1 and Node-2) with 4 capacity units that can be reserved. A first reservation requests for 2 affine capacity units (i.e. on the same node) is processed by the VIM, and these 2 capacity units are reserved from Node-1. A second reservation request for 2 affine capacity units is also processed by the VIM, and using a uniform reservation policy these 2 capacity units are reserved from Node-2. A third reservation request for 3 affine capacity units cannot be successfully processed as there are not enough free capacity units neither from Node-1 nor from Node-2.

It is also possible for the NFVO to perform actions to mitigate resource reservation contention by monitoring the capacity usage of resources from the NFVI-PoP(s), as reported by the corresponding VIM(s). For instance, requesting resource reservation on a highly loaded NFVI-PoP can increase the chances of rejection of the resource reservation.

A.2.7 Co-existence of reservation with quota

The quota mechanism is used to constrain the NFVI resources that a consumer of these resources can obtain. If applicable, the VIM will also apply the quota to the reservation being made. Local rules will determine the behaviour in the VIM if a reservation is received which is in excess of an applicable consumer quota.

A.2.8 Resource reservation types

Resource reservation can be performed at different levels, namely:

1) for virtualised containers, virtual networks, network ports and/or storage volumes; or/and

2) for virtualised resource capacity (on compute, storage, and network resource types); or/and

3) for physical compute hosts.

The first case considers the reservation of virtualised containers (e.g. VMs) based on defined container configurations, e.g. it supports the reservation based on certain virtualisation container flavours that determine the number and disposition of vCPUs, virtual memory, virtual storage and number of virtual network interfaces. Reservation for defined virtual networks, network ports and storage volumes is also part of this category.

The second case considers the reservation of resource capacity without a specific virtualised container disposition. For example, a resource reservation in this format may indicate the total required capacity in terms of number of vCPUs and virtual memory. Reservation of total capacity for virtual storage, or number of public IP addresses is also part of this category.

The third case considers the reservation of physical compute hosts based on defined capabilities associated to the physical compute hosts (e.g. hypervisor capabilities ETSI GS NFV-PER 001 [i.8]).

A.3 Management of permitted allowance

A.3.1 Introduction

To ensure consumption of resources stays within the limits defined by service providers, permitted allowance can be used at NFVO level to control resource consumption by VNFMs in relation to some granularity associated with the permitted allowance. The granularity might vary (VNFM, VNF, group of VNFs, NS, etc.). Permitted allowance is maintained by the NFVO.

All VNF LCM request that imply potential resource changes, i.e. instantiation, scaling in/out, update, terminate, upgrade and healing of VNF instances are using the grant operation and as part of the processing of the grant operation, the permitted allowance is checked and the current level maintained.

A.3.2 Summary of key aspects

Key aspects of the management of permitted allowance are:

- Service providers determine the appropriate level of resource for the permitted allowance and the corresponding granularity.
- Permitted allowances are provided to the NFVO.
- NFVO supports the permitted allowance by checking the matching one during the processing of the grant request.
- NFVO maintains the current level of the permitted allowance based on the granted requests.

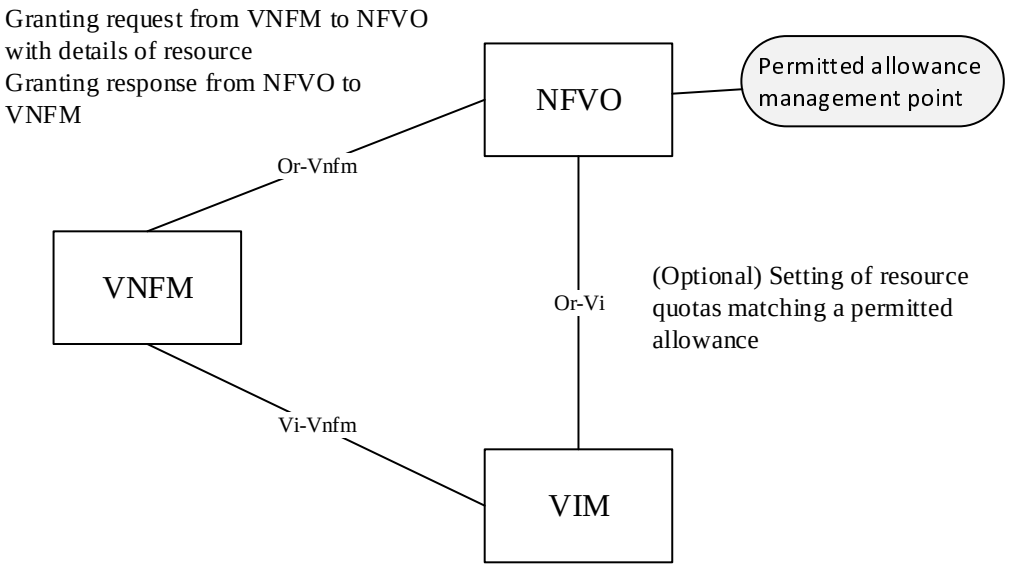

Figure A.3.2-1: Architectural outline of permitted allowance

A.3.3 Setting of permitted allowance

To ensure consumption of resources stays within the limits defined by service providers, the operator or the OSS can define permitted allowance regarding the type and quantity of resource associated with a given granularity. This permitted allowance might be applicable across multiple VIMs.

This permitted allowance information can be communicated to the NFVO over the Os-Ma-nfvo reference point or configured by some other process.

A.3.4 Permitted allowance management by NFVO

The permitted allowance is managed by NFVO, as a maximum and current level of resources. The maximum level corresponds to the definition of the permitted allowance and the current level is what is being marked as consumed as a result of the grant requests.

When receiving a grant request from a VNFM, as part of the processing of the grant, the NFVO matches the request to the permitted allowance with corresponding granularity.

If the request is asking for resources, i.e. instantiate, scale out, etc., NFVO checks if adding the desired resources provided as part of the grant request to the current level of resources still maintains the current level below the maximum level. If so, the request stays within the permitted allowance.

If the request is freeing resources, i.e. terminate, scale in, the NFVO subtracts the provided resources from the current level, making it lower.

In case the VNF LCM operation fails at VNFM, resources might be marked as used (not used) in the permitted allowance while not used (used) in reality. NFVO would need to check that the resources are affectively used (not used), for instance by checking for correct lifecycle instantiation/scale/termination events of a VNF to avoid this problem.

A.3.5 Permitted allowance awareness by the VNFM

A VNFM when processing a VNF LCM request that imply potential resource changes, i.e. instantiation, scaling in/out, update, terminate, upgrade and healing of VNF instances issues a grant request to the NFVO with the details of the operation, the VNF and the resource change (resource needed or resource released).

One of the actions of the processing of the grant request is to validate the request against matching permitted allowance. The VNFM is not aware of the details of the permitted allowance used by the NFVO for the grant operation.

If the response from the grant is successful, the VNFM can issue resource requests.

A.3.6 Permitted allowance contention mitigation

The NFVO is managing permitted allowance and when a permitted allowance reaches its limit, NFVO should issue a notification and should reject granting requests asking for more resources and matched to this permitted allowance.

The OSS or the operator are expected to have the ability to monitor these notifications and might react by extending the permitted allowance that reached its limit.

A.3.7 Co-existence of permitted allowance and resource quota enforcement

If the definition of a permitted allowance is compatible with the definition of quota, i.e. applicable to a single VIM, using the resource granularity supported by quotas, the NFVO might choose to enforce a permitted allowance by defining in the VIM a quota that correspond to a given allowance using a specific tenant.

In this case, the tenant associated with the quota would be communicated to the VNFM in the grant response and the VNFM will use it for all resource allocation requests associated to the granted VNF LCM request.

A.3.8 Co-existence of permitted allowance and resource management with reservation

The permitted allowance is managed at NFVO level while the reservation is made at VIM level. So, they can both co-exist without impact.

As well as actual resource consumption, resources reserved can count towards permitted allowance. The handling of permitted allowance for reserved resources is similar to normal resources as described in clause A.3.4.

Annex B (informative): Virtualised resources capacity management

B.1 Introduction

Virtualised resources capacity management encompasses functionalities to gather information about virtualised resource capacity usage. Both the VIM and NFVO perform functionality related to virtualised resources capacity.

B.2 Virtualised resources capacity information management by the VIM

B.2.1 Functionality

The VIM executes the following functionality as baseline to support virtualised resources capacity information management:

- It manages inventory related information of NFVI hardware resources (compute, storage, network) and software resources (e.g. hypervisors), including the discovery of capabilities of such resources.
- It keeps information about reservation and usage of virtualised resources identifying the association of the virtualised resources to the physical compute, storage and network resources.

NOTE: The particular allocation, update, migration, scaling, operation and termination of virtualised resources are virtualised resource management functions.

The VIM executes the following functionality to actually perform virtualised resources capacity information management:

- It manages information about virtualised resources capacity per NFVI-PoP and resource zone, detailing total, available, allocated and reserved virtualised resource capacity per resource type.
- It provides information about virtualised resources capacity and notifies changes about the virtualised resources capacity.
- It provides information about NFVI-PoP(s) it administers, such as network connectivity endpoints and geographical location.

B.3 Virtualised resources capacity management by the NFVO

B.3.1 Functionality

The NFVO performs the following functionality related to virtualised resources capacity information management:

- It retrieves and processes notifications from VIM instances with information about NFVI-PoP virtualised resources capacity usage at different granularities and levels as provided by the VIM, including total per NFVI-PoP and per resource zone.
- It retrieves information from VNFM instances about virtualised resources usage and mapping with instantiated VNFs.

- It retrieves information about the connectivity to and in-between NFVI-PoPs and Network Point of Presences (N-PoPs) and builds network topology map information.
- It keeps information about retrieved virtualised resources capacity and synchronizes such information on-demand or periodically with VIMs, WAN Infrastructure Managers (WIMs) in order to keep the information updated.
- It keeps information about retrieved VNF's resource usage and synchronizes such information on-demand or periodically with VNFMs in order to keep the information updated.
- It aggregates the capacity information received from VIMs and WIM, and correlates such information with VNF's resource usage from VNFMs to quantify and determine the virtualised resource capacity usage mapped to VNF and NS instances throughout time.

The NFVO makes use of the virtualised resources capacity information to:

- Support analytics for virtualised capacity planning to determine best usage of NFVI resources across NFVI-PoPs.
- Generate virtualised resources capacity reports and notify about resource shortage.
- Validate NS resource usage and distribution of resource usage across operator's Infrastructure Domains.
- Validate and grant VNF lifecycle operations requested from VNFM, as those may impact the way requested resources are allocated within one NFVI-PoP or across multiple NFVI-PoPs.
- Placement optimization for the instantiation and LCM of VNFs and NSs, including:
  - Identifying and selecting the target VIM and WIM to which virtualised resources will be reserved and/or consumed for VNFs and NS.
  - Selecting the target resource zones in NFVI-PoPs to accommodate VNF instantiation according to input resource, performance and resiliency requirements.

Annex C (informative): VNF management

C.1 Introduction

This annex reports on concepts related to VNF management.

Clause C.2 introduces use cases related to VNF management.

C.2 Use cases

C.2.1 Use case for stopping a VNF instance

C.2.1.1 Introduction

The goal of the use case is to enable stopping a running VNF instance without releasing the virtualised resources that have been instantiated to such VNF instance. As part of this process, the guest Operating System (OS) of the VNF instance may be shutdown. The VNFM is responsible for executing the procedure.

Stopping a VNF instance allows fast re-activation of a VNF without having to re-instantiate the virtualised resources. Together with starting a VNF instance, it provides a means to reboot a VNF instance, e.g. to be used to reactivate a VNF whose application was faulty and there were no other means to recover from the fault.

Both EM and NFVO need to be able to request stopping a VNF instance. For instance, the EM as manager of the application from OSS/BSS perspective is involved in the procedures related to commissioning and decommissioning of the VNF into service and failure correction. The NFVO needs to also be able to trigger the operation, e.g. as part of NS lifecycle and fault management procedures.

C.2.1.2 Steps

Actors:

- NFV-MANO (VIM, NFVO and VNFM).
- VNF instance.

Pre-Conditions:

- The VNF instance is instantiated and running.
- NFV-MANO (VIM, NFVO and VNFM) is running.

Steps:

1) The VNFM receives a request from the NFVO or the EM to stop the VNF instance.
2) The VNFM sends VNF lifecycle change notification to consumers (NFVO and/or EM) about the start of the stopping procedure.
3) The VNFM knows the shutdown order between VNFC instances of the VNF (e.g. in accordance with workflow(s) in VNFD) and sends command to VIM to shut down the associated virtualised containers (e.g. VMs).

NOTE: If the workflow requires a graceful stop, as part of this process the VNFM will interact with VNF/EM to gracefully stop the application.

4) VIM processes the request and signals to the hypervisor in the NFVI to shut down the virtualised container(s).

5) VIM returns confirmation of shutting down the virtualised container(s) to the VNFM.

6) VNFM sends notification with the result of the operation to consumers (NFVO and/or EM).

Post-Conditions:

- The VNF instance is stopped.

C.2.2 Use case for starting a VNF instance

C.2.2.1 Introduction

The goal of the use case is to enable starting a VNF instance that was previously in the state "stopped" without having to modify the virtualised resources that were previously instantiated. As part of this process, the guest OS of the VNF instance may be booted if it has been shut down. The VNFM is responsible for executing the procedure.

Starting a VNF instance allows fast re-activation of a VNF without having to re-instantiate the virtualised resources. Together with stopping a VNF instance, it provides a means to reboot a VNF instance, e.g. to be used to reactivate a VNF whose application was faulty and there were no other means to recover from the fault.

Both EM and NFVO need to be able to request starting a VNF instance. For instance, the EM as manager of the application from OSS/BSS perspective is involved in the procedures related to commissioning and decommissioning of the VNF into service and failure correction. The NFVO needs to also be able to trigger the operation, e.g. as part of NS lifecycle and fault management procedures.

C.2.2.2 Steps

Actors:

- NFV-MANO (VIM, NFVO and VNFM).
- VNF instance.

Pre-Conditions:

- The VNF instance is instantiated and stopped.
- NFV-MANO (VIM, NFVO and VNFM) is running.

Steps:

1) The VNFM receives a request from the NFVO or EM to start the VNF instance.

2) The VNFM sends VNF lifecycle change notification to consumers about the start of the starting procedure.

3) The VNFM knows the boot-up order between VNFC instances of the VNF (e.g. in accordance with workflow(s) in VNFD) and sends command to VIM to boot up the associated virtualised containers (e.g. VMs).

4) VIM processes the request and signals to the hypervisor in the NFVI to boot up the virtualised container(s).

5) VIM returns confirmation of booting the virtualised container(s) to the VNFM.

6) VNFM sends notification with the result of the operation to consumers (NFVO and/or EM).

Post-Conditions:

- The VNF instance is started.

Annex D (informative): Network service management additional information

D.1 Introduction

Network service management is the main functionality exposed on the external reference point Os-Ma-nfvo, which is illustrated in Figure D.1-1.

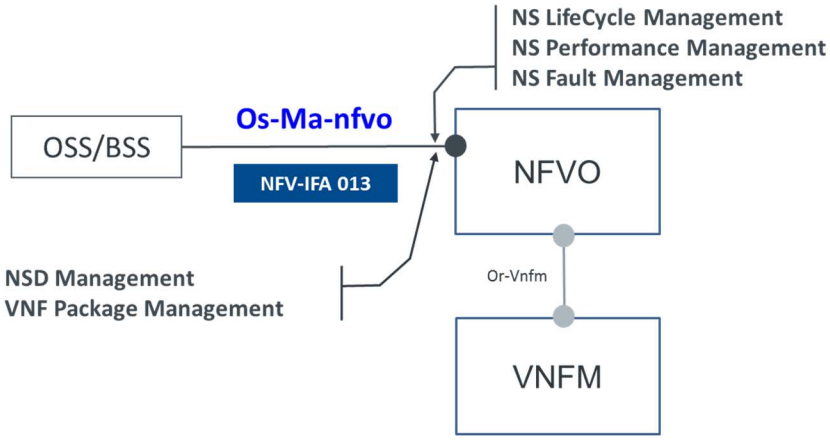

Figure D.1-1: Consuming network service management

The following clauses provide general use cases for network service management.

The network slice management function is one of the sub-functions in the OSS. The network slice management is achieved via Network Service management. The use cases focus on this consumer (or tenant) and sometimes specifically point out the use of NS instance priorities.

For details on the Os-Ma-nfvo reference point see ETSI GS NFV-IFA 013 [i.13]. Templates for NSD Management are described in ETSI GS NFV-IFA 014 [i.17], templates for VNF Package Management in ETSI GS NFV-IFA 011 [i.19].

D.2 General use cases

D.2.1 Use case for creating an NS instance

D.2.1.1 Introduction

The goal of the use case is to support the creation of a network slice via the NS construct, in order to allocate the necessary virtualised resources for the network slice instance.

The Network Slice Management Function of the consumer has determined that an NS instance is required for the creation of a Network Slice instance. In this use case the NS instance creation is based on an NSD that was already on-boarded with the NFVO.

The tenant/consumer information is retained in the NS instance runtime information.

NOTE 1: This scenario applies the same in the case of Network Subnet Slice, as it is transparent to the NFVO how the consumer uses the NS instance.

NOTE 2: The Consumer may decide to reuse the NS instance for another network slice instance(s), or network subnet instance(s) that have identical resources and SLA requirements, but this is transparent to NFVO.

NOTE 3: This use case covers also the case where the exposure of the NS instance to other tenants is handled via the single Consumer tenant (hence the other tenants would be transparent to NFV-MANO).

D.2.1.2 Trigger

Table D.2.1.2-1 describes the use case trigger.

Table D.2.1.2-1: Network Service created for Network Slicing, trigger

Trigger	Description
NFVO receives a request to instantiate an NS.	The Consumer of the ETSI GS NFV-IFA 013 [i.13] LCM interface (e.g. OSS, 3GPP Management System, or network slice management functions), requests the NFVO to instantiate an NS.

D.2.1.3 Actors and roles

Table D.2.1.3-1 describes the use case actors and roles.

Table D.2.1.3-1: Network Service created for Network Slicing, actors and roles

#	Actor	Description
1	NFVO	NFV Orchestrator for the NS instances involved.
2	Consumer	OSS, or other management system, e.g. network slice management. The Consumer acts as tenant for the instantiated network services.

D.2.1.4 Pre-conditions

Table D.2.1.4-1 describes the pre-conditions.

Table D.2.1.4-1: Network Service created for Network Slicing, pre-conditions

#	Pre-condition	Description
1	The necessary descriptors and packages are onboarded.	
2	• NFV-MANO (VIM, NFVO and VNFM) is running.	

D.2.1.5 Post-conditions

Table D.2.1.5-1 describes the post-conditions for base flow #1 (i.e. BF#1).

Table D.2.1.5-1: Network Service created for Network Slicing, post-conditions

#	Post-condition	Description
1	The NS instance is in INSTANTIATED state and can further be lifecycle managed by the NFVO.	
2	The Consumer of the NS instance is notified about success of the NS instance creation.	

D.2.1.6 Operational Flows

Table D.2.1.6-1 describes the base flow for the NS instance that is created and instantiated by a Consumer for purposes of using it as a building block for a network slice, or for a network subnet slice.

Table D.2.1.6-1: Network Service created for Network Slicing, base flow

#	Flow	Description
0	Consumer -> NFVO	The NFVO receives the trigger: The Consumer requests to instantiate an NS.
1	NFVO	The NFVO performs the steps described in ETSI GS NFV-IFA 013 [i.13], clause 7.3.2 "Create NS instance identifier operation" to create the NS instance ID.
2	NFVO -> Consumer	The NFVO returns the NS instance ID to the Consumer.
3	Consumer -> NFVO	The NFVO receives a request from the Consumer to instantiate the NS instance.
4	NFVO -> Consumer	The NFVO returns to the Consumer upon successful result the lifecycleOperationOccurrenceId.
5	NFVO -> Consumer	The NFVO sends the "start" Lifecycle Change Notification as per NsLifecycleChangeNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
6	NFVO	The NFVO instantiates the NS instance as described in ETSI GS NFV-IFA 013 [i.13], clause 7.3.3 "Instantiate NS operation".
7	NFVO -> Consumer	Upon successful, as well as unsuccessful, completion of the operation, the NFVO sends the "result" lifecycle operation occurrence notification to the Consumer as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.

D.2.2 Use case NS scaling

D.2.2.1 Introduction

The goal of the use case is to demonstrate, using the example of scaling, how LCM operations of an NS instance can be affected by priorities.

D.2.2.2 Trigger

Table D.2.2.2-1 describes the use case trigger.

Table D.2.2.2-1: Network Service Scaling, trigger

	Trigger	Description
BF#1	NFVO receives a request to scale an NS instance	Scaling operations can be triggered by the Consumer of the NS instance (e.g. OSS, 3GPP Management System, or network slice management functions).
BF#2	Internal decision of the NFVO	Scaling operations can be triggered by NFVO decisions, e.g. policy enforcement.
BF#3	VNFM receives a scaling request or takes an auto-scale decision.	The VNFM may receive a scaling request from the EM or VNF as described in ETSI GS NFV-IFA 008 [i.16], clause 7.2.4 or take an auto-scale decision.

D.2.2.3 Actors and roles

Table D.2.2.3-1 describes the use case actors and roles.

Table D.2.2.3-1: Network Service scaling, actors and roles

#	Actor	Description
1	NFVO	NFV Orchestrator for the NS instances involved.
2	Consumer	OSS, or other management system, e.g. network slice management. The Consumer acts as tenant for the instantiated network services.
3	LCM providing FB (e.g. VIM, VNFM, WIM, NFVO)	Depending on the type of service resource, life cycle including scaling is managed by different functional blocks of the NFV reference architecture: - NFVI resources are managed by the VIM. - VNFs are managed by the VNFM. - VL between NFVI-PoPs are managed by a WIM. - Nested NSs are managed by another NFVO.
4	VNFM	VNFM is in charge of the VNF.

D.2.2.4 Pre-conditions

Table D.2.2.4-1 describes the pre-conditions.

Table D.2.2.4-1: Network Service scaling, preconditions

#	Pre-condition	Description
1	The NS instance is in INSTANTIATED state and can be lifecycle managed by the NFVO.	
2	• NFV-MANO (VIM, NFVO and VNFM) is running.	
3	• Priority and other constraints are defined for the NS instance.	

D.2.2.5 Post-conditions

Table D.2.2.5-1 describes the post-conditions for base flow #1 (i.e. BF#1).

Table D.2.2.5-1: Network Service scaling, postconditions

#	Post-condition	Description
1	The scaling operation has been performed on the NS instance and the new configuration/status has been reached.	
2	The Consumer of the NS instance is notified about success of the NS instance scaling.	

D.2.2.6 Operational Flows

Table D.2.2.6-1 describes the base flow #1 (BF#1) for the NS instance that is scaled by a Consumer.

Table D.2.2.6-1: Network Service scaling, base flow #1

#	Flow	Description
0	Consumer -> NFVO	The NFVO receives the trigger: The Consumer requests a scaling operation of the NS instance. The Consumer provides parameters as described in ETSI GS NFV-IFA 013 [i.13], clause 7.3.4.2.
1	NFVO	The NFVO checks whether the scaling request is valid.
2	NFVO -> Consumer	The NFVO sends the "start" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
3	NFVO	The NFVO evaluates the necessary resources/VNFs, analysing whether LCM on the affected service resources can be allowed. This includes check against the models of virtualised resource management described in clause 5.1, in particular whether the priority of the NS instance in relation to other pending operations allows immediate or deferred execution of the scaling operation.
4	NFVO -> LCM providing FB	The NFVO triggers the necessary operations on the affected service resources/nested NSs. These operations may include: - Allocation/release of NFVI resources via a VIM - LCM operations on VNFs via a VNFM - LCM operations on a VL via a VIM or WIM - LCM operations on a nested NS via an NFVO
5	LCM providing FB	The LCM providing FB performs the scaling.
6	LCM providing FB -> NFVO	The LCM providing FB notifies the NFVO about completion of the scaling operation, e.g.: - The VNFM would send VnfLcmOperationOccurrenceNotification to NFVO to indicate completion of the operation as described in ETSI GS NFV-IFA 007 [i.15], clause 8.6.2. - The VIM would send one of the resource change notifications mentioned in ETSI GS NFV-IFA 005 [i.14], clause 5.3.8.
7	NFVO -> Consumer	Upon successful, as well as unsuccessful, completion of the operation, the NFVO sends the "result" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.

Table D.2.2.6-2 describes the base flow #2 (BF#2) for the NS instance that is scaled by decision of the NFVO.

Table D.2.2.6-2: Network Service scaling, base flow #2

#	Flow	Description
0	NFVO	The scaling operations is triggered by an internal decision of the NFVO.
1	NFVO	The NFVO checks whether the scaling request is valid.
2	NFVO -> Consumer	The NFVO sends the "start" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
3	NFVO	The NFVO evaluates the necessary resources/VNFs, analysing whether LCM on the affected service resources can be allowed. This includes check against the models of virtualised resource management described in clause 5.1, in particular whether the priority of the NS instance in relation to other pending operations allows immediate or deferred execution of the scaling operation.
4	NFVO -> LCM providing FB	The NFVO triggers the necessary operations on the affected service resources/nested NSs. These operations may include: - Allocation/release of NFVI resources via a VIM - LCM operations on VNFs via a VNFM - LCM operations on a VL via a VIM or WIM - LCM operations on a nested NS via an NFVO
5	LCM providing FB	The LCM providing FB performs the scaling.
6	LCM providing FB -> NFVO	The LCM providing FB notifies the NFVO about completion of the scaling operation, e.g.: - The VNFM would send VnfLcmOperationOccurrenceNotification to NFVO to indicate: completion of the operation as described in ETSI GS NFV-IFA 007 [i.15], clause 8.6.2. - The VIM would send one of the resource change notifications mentioned in ETSI GS NFV-IFA 005 [i.14], clause 5.3.8.
7	NFVO -> Consumer	Upon successful, as well as unsuccessful, completion of the operation, the NFVO sends the "result" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.

Table D.2.2.6-3 describes the base flow #3 (BF#3) for the NS instance that is scaled triggered via the VNFM.

Table D.2.2.6-3: Network Service scaling, base flow #3

#	Flow	Description
0	VNFM	The VNFM may receive a scaling request from the EM or VNF as described in ETSI GS NFV-IFA 008 [i.16], clause 7.2.4 or take an auto-scale decision.
1	VNFM	The VNFM checks whether the scaling request is valid.
2	VNFM -> NFVO	The VNFM sends a Grant VNF Lifecycle Operation operation to NFVO as described in ETSI GS NFV-IFA 007 [i.15], clause 6.3.2.
3	NFVO	The NFVO checks whether the granting request is valid.
4	NFVO -> Consumer	The NFVO sends the "start" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2 to the Consumer of the NS instance, in case of network slicing typically a network slicing management function in the OSS.
5	NFVO	The NFVO evaluates the necessary resources/VNFs, analysing whether LCM on the affected service resources can be allowed. This includes check against the models of virtualised resource management described in clause 5.1, in particular whether the priority of the NS instance in relation to other pending operations allows immediate or deferred execution of the scaling operation.
6	NFVO -> VNFM	The NFVO replies to the VNFM with the GrantVnfLifecycleOperationResponse as described in ETSI GS NFV-IFA 007 [i.15], clause 6.3.2.1.
7	VNFM	The VNFM performs the scaling.
8	VNFM -> NFVO	The VNFM sends a VnfLcmOperationOccurrenceNotification to NFVO to indicate completion of the operation as described in ETSI GS NFV-IFA 007 [i.15], clause 8.6.2.
9	NFVO -> Consumer	Upon successful, as well as unsuccessful, completion of the operation, the NFVO sends the "result" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.

D.2.3 Use case: Re-instantiation of multiple NS instances with different priorities after NFVI failure

D.2.3.1 Introduction

The goal of the use case is to demonstrate how priorities as introduced for network slicing can help when re-establishing service by re-instantiating multiple NS instances. Such situation can happen for instance after a failure of an NFVI PoP.

D.2.3.2 Trigger

Table D.2.3.2-1 describes the use case trigger.

Table D.2.3.2-1: Re-instantiation of multiple NS instances with different priorities after NFVI failure, trigger

Trigger	Description
Start re-instantiation	The re-instantiation can be triggered by the Consumer or by an automatic decision in NFVO, e.g. via a policy.

D.2.3.3 Actors and roles

Table D.2.3.3-1 describes the use case actors and roles.

Table D.2.3.3-1: Re-instantiation of multiple NS instances with different priorities after NFVI failure, actors and roles

#	Actor	Description
1	NFVO	NFV Orchestrator for the NS instances involved.
2	Consumer	OSS, or other management system, e.g. network slice management. The Consumer acts as tenant for the instantiated network services.
3	VNFM	VNFM in charge of the VNF instances that need to be re-instantiated.

D.2.3.4 Pre-conditions

Table D.2.3.4-1 describes the pre-conditions.

Table D.2.3.4-1: Re-instantiation of multiple NS instances with different priorities after NFVI failure, preconditions

#	Pre-condition	Description
1	Multiple NS instances that were in INSTANTIATED state transit to the NOT_INSTANTIATED state due to the resources they use being lost and thus need to be re-instantiated (i.e. NS healing is not possible).	
2	Priority and other constraints are defined for the NS instances.	It is assumed that the NS instances are defined with different priority values. Nested NS instances usually have the same priority as the parent NS, although it is not necessary. If multiple NSs have the same priority, there need to be other ways to decide which NS to instantiate. This is out of scope for this use-case.
3	NFV-MANO (VIM, NFVO and VNFM) is still running after the NFVI failure (or has been already re-established).	
4	The decision has been made that NS instances need to be re-instantiated. The set of NS instances to be re-instantiated may be different to the set of NS instances impacted by the outage.	In some configurations, there might be another set of NS instances or VNFs available that could replace the failed ones. This use case assumes that fallback to another set of VNF instances in another NFVI-PoP is not possible.
5	The Consumer already has been notified about the NFVI failure and the affected NS instances, VNFs, etc.	

D.2.3.5 Post-conditions

Table D.2.3.5-1 describes the post-conditions for base flow #1 (i.e. BF#1).

Table D.2.3.5-1: Re-instantiation of multiple NS instances with different priorities after NFVI failure, base flow #1, post-conditions

#	Post-condition	Description
1	The NS instances have been re-instantiated.	
2	The Consumers of the NS instances are notified about success of the re-instantiation.	

Table D.2.3.5-2 describes the post-conditions for base flow #2 (i.e. BF#2).

Table D.2.3.5-2: Re-instantiation of multiple NS instances with different priorities after NFVI failure, post-conditions

#	Post-condition	Description
1	The NS instances with higher priority have been re-instantiated.	In case of no resource shortage these are all NS instances required.
2	The Consumers are notified which NS instances were successfully instantiated.	Other failures during the re-instantiation are out of scope for this use case.
3	In case of resource shortage some NS instances could not be re-instantiated.	It is assumed that due to the NFVI failure, there are not enough resources available to re-instantiate all the required NS instances.
4	The Consumers are notified which NS instances were not instantiated.	

D.2.3.6 Operational Flows

Depending on resource need and availability different flows are possible:

Table D.2.3.6-1: Re-instantiation of multiple NS instances with different priorities after NFVI failure, base flows

Base flow	Description
BF#1	Due to an NFVI failure, the resources used by the constituents of multiple NS instances are lost and these NSs need to be re-instantiated. The order in which these resources are re-instantiated follows the priorities of the NS instances using them.
BF#2	Due to an NFVI failure, the resources used by the constituents of multiple NS instances are lost and these NSs need to be re-instantiated, but the available resources are not sufficient to re-instantiate all NS instances. The priority is used to decide which NS instances are instantiated.
NOTE: In many cases, NFVO can instantiate the NS instances with priority higher than a certain value. In some cases, however, the next NS instance to be instantiated might have high resource need while the need of lower priority NS instances could still be satisfied. The decision for this case is out of scope for this use case.	

Table D.2.3.6-2 describes the base flow #1 (BF#1) for re-instantiation after NFVI failure, when the available resources are sufficient for all NS instances affected by the NFVI failure.

Table D.2.3.6-2: Re-instantiation of multiple NS instances with different priorities after NFVI failure, base flow #1

#	Flow	Description
0	Trigger to start re-instantiation	The re-instantiation can be triggered by the Consumer or by an automatic decision in NFVO, e.g. via a policy.
1	NFVO	The NFVO evaluates the list of NS instances are to be re-instantiated.
2	NFVO -> Consumer	The NFVO sends the "start" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
3	NFVO	The NFVO triggers the necessary operations to get the NS constituents re-instantiated, starting with the NS instance with higher priority. This includes requesting the VNFM to re-instantiate the constituent VNFs. See note.
4	NFVO -> Consumer	As soon as an NS instance is re-instantiated, NFVO sends a notification to its Consumer: the "result" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
NOTE: The operations are identical to the first instantiation.		

Table D.2.3.6-3 describes the base flow #2 (BF#2) for re-instantiation after NFVI failure, when the available resources are not sufficient for all NS instances affected by the NFVI failure. The first 4 steps are identical to BF#1.

Table D.2.3.6-3: Re-instantiation of multiple NS instances with different priorities after NFVI failure, base flow #2

#	Flow	Description
0	Trigger to start re-instantiation	The re-instantiation can be triggered by the Consumer or by an automatic decision in NFVO, e.g. via a policy.
1	NFVO	The NFVO evaluates the list of NS instances to be re-instantiated.
2	NFVO -> Consumer	The NFVO sends the "start" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
3	NFVO	The NFVO triggers the necessary operations to get the NS constituent re-instantiated, starting with the NS instance with higher priority. See note 1.
4	NFVO -> Consumer	As soon as an NS instance is re-instantiated, NFVO sends a notification to its Consumer: the "result" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
5	NFVO	At a certain point, NFVO will detect that available resources are not sufficient for the instantiation of the next NS instance. Thus, it has to abandon the re-instantiation of the remaining NS instances. See note 2.
6	NFVO -> Consumer	The NFVO sends a notification to the Consumers of the NS instances that could not be re-instantiated notifying them about the resource shortage as per NsLcmCapacityShortageNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.5. See note 3. The NFVO sends a notification to the Consumers of the NS instances that could not be re-instantiated: the "result" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.

NOTE 1: The operations are identical to the first instantiation.
NOTE 2: In many cases, NFVO can instantiate the NS instances with priority higher than a certain value. In some cases, however, the next NS instance to be instantiated might have high resource need while the need of lower priority NS instances could still be satisfied. The decision for this case is out of scope for this use case.
NOTE 3: The NsLcmCapacityShortageNotification allows to provide detailed information about the shortage and also allows to notify the same consumer later that the resource shortage situation has ended and the LCM operation could be successful in case the Consumer tries again.

D.2.4 Use case: Instantiation of NS in parallel to other LCM operations

D.2.4.1 Introduction

The goal of the use case is to demonstrate how NS instance priorities will be used when multiple NS LCM operations are running in parallel, so there may be several resource requests and high priority NS instances should be served before low priority NS instances.

Since NS LCM operations, especially instantiation and scaling, may result in many operations by VIM, VNFM, WIM and even VNF, they may be long running operations. Thus, the probability of parallel operations could be high. A restriction to execute only one operation at a time (sequential execution of LCM operations) eliminates the potential concurrency problems, but may reduce the usability of NFV and thus cannot be assumed.

This use cases discusses an incoming NS instantiation request while other NS LCM operations including their resource requests are being executed. However, similar conditions may happen for instance during NS scaling.

D.2.4.2 Trigger

Table D.2.4.2-1 describes the use case trigger.

Table D.2.4.2-1: Instantiation of NS in parallel to other LCM operation, trigger

Trigger	Description
NFVO receives a request to instantiate an NS	The Consumer of the ETSI GS NFV-IFA 013 [i.13] LCM interface (e.g. OSS, 3GPP Management System, or network slice management functions), requests the NFVO to instantiate an NS. As described in the introduction, NFVO is already executing another NS LCM operation, and a resource shortage can be foreseen.

D.2.4.3 Actors and roles

Table D.2.4.3-1 describes the use case actors and roles.

Table D.2.4.3-1: Instantiation of NS in parallel to other LCM operation, actors and roles

#	Actor	Description
1	NFVO	NFV Orchestrator for the NS instances involved.
2	Consumer	OSS, or other management system, e.g. network slice management The Consumer acts as tenant for the instantiated network services.
3	VNFM	VNFM in charge of the VNF instantiation or other VNF LCM operation.

D.2.4.4 Pre-conditions

Table D.2.4.4-1 describes the pre-conditions.

Table D.2.4.4-1: Instantiation of NS in parallel to other LCM operation, preconditions

#	Pre-condition	Description
1	The necessary descriptors and packages are onboarded.	
2	NFV-MANO (VIM, NFVO and VNFM) is running.	
3	• Priority values (and other constraints) are known for all affected NS instances.	It is assumed that all NS instances handled by the NFVO have certain priority values. This Use Case focuses on conflict resolution scenario where priority values of all participating NS instances are different. For the NS instances with the same priority, there may be other ways to resolve resource conflicts (e.g. first come - first serve), these are out of scope for this use-case.
4	• A long running NS LCM operation is in execution, while the trigger occurs.	The on-going NS LCM operation associated with different NS instances also requests additional resources. In this use case it is assumed, that some resources have already been allocated, but the operation is not yet completed.
5	• The available resources are not sufficient to fulfil all requests.	The "all requests" here include the resource requests of on-going NS LCM operations together with the resource requests of the new NS instantiation operation. It is assumed (for simplicity) that the available resources were sufficient to fulfil resource requests of the on-going operation.

D.2.4.5 Post-conditions

Table D.2.4.5-1 describes the post-conditions.

Table D.2.4.5-1: Instantiation of NS in parallel to other LCM operation, post-conditions

#	Post-condition	Description
1	The LCM operation for the NS instance with higher priority is successfully executed.	The resource requests for the higher priority NS are fulfilled while the requests for the lower priority NS are rejected.
2	The LCM operation for the NS instance with lower priority is rejected and no resources are allocated.	This includes the case when the operation for the lower priority NS instance was already being executing, some resources have already been allocated or VNFs instantiated, but the NS LCM operation(s) have not been completed.
3	The Consumer of the NS instances with higher priority is notified about success.	
4	The Consumer of the NS instances with lower priority is notified that instantiation or other NS LCM request was not possible due to resource shortage.	These are just normal LCM notifications, see note.
NOTE: For the scope of this use case, the notification about the abandoned NS LCM operation may be the same or different to resource shortage notification.		

D.2.4.6 Operational Flows

This clause shows both options: the latest instantiation request can be higher priority or lower priority than the LCM operations that are already executing.

Table D.2.4.6-1 describes the two base flows.

Table D.2.4.6-1: Instantiation of NS in parallel to other LCM operation, base flows

Flow	Description
BF#1	NFVO receives a request to instantiate a lower priority NS instance while an LCM operation on a higher priority NS instance is being executed.
BF#2	NFVO receives a request to instantiate a higher priority NS instance while an LCM operation on a lower priority NS instance is being executed.

Table D.2.4.6-2 describes the base flow #1.

Table D.2.4.6-2: Instantiation of NS instance in parallel to other LCM operation, base flow #1

#	Flow	Description
0	Consumer -> NFVO	The NFVO receives the trigger: The Consumer requests to instantiate an NS.
1	NFVO	The NFVO checks whether the scaling request is valid.
2	NFVO -> Consumer	The NFVO sends the "start" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
3	NFVO	The NFVO evaluates the request: - calculate necessary resources for the new NS instance; - consider the available resources and the pending resource requests from LCM operations currently being executed; - determine that the resources requests by the new NS instance cannot be fulfilled; - compare priority of the new NS instantiation requests and the NS instances of the LCM operations currently being executed; - determine that the new request has lower priority than the NS instances of the LCM operations currently being executed; - decide to reject instantiation request.
4	NFVO -> Consumer	The NFVO sends a notification to the Consumers of the NS instances that cannot be re-instantiated notifying them about the resource shortage as per NsLcmCapacityShortageNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.5. See note 3. The NFVO sends a notification to the Consumer that the NS cannot be instantiated because of resource shortage as per "result" NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2. See note 2.

#	Flow	Description
NOTE 1:	This flow is identical to the case when the priorities of all NS instances are the same. The requests of NS LCM operations are executed first come first serve.	
NOTE 2:	See also the use case in clause D.2.5. Pre-emption of already running NS instances is out of scope for this use case.	
NOTE 3:	The NsLcmCapacityShortageNotification allows to provide detailed information about the shortage and also allows to notify the same consumer later that the resource shortage situation has ended and the LCM operation could be successful in case the Consumer tries again.	

Table D.2.4.6-3 describes the base flow #2.

Table D.2.4.6-3: Instantiation of NS instance in parallel to other LCM operation, base flow #2

#	Flow	Description
0	Consumer -> NFVO	The NFVO receives the trigger: The Consumer requests to instantiate an NS.
1	NFVO	The NFVO checks whether the scaling request is valid.
2	NFVO -> Consumer	The NFVO sends the "start" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
3	NFVO	The NFVO evaluates the request: - calculate the necessary resources for the new NS instance; - consider the available resources and the pending resource requests from LCM operations currently being executed; - determine that the resources requests by the new NS instance cannot be fulfilled; - compare priority of the new NS instantiation requests and the NS instances of the LCM operations currently being executed; - determine that the new request has higher priority than the NS instances of the LCM operations currently being executed; - calculate that the new request could be fulfilled if one (or more) operations for lower priority NS instances are stopped, see notes 1 and 2; - decide to abandon running LCM operation(s) and rollback the instantiation of constituents already done for the lower priority NS instances.
4	NFVO -> Consumer	The NFVO sends the CoordinateLcmOperation request to the Consumer of the NS instances that are to be terminated or scaled in as described in ETSI GS NFV-IFA 013 [i.13], clause 6.1.2. See note 7.
5	Consumer -> NFVO	The Consumer sends the CoordinateLcmOperationResponse with Action= "CONTINUE", allowing the NFVO to proceed with the LCM operation on the lower priority NS instances. See notes 8 and 9.
6	NFVO -> VNFM	The NFVO issues the appropriate commands to VNFM to abandon the running LCM operations for the lower priority NS instances and rollback the instantiation of constituents already done for the lower priority NS instances, see note 3. In some cases, this includes to issue termination commands to VNFM to terminate VNF instances that were already fully instantiated as part of the LCM operation(s) for the lower priority NS instances.
7	NFVO & VNFM	The NFVO and VNFM deallocate all resources already allocated during the abandoned LCM operations for the lower priority NS instances, see note 4.
8	NFVO -> Consumer of lower priority NS instance	The NFVO sends a notification to the Consumers of the NS instance affected by abandoning the LCM operation notifying them about the resource shortage as per NsLcmCapacityShortageNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.5. See note 10. The NFVO sends a notification to the Consumer of the lower priority NS instances that the LCM operation could not be executed because of resource shortage, as per "result" NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2. See notes 5 and 6.
9	NFVO -> VNFM	The NFVO triggers the necessary operations to instantiate the higher priority NS instances, see note 3.
10	VNFM -> NFVO	The VNFM notifies the NFVO that the requested VNFs are instantiated.
11	NFVO -> Consumer of new NS instance with higher priority	The NFVO sends a notification to the Consumer of the higher priority NS instance that it is instantiated as per "result" NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.

#	Flow	Description
NOTE 1:	There may be more complex situations if multiple operations are currently executing in parallel and NFVO needs to analyse by priority and resource availability, which of the operations should be abandoned or allowed to proceed.	
NOTE 2:	Also, resource reservation needs to be considered, but this is not part of this use case. The assumption here is that there are no resource reservations for the NS instances.	
NOTE 3:	Instantiation and other NS LCM operations may include VLs and nested NS instances which are not shown here.	
NOTE 4:	This use case does not distinguish between indirect and direct mode for the resource allocation.	
NOTE 5:	The NS LCM operation in execution might be triggered not by OSS but by some internal trigger. In that case also a notification to the Consumer of the low priority NS instance is necessary that LCM operations could not be executed.	
NOTE 6:	For the scope of this use case, the notification about the abandoned LCM operation may be the same or different to resource shortage notification.	
NOTE 7:	The LCM coordination interface includes several responses including delay times in the CoordinateLcmOperationResponse. In this use case it is assumed that the Consumer allows the NFVO to proceed.	
NOTE 8:	In case the Consumer does not allow the operation, NFVO will reject the initial request for lack of resources, or tries to find another NS instance with lower priority that could be terminated. Details are out of scope for this use case.	
NOTE 9:	For other responses of the LCM coordination interface see ETSI GS NFV-IFA 013 [i.13], clause 6.1.2 and clause F.1.	
NOTE 10:	The NsLcmCapacityShortageNotification allows to provide detailed information about the shortage and also allows to notify the same consumer later that the resource shortage situation has ended and the LCM operation could be successful in case the Consumer tries again.	

D.2.5 Use case: Resolve resource allocation conflict by pre-empting a lower priority NS instance that is up and running

D.2.5.1 Introduction

The goal of the use case is to demonstrate how NS instance priorities will be used to resolve a resource allocation conflict during NS instantiation. NFV-MANO here determines resource shortage during the instantiation and decides to provide resources for the higher priority NS instance by terminating or scaling in a lower priority NS instance.

This use cases discusses an incoming NS instantiation request while similar conditions may happen for instance during NS scaling or healing.

D.2.5.2 Trigger

Table D.2.5.2-1 describes the use case trigger.

Table D.2.5.2-1: Resolve resource allocation conflict by pre-empting a lower priority NS instance that is up and running, trigger

Trigger	Description
NFVO receives a request to instantiate an NS.	The Consumer of the ETSI GS NFV-IFA 013 [i.13] LCM interface (e.g. OSS, 3GPP Management System, or network slice management functions), requests the NFVO to instantiate an NS. As described in the introduction, a resource shortage can be foreseen.

D.2.5.3 Actors and roles

Table D.2.5.3-1 describes the use case actors and roles.

Table D.2.5.3-1: Resolve resource allocation conflict by pre-empting a lower priority NS instance that is up and running, actors and roles

#	Actor	Description
1	NFVO	NFV Orchestrator for the NS instances involved.
2	Consumer	OSS, or other management system, e.g. network slice management. The Consumer acts as tenant for the instantiated network services.
3	VNFM	VNFM in charge of the VNF instantiation or other VNF LCM operation.

D.2.5.4 Pre-conditions

Table D.2.5.4-1 describes the pre-conditions.

Table D.2.5.4-1: Resolve resource allocation conflict by pre-empting a lower priority NS instance that is up and running, preconditions

#	Pre-condition	Description
1	The necessary descriptors and packages are onboarded.	
2	Priority values (and other constraints) are known for all affected NS instances.	It is assumed that all NS instances handled by the NFVO have certain priority values. This use case focuses on conflict resolution scenario where priority values of all participating NS instances are different.
3	The available resources are not sufficient to fulfil the instantiation request.	
4	Some NS instances with lower priority are running.	It is assumed that enough resources can be made available by scale in or terminating some of the lower priority NS instances.

D.2.5.5 Post-conditions

Table D.2.5.5-1 describes the post-conditions.

Table D.2.5.5-1: Resolve resource allocation conflict by pre-empting a lower priority NS instance that is up and running, post-conditions

#	Post-condition	Description
1	The LCM operation for the NS instance with higher priority is successfully executed.	The resource requests for the higher priority NS are fulfilled.
2	Some lower priority NS instance was forcefully scaled in or even terminated to provide the resource for the higher priority NS.	The affected lower priority NS instance in many cases will enter some overload situation.
3	The Consumer of the NS instances with higher priority is notified about successful instantiation of the NS by pre-empting a lower priority NS.	
4	The Consumer of the NS instances with lower priority is notified that the instance was forcefully scaled in or terminated due to a resource shortage and conflict with a higher priority NS.	

D.2.5.6 Operational Flows

Table D.2.5.6-1 describes the operational flow.

Table D.2.5.6-1: Resolve resource allocation conflict by pre-empting a lower priority NS instance that is up and running, base flow

#	Flow	Description
0	Consumer -> NFVO	The NFVO receives the trigger: The Consumer requests to instantiate an NS.
1		The NFVO sends the "start" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
2	NFVO	The NFVO evaluates the request: - calculate the necessary resources for the new NS instance; - determine that the resources required by the new NS instance cannot be fulfilled; - compare priority of the NS for which the instantiation request has been received with the already instantiated NS instances; - find running NS instances with lower priority that could be scaled in or terminated so that the necessary resources could be made available. See notes 1 and 2. The NFVO creates an LCM operation on the NS instance with lower priority that executes the necessary scale in or termination.
3	NFVO -> Consumer	The NFVO sends the CoordinateLcmOperation request to the Consumer of the NS instances that are to be terminated or scaled in as described in ETSI GS NFV-IFA 013 [i.13], clause 6.1.2. See note 5.
4	Consumer -> NFVO	The Consumer sends the CoordinateLcmOperationResponse with Action= "CONTINUE", allowing the NFVO to proceed with the LCM operation on the lower priority NS instances. See notes 6 and 7.
5	NFVO -> Consumer of lower priority NS instance	The NFVO sends the "start" lifecycle operation occurrence notification as per NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
6	NFVO -> VNFM	The NFVO issues the appropriate commands to VNFM to scale in or terminate VNF instances serving the lower priority NS instance. See note 1.
7	NFVO & VNFM	The NFVO and VNFM deallocate the resources accordingly.
8	NFVO -> Consumer of lower priority NS instance	The NFVO sends a notification to the Consumers of the NS instances that were forcefully scaled in or terminated, notifying them about the resource shortage as per NsLcmCapacityShortageNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.5. See note 8. The NFVO sends a notification to the Consumer of the NS instances that were forcefully scaled in or terminated, notifying them about the complete scale in or termination as per "result" NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.
9	NFVO -> VNFM	The NFVO triggers the necessary operations to instantiate the higher priority NS instances, see notes 3 and 4.
10	VNFM -> NFVO	The VNFM notifies the NFVO that the requested VNFs are instantiated.
11	NFVO -> Consumer of new NS instance with higher priority	The NFVO sends a notification to the Consumer of the higher priority NS instance that it is instantiated as per "result" NsLcmOperationOccurrenceNotification in ETSI GS NFV-IFA 013 [i.13], clause 8.3.2.2.

NOTE 1: NSD and VNFD describe the possible ways of scaling the affected NS or VNF.
NOTE 2: The algorithm to decide which NS instances to scale in or terminate might be complex and is outside this use case.
NOTE 3: Instantiation and other NS LCM operations may include VLs and nested NS instances which are not shown here.
NOTE 4: In this use case it is assumed that no other LCM requests are received during the instantiation.
NOTE 5: The LCM coordination interface includes several responses including delay times in the CoordinateLcmOperationResponse. In this use case it is assumed that the Consumer allows the NFVO to proceed.
NOTE 6: In case the Consumer does not allow the operation, NFVO will reject the initial request for lack of resources, or tries to find another NS instance with lower priority that could be terminated. Details are out of scope for this use case.
NOTE 7: For other responses of the LCM coordination interface see ETSI GS NFV-IFA 013 [i.13], clause 6.1.2 and clause F.1.
NOTE 8: The NsLcmCapacityShortageNotification allows to provide detailed information about the shortage and also allows to notify the same consumer later that the resource shortage situation has ended and the LCM operation could be successful in case the Consumer tries again.

D.3 NS management supporting network slicing

D.3.1 Introduction

This annex describes how NFV will support network slicing via NFV Network Services. Features of NFV Network Service and NFV-MANO definitions that can be used to support network slicing are described and evaluated.

The functions that are managing network slicing will use the NFV-MANO (Os-Ma-Nfvo) reference point to request and manage NFV Network Service instances. The same reference point is used to control performance, privacy and other advanced functions needed for network slicing.

The NFV Network Service Descriptor contains related parameters for NFV Network Service instantiation.

This annex describes how NFV-MANO can satisfy the requirements specific to network slicing by the NFV Network Service Descriptor features. The functions that are managing network slicing will use the required parameters over the existing NFV reference points and interfaces.

The relationship between Network Slicing and the NFV constructs was studied in ETSI GR NFV-EVE 012 [i.12].

The following assumptions are made regarding sharing aspects of an NS instance, in context of network slicing:

1) The NFVO relies on the Consumer (e.g. OSS/NSMF/NSSMF) to track and handle the various tenants to which the Consumer allocates a specific NS instance. Therefore, the tenant(s) that are making use of any one NS instance are not known by NFVO. The NFVO does not need to handle tenant aspects related to an NS instance.

2) An NS instance may be shared between different network slices or network slices subnets, but the NFVO is not aware of how the Consumer is using the different NS instances. This means that the NFVO is not aware of which network slice instance(s) or network slice subnet(s) are making use of a specific NS instance.

D.3.2 NS instance sharing between Network Slices and tenants

The goal of the use case is to support sharing of resources between network slices with matching and sufficient resource requirements as expressed in the NSD, which is realized via sharing of the same NS instance.

The consumer (tenant) X has determined that an existing NS instance, used by tenant X as part of a network slice instance A, also fulfils the resource requirements for another network slice instance B.

The network slice instance B may belong to the same tenant X, or it may belong to a tenant Y that is handled by the consumer/tenant X.

The main tenant/consumer information (e.g. identity for tenant X) is retained by NFVO in the NS instance runtime information.

There are several NS sharing scenarios addressed:

1) The network slice instance B belongs to the same consumer (tenant) X as network slice instance A:

   a) In this case the NFVO is aware of the consumer X as the owner tenant for the NS instance, but it would not need to be privy to the information on the usage of the NS instance by the consumer/tenant X (i.e. if used for one or many network slice instances or network subnet slice instances):

      i) The use cases for this network slicing scenario are not new to NFV-MANO but are based on regular NS LCM operations as described in ETSI GS NFV-IFA 013 [i.13].

2) The network slice instance B belongs to a different consumer (tenant) Y, but the resource sharing aspects with other tenants such as tenant Y, are handled by consumer X (e.g. OSS):

   a) In this case the NFVO is unaware of the various tenants handled by X, and only interacts with consumer X as the sole tenant for the NS instance:

      i) The use cases for this network slicing scenario are not new to NFV-MANO but are based on regular NS LCM operations as described in ETSI GS NFV-IFA 013 [i.13].

Annex E (informative): Policy management in NFV-MANO

E.1 Introduction

Policy is one of the key enablers for constructing flexible management and orchestration functions in the NFV-MANO architecture. Assisted with policies, NFV-MANO functions can be provided with more automatic characteristics which fit in with the dynamic requirements of resource management and network service orchestration in the virtualised network environment.

NFV-MANO policies are mainly applicable to NFV-MANO reference points to assist for corresponding NFV-MANO functions like NS LCM, VNF LCM or resource management. NFV-MANO specific policy management use cases are investigated in ETSI GR NFV-IFA 023 [i.9], and operations of policy transfer, policy deletion, policy query, policy activation, policy deactivation and corresponding notification management are derived in policy management interface. Although policy management use cases for each NFV-MANO reference point are not exhaustively elaborated, the study recommends to enhance the existing NFV-MANO reference point with policy management interface, which finally supports the management of policies enforced by the NFVO, VNFM or VIM.

E.2 Scope of polices in NFV-MANO reference point

Table E.2-1 lists the category of NFV-MANO policy(ies) applied to each reference point. Corresponding functional description for the policy categories can refer to ETSI GR NFV-IFA 023 [i.9], clause 6.2 and clauses 7.2.2 to 7.2.6.

Table E.2-1: NFV-MANO policy(ies) on each reference point

Policy Category	Reference Point
NS instantiation policy	Os-Ma-nfvo (see clause 7.2.2 of ETSI GR NFV-IFA 023 [i.9])
NS scaling policy	Os-Ma-nfvo (see clause 7.2.2 of ETSI GR NFV-IFA 023 [i.9])
NS update policy	Os-Ma-nfvo (see clause 7.2.2 of ETSI GR NFV-IFA 023 [i.9])
NS healing policy	Os-Ma-nfvo (see clause 7.2.2 of ETSI GR NFV-IFA 023 [i.9])
NS termination policy	Os-Ma-nfvo (see clause 7.2.2 of ETSI GR NFV-IFA 023 [i.9])
VNF instantiation policy	Or-Vnfm (see clause 7.2.3 of ETSI GR NFV-IFA 023 [i.9])
	Ve-Vnfm-em (see clause 7.2.5 of ETSI GR NFV-IFA 023 [i.9])
VNF scaling policy	Or-Vnfm (see clause 7.2.3 of ETSI GR NFV-IFA 023 [i.9])
	Ve-Vnfm-em (see clause 7.2.5 of ETSI GR NFV-IFA 023 [i.9])
VNF healing policy	Or-Vnfm (see clause 7.2.3 of ETSI GR NFV-IFA 023 [i.9])
	Ve-Vnfm-em (see clause 7.2.5 of ETSI GR NFV-IFA 023 [i.9])
VNF termination policy	Or-Vnfm (see clause 7.2.3 of ETSI GR NFV-IFA 023 [i.9])
	Ve-Vnfm-em (see clause 7.2.5 of ETSI GR NFV-IFA 023 [i.9])
Virtualised resource allocation policy	Or-Vi (see clause 7.2.4 of ETSI GR NFV-IFA 023 [i.9])
	Vi-Vnfm (see clause 7.2.6 of ETSI GR NFV-IFA 023 [i.9])
Virtualised resource reservation policy	Or-Vi (see clause 7.2.4 of ETSI GR NFV-IFA 023 [i.9])
Virtualised resource quota (management) policy	Or-Vi (see clause 7.2.4 of ETSI GR NFV-IFA 023 [i.9])
Virtualised resource capacity (management) policy	Or-Vi (see clause 7.2.4 of ETSI GR NFV-IFA 023 [i.9])

Annex F (informative): VNF Snapshots

F.1 Introduction

The feature for VNF snapshotting is introduced to the NFV system by adding new or enhancing existing requirements, interfaces, operations, and information elements on multiple reference points of the NFV architecture. This annex provides further information on the concepts of VNF Snapshots and VNF Snapshot Packages and provides end-to-end procedures to illustrate the lifecycle management of these objects and the expected behaviour of the involved functional blocks.

F.2 VNF Snapshot lifecycle

ETSI GR NFV-TST 005 [i.10] introduces the general lifecycle of a VNF Snapshot in its clause 5.2.1. Figure F.2-1, based on Figure 5.2.1 from [i.10], illustrates the relationship and transitions between a VNF instance, a VNF Snapshot object and a VNF Snapshot Package object.

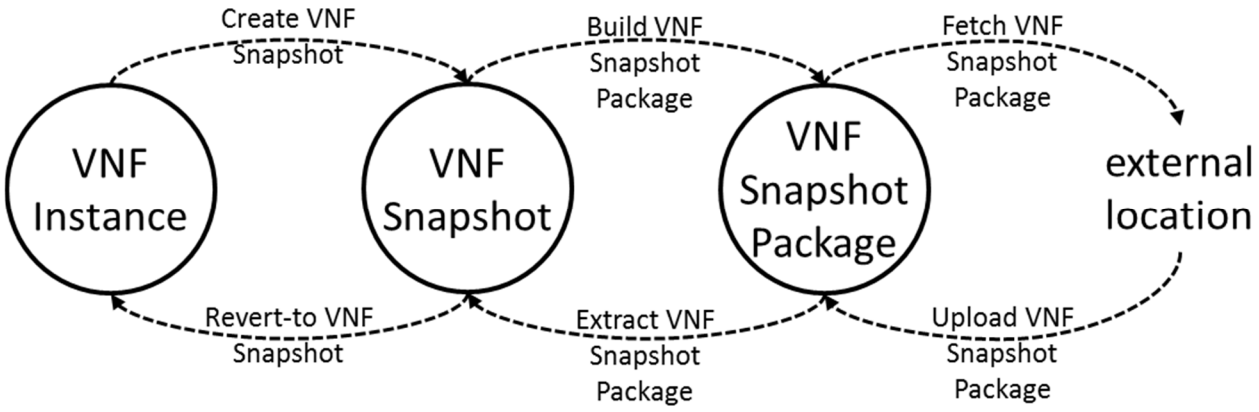

Figure F.2-1: Relationship and transitions between VNF, VNF Snapshot and VNF Snapshot Package

For definitions of the terms VNF Snapshot, VNFC Snapshot, VNF Snapshot Package refer to ETSI GR NFV 003 [i.2].

A VNF Snapshot object can either be generated by creating a VNF Snapshot from a VNF instance or by extracting a VNF Snapshot Package. An existing VNF Snapshot object can be used to revert a VNF instance to the state captured in this VNF Snapshot and it can be used to build a VNF Snapshot Package from it.

It is to note that a VNF Snapshot includes one to many VNFC Snapshots, which represent a replication of a VNFC instance at a specific point in time, capturing its full or partial state. Dependent on the implementation of the Virtualisation layer, a VNFC Snapshot may not be represented by physical accessible files or other storage objects but are represented by reference tags kept by the Virtualisation layer instead. Therefore, they can only be used to revert a VNF instance to a previously captured state and are not able to be directly exported to other systems.

The VNFM is the functional block responsible to maintain the VNF Snapshot objects, including exposure of operations for the lifecycle management of VNF Snapshots and keeping runtime information on existing VNF Snapshots. The runtime information on existing VNF Snapshots is kept in objects of the "VnfSnapshotInfo" information element, including one to many "VnfcSnapshotInfo" information elements.

Operations for the lifecycle management of VNF Snapshots are exposed by the VNFM via the VNF Lifecycle Management interfaces over the Or-Vnfm and Ve-Vnfm-em reference points and are further accessible via the "Update NS operation" of the NS Lifecycle Management interface exposed by the NFVO via the Os-Ma-nfvo reference point.

A VNF Snapshot Package object can either be generated by building a VNF Snapshot Package from a VNF Snapshot or by uploading a VNF Snapshot Package from an external location. An existing VNF Snapshot Package object can be extracted into a VNF Snapshot object and it can be fetched from an external location. A VNF Snapshot Package includes one to many VNFC Snapshot images together with VNF Snapshot runtime information and additional artifacts.

The NFVO is the functional block responsible to maintain the VNF Snapshot Package objects, including exposure of operations for the lifecycle management of VNF Snapshot Packages and keeping runtime information on existing VNF Snapshot Packages. The runtime information on existing VNF Snapshot Packages is kept in objects of the "VnfSnapshotPkgInfo" information element.

Operations for the lifecycle management of VNF Snapshot Packages are exposed by the NFVO via the VNF Snapshot Package Management interface over the Os-Ma-nfvo reference point.

F.3 VNF/VNFC Snapshot procedures

F.3.1 Introduction

This clause describes example end-to-end procedures concerning the operations for managing VNF/VNFC Snapshots. Because VNF Snapshots comprise of one to many VNFC Snapshots, the procedures for VNF Snapshots repeat common steps of the VNFC Snapshot procedures.

The VNF Snapshot procedure descriptions are based on an originating request from the OSS/BSS via the Os-Ma-nfvo reference point.

All procedure descriptions are illustrated by sequence charts which contain the names of the messages as specified in the respective interface specifications, including the relevant input and output parameters. The sequenced messages in the charts are numbered and are complemented by step descriptions with corresponding numbers.

F.3.2 Create VNF Snapshot procedure

The procedure to create a VNF Snapshot comprises of the following steps as depicted in Figures F.3.2-1 to F.3.2-3:

Step 1: The OSS/BSS initiates the CreateSnapshot operation and sends a request to update an NS to the NFVO, including the update type and the identifier of the VNF instance to be snapshotted.

Step 2: The NFVO determines the responsible VNFM for the VNF instance to be snapshotted, utilizing the indicated VNF instance identifier.

Step 3: The NFVO validates the policies for the CreateSnapshot operation for the indicated VNF instance. Policy rules could be provided by the VNFD of the VNF instance, for example allowing or denying taking Snapshots.

Step 4: The NFVO sends a request to create a VNF Snapshot to the VNFM, including the identifier of the VNF instance to be snapshotted.

Step 5: The VNFM creates a new object of a VnfSnapshotInfo information element and generates a unique VNF Snapshot identifier vnfSnapshotInfoId.

Step 6: In case the VNFD indicates that the VNF requires the preparation for VNF Snapshot creation, the VNFM sends a CoordinateLcmOperationRequest message to the VNF, indicating the start of a CreateSnapshot lifecycle operation. The VNFM pauses its execution of the CreateSnapshot operation and waits for a confirmation from the VNF.

Step 7: The VNF performs internal pre-snapshot procedures to prepare for taking a VNF Snapshot.

Step 8: The VNF sends a CoordinateLcmOperationResponse message to the VNFM to indicate the successful completion of the preparation for the VNF Snapshot.

Step 9: The VNFM resumes the execution of the CreateSnapshot operation and determines the parameters for the operation from the VNFD of the VNF instance. Those parameters indicate for example if the virtualised compute resource needs to be stopped prior to the snapshotting, if a virtualised storage resource needs to be detached, or if the filesystem of the virtualised compute resource needs to be quiesced.

Step 10: The VNFM determines the VNFC instances to be snapshotted and to be included in the VNF Snapshot.

The steps 11 to 31 are repeated for all identified VNFC instances to be snapshotted:

Step 11: The VNFM creates a new object of a VnfcSnapshotInfo information element and generates a unique VNFC Snapshot identifier vnfcSnapshotInfoId.

Step 12: The VNFM determines the identifiers of the virtualised compute and virtualised storage resources of the VNFC instance to be snapshotted from the VnfcResourceInfo information element.

Steps 13/14: If the parameters for the CreateSnapshot operation indicate that the virtualised compute resource of the VNFC instance needs to be stopped before snapshotting, the VNFM sends a request to the VIM to stop the indicated virtualised compute resource. The VIM sends a corresponding response after completion of the operation.

Steps 15/16: If the parameters for the CreateSnapshot operation indicate that a virtualised storage resource needs to be detached from the virtualised compute resource of the VNFC instance before snapshotting, the VNFM sends a request to the VIM to detach the indicated virtualised storage resource from the indicated virtualised compute resource. The VIM sends a corresponding response after completion of the operation.

Steps 17/18: If the parameters for the CreateSnapshot operation indicate that the file system of the virtualised compute resource of the VNFC instance needs to be quiesced before snapshotting, the VNFM sends a request to the VIM to quiesce the filesystem of the indicated virtualised compute resource. The VIM sends a corresponding response after completion of the operation.

Step 19: The VNFM sends a request to the VIM to create a snapshot of the indicated virtualised compute resource.

Step 20: The VIM returns a response to the VNFM upon completion of the snapshot of the indicated virtualised compute resource, including an identifier of the virtualised compute resource snapshot which serves as reference to the created snapshot of the virtualised compute resource.

Step 21: The VNFM stores the received identifier of the virtualised compute resource snapshot in the respective VnfcSnapshotInfo information element object.

Step 22: The VNFM sends a request to the VIM to create a snapshot of the indicated virtualised storage resource.

Step 23: The VIM returns a response to the VNFM upon completion of the snapshot of the indicated virtualised storage resource, including an identifier of the virtualised storage resource snapshot which serves as reference to the created snapshot of the virtualised storage resource.

Step 24: The VNFM stores the received identifier of the virtualised storage resource snapshot in the respective VnfcSnapshotInfo information element object.

Steps 25/26: If the file system of the virtualised compute resource of the VNFC instance has been quiesced before snapshotting, the VNFM sends a request to the VIM to unquiesce the filesystem of the indicated virtualised compute resource. The VIM sends a corresponding response after completion of the operation.

Steps 27/28: If a virtualised storage resource has been detached from the virtualised compute resource of the VNFC instance before snapshotting, the VNFM sends a request to the VIM to attach the indicated virtualised storage resource back to the indicated virtualised compute resource. The VIM sends a corresponding response after completion of the operation.

Steps 29/30: If the virtualised compute resource of the VNFC instance has been stopped before snapshotting and if the parameters for the CreateSnapshot operation indicate that it needs to be started after snapshotting, the VNFM sends a request to the VIM to start the indicated virtualised compute resource. The VIM sends a corresponding response after completion of the operation.

Step 31: The VNFM completes the data for the VnfcSnapshotInfo information element object, e.g. adds the VnfcInfo information element object for the respective VNFC instance.

Step 32: In case the VNFD indicates that the VNF requires a return to normal after VNF Snapshot creation, the VNFM sends a CoordinateLcmOperationRequest message to the VNF, indicating the end of a CreateSnapshot lifecycle operation. The VNFM pauses its execution of the CreateSnapshot operation and waits for a confirmation from the VNF.

Step 33: The VNF performs internal post-snapshot procedures to return to normal after taking a VNF Snapshot.

Step 34: The VNF sends a CoordinateLcmOperationResponse message to the VNFM to indicate the successful completion of the return to normal after the VNF Snapshot.

Step 35: The VNFM resumes the execution of the CreateSnapshot operation and completes the data for the VnfSnapshotInfo information element object, e.g. adds the VnfInfo and VNFD information element objects for the respective VNF instance.

Step 36: The VNFM completes the CreateSnapshot operation and sends the response to the NFVO, including an identifier of the stored VnfSnapshotInfo information element object.

Step 37: The NFVO sends the response to the originating request to the OSS/BSS, including an identifier of the stored VnfSnapshotInfo information element object. The VnfSnapshotInfoId can be used to reference the created VNF Snapshot in subsequent requests.

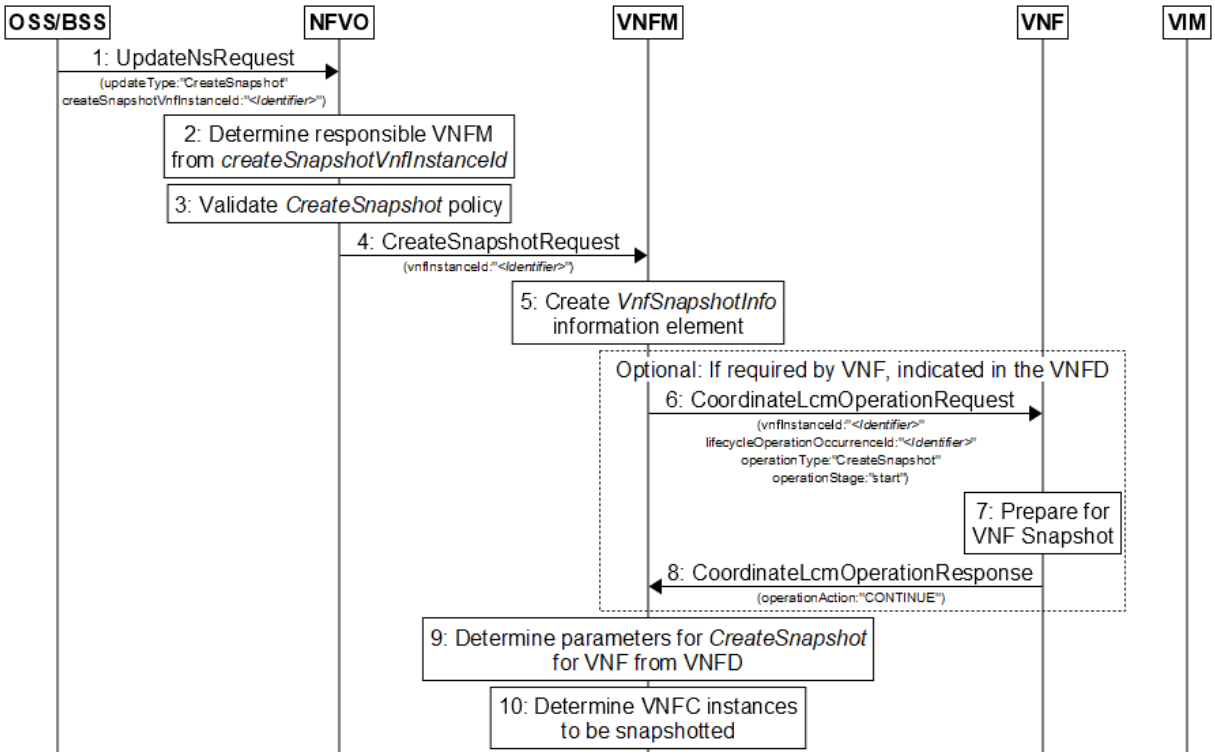

Figure F.3.2-1: Create VNF Snapshot triggered from OSS/BSS, part 1

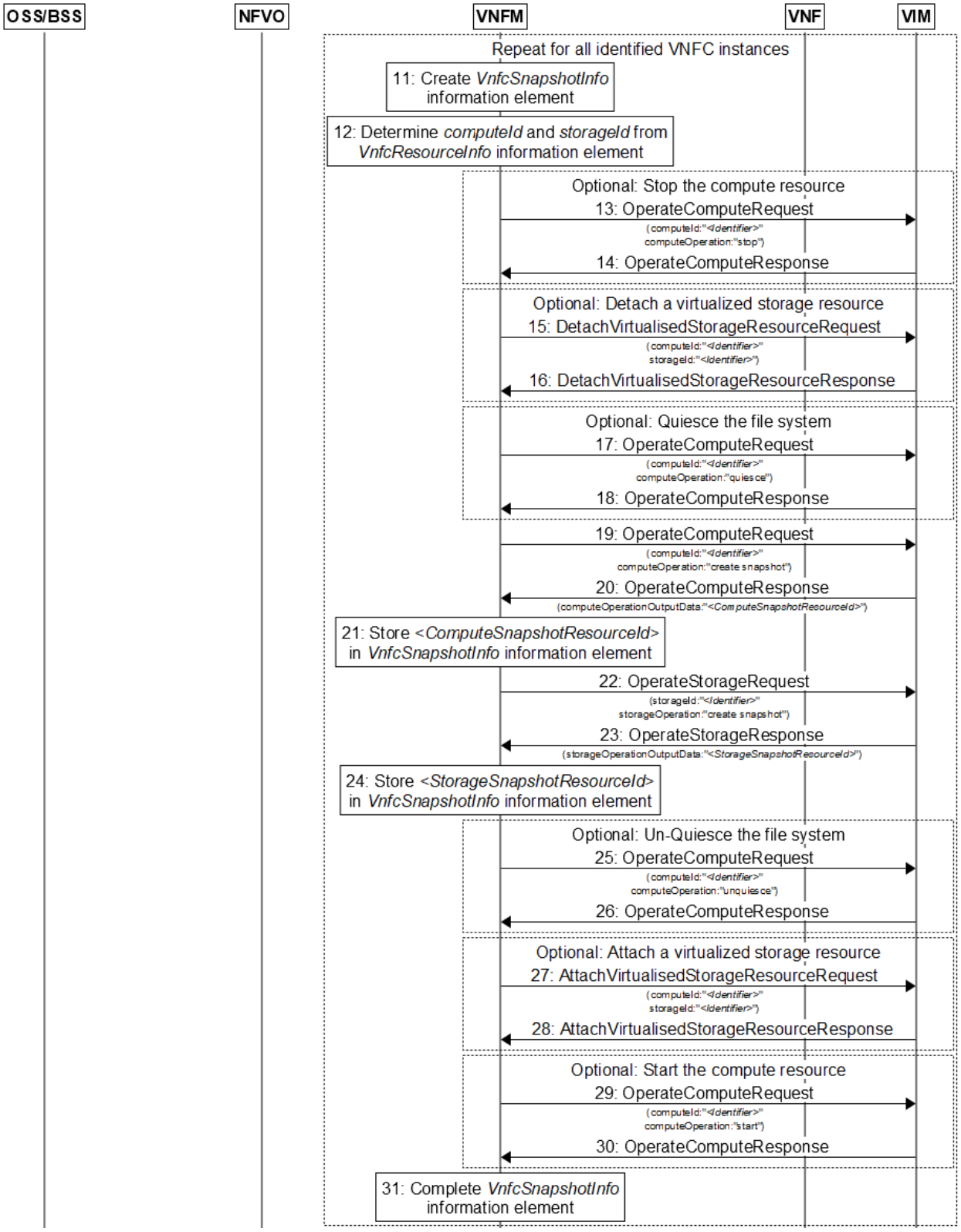

Figure F.3.2-2: Create VNF Snapshot triggered from OSS/BSS, part 2

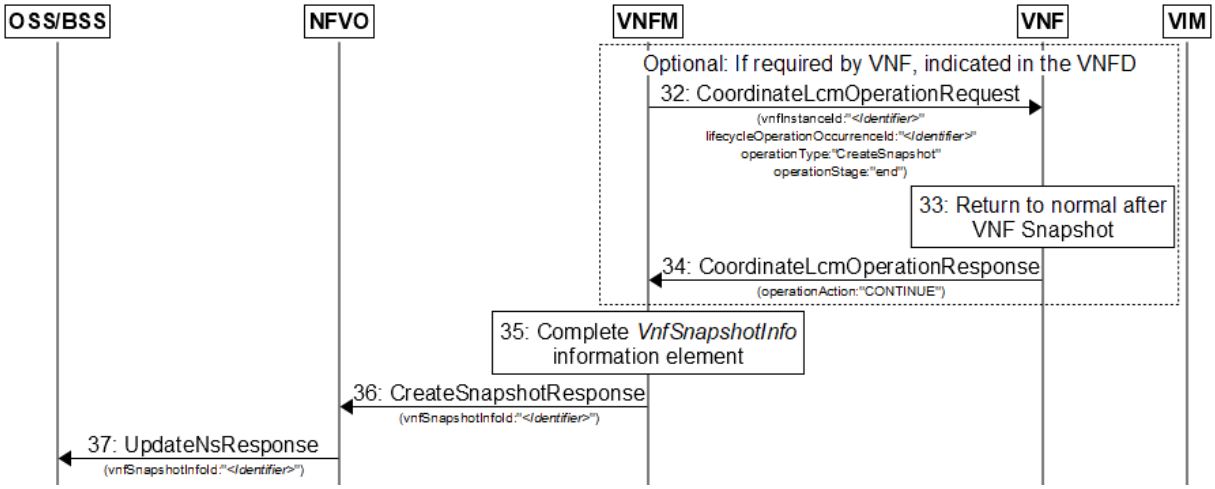

Figure F.3.2-3: Create VNF Snapshot triggered from OSS/BSS, part 3

F.3.3 Query VNF Snapshot information procedure

The procedure to query for VNF Snapshot information comprises of the following steps as depicted in Figure F.3.3-1:

Step 1: The OSS/BSS initiates the query for VNF Snapshot information and sends a request to query an NS to the NFVO, including a query filter containing the identifier of the VNF instance for which VNF Snapshot information is searched.

Step 2: The NFVO determines the responsible VNFM for the VNF instance for which VNF Snapshot information is searched, utilizing the indicated identifier.

Step 3: The NFVO sends a request to query for VNF Snapshot information to the VNFM, including a query filter containing the identifier of the VNF instance for which VNF Snapshot information is searched.

Step 4: The VNFM searches through the VnfSnapshotInfo objects it maintains and determines all available VnfSnapshotInfo objects' information elements containing a reference to the indicated VNF instance.

Step 5: The VNFM completes the query for VNF Snapshot information operation and sends the response to the NFVO, including all determined VnfSnapshotInfo objects' information elements.

Step 6: The NFVO sends the response to the originating request to the OSS/BSS, including all determined VnfSnapshotInfo objects' information elements.

NOTE: This procedure covers the use case that the OSS/BSS queries for information on all available VNF Snapshots that exist for a certain VNF instance. The procedure is similar for other use cases, it only differs on the used filter information used in the query requests and the corresponding matching results returned in the query responses. Another use case example could be to query for the available information on a specific VNF Snapshot, in this case the query filter would contain an identifier of the respective VnfSnapshotInfo object.

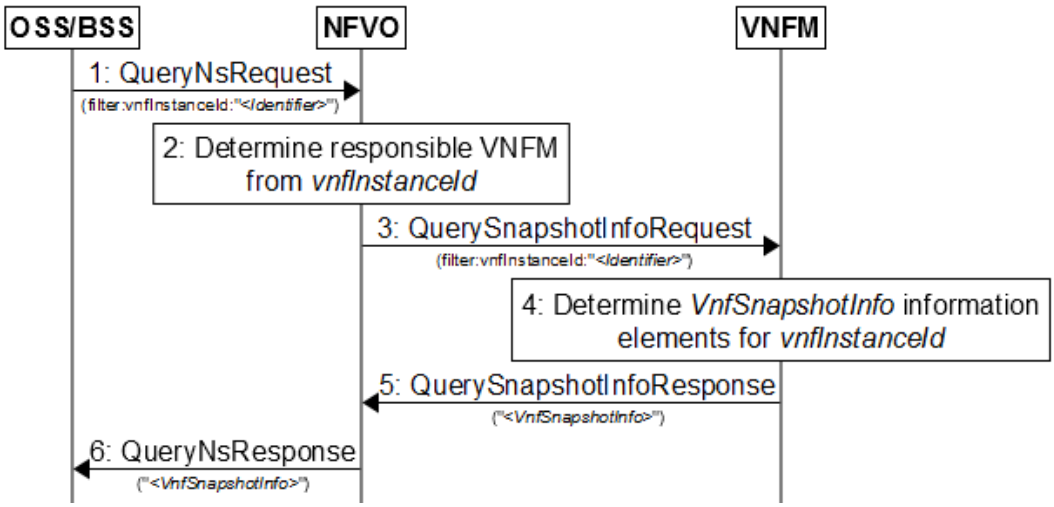

Figure F.3.3-1: Query VNF Snapshot information triggered from OSS/BSS

F.3.4 Revert-To VNF Snapshot procedure

The procedure to revert to a VNF Snapshot comprises of the following steps as depicted in Figures F.3.4-1 to F.3.4-3:

Step 1: The OSS/BSS initiates the RevertToSnapshot operation and sends a request to update an NS to the NFVO, including the update type, the identifier of the VNF Snapshot information object and the identifier of the VNF instance to be reverted.

Step 2: The NFVO determines the responsible VNFM for the VNF instance to be snapshotted, utilizing the indicated VNF instance identifier.

Step 3: The NFVO validates the policies for the RevertToSnapshot operation for the indicated VNF instance. Policy rules could be provided by the VNFD of the VNF instance, for example allowing or denying reverting Snapshots.

Step 4: The NFVO sends a request to revert to a VNF Snapshot to the VNFM, including the identifier of the VNF Snapshot information object and the identifier of the VNF instance to be reverted.

Step 5: In case the VNFD indicates that the VNF requires the preparation for VNF Snapshot reversion, the VNFM sends a CoordinateLcmOperationRequest message to the VNF, indicating the start of a RevertToSnapshot lifecycle operation. The VNFM pauses its execution of the RevertToSnapshot operation and waits for a confirmation from the VNF.

Step 6: The VNF performs internal pre-snapshot procedures to prepare for reverting to a VNF Snapshot.

Step 7: The VNF sends a CoordinateLcmOperationResponse message to the VNFM to indicate the successful completion of the preparation for the reversion to a VNF Snapshot.

Step 8: The VNFM resumes the execution of the RevertToSnapshot operation and determines the parameters for the operation from the VNFD of the VNF instance. Those parameters indicate for example if the virtualised compute resource needs to be stopped prior to the reversion, or if a virtualised storage resource needs to be detached.

Step 9: The VNFM determines the VNFC instances to be reverted.

The steps 10 to 13/14 are repeated for all identified VNFC instances to be reverted:

Step 10: The VNFM determines the identifiers of the virtualised compute and virtualised storage resources of the VNFC instance to be reverted from the VnfcResourceInfo.

Steps 11/12: If the parameters for the RevertToSnapshot operation indicate that the virtualised compute resource of the VNFC instance needs to be stopped before reversion, the VNFM sends a request to the VIM to stop the indicated virtualised compute resource. The VIM sends a corresponding response after completion of the operation.

Steps 13/14: If the parameters for the RevertToSnapshot operation indicate that a virtualised storage resource needs to be detached from the virtualised compute resource of the VNFC instance before reversion, the VNFM sends a request to the VIM to detach the indicated virtualised storage resource from the indicated virtualised compute resource. The VIM sends a corresponding response after completion of the operation.

Step 15: The VNFM determines the identifiers of the virtualised compute and storage resource snapshots from the VnfcSnapshotInfo.

Step 16: The VNFM sends a request to the VIM to revert the indicated virtualised compute resource to the indicated virtualised compute resource snapshot.

Step 17: The VIM returns a response to the VNFM upon completion of the reversion of the indicated virtualised compute resource, including an indication of the result of the operation.

Step 18: The VNFM sends a request to the VIM to revert the indicated virtualised storage resource to the indicated virtualised storage resource snapshot.

Step 19: The VIM returns a response to the VNFM upon completion of the reversion of the indicated virtualised storage resource, including an indication of the result of the operation.

Steps 20/21: If a virtualised storage resource has been detached from the virtualised compute resource of the VNFC instance before reversion, the VNFM sends a request to the VIM to attach the indicated virtualised storage resource back to the indicated virtualised compute resource. The VIM sends a corresponding response after completion of the operation.

Steps 22/23: If the virtualised compute resource of the VNFC instance has been stopped before reversion and if the parameters for the RevertToSnapshot operation indicate that it needs to be started after reversion, the VNFM sends a request to the VIM to start the indicated virtualised compute resource. The VIM sends a corresponding response after completion of the operation.

Step 26: In case the VNFD indicates that the VNF requires a return to normal after VNF Snapshot reversion, the VNFM sends a CoordinateLcmOperationRequest message to the VNF, indicating the end of a RevertToSnapshot lifecycle operation. The VNFM pauses its execution of the RevertToSnapshot operation and waits for a confirmation from the VNF.

Step 25: The VNF performs internal post-snapshot procedures to return to normal after reverting to a VNF Snapshot.

Step 26: The VNF sends a confirmation message to the VNFM to indicate the successful completion of the return to normal after the reversion to a VNF Snapshot.

Step 27: The VNFM resumes the execution of and completes the RevertToSnapshot operation and sends the response to the NFVO.

Step 28: The NFVO sends the response to the originating request to the OSS/BSS.

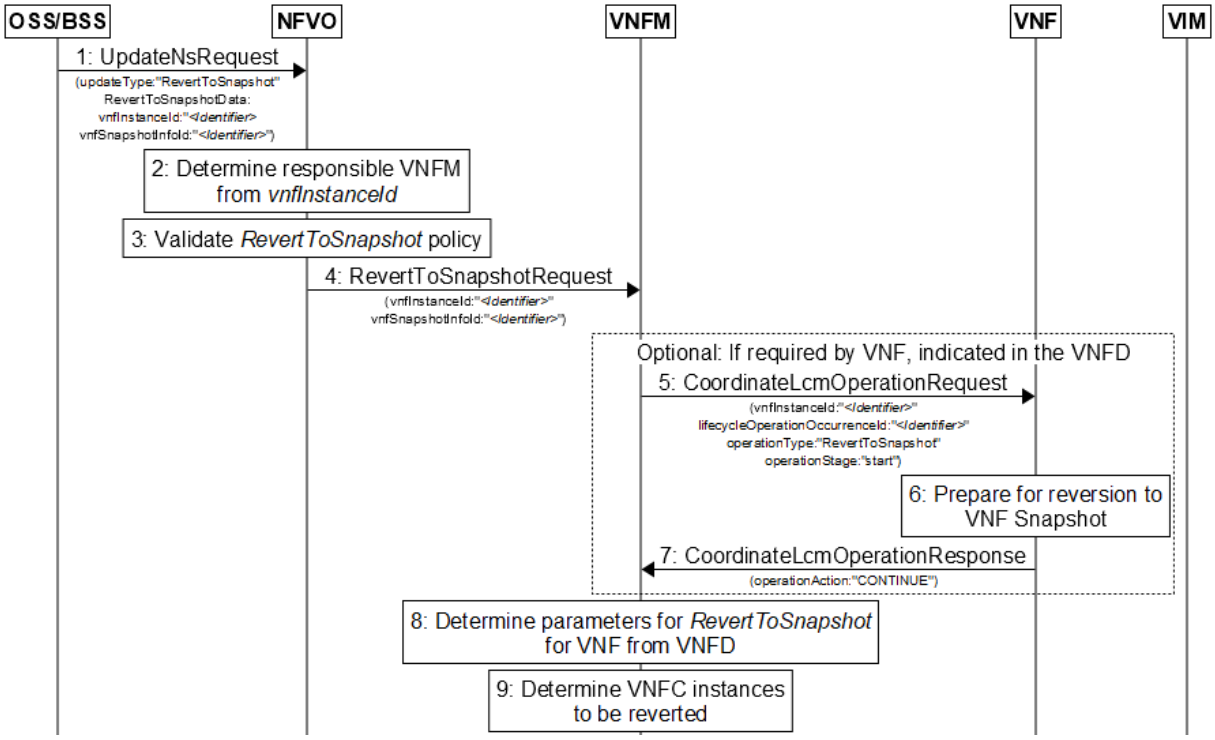

Figure F.3.4-1: Revert-To VNF Snapshot triggered from OSS/BSS, part 1

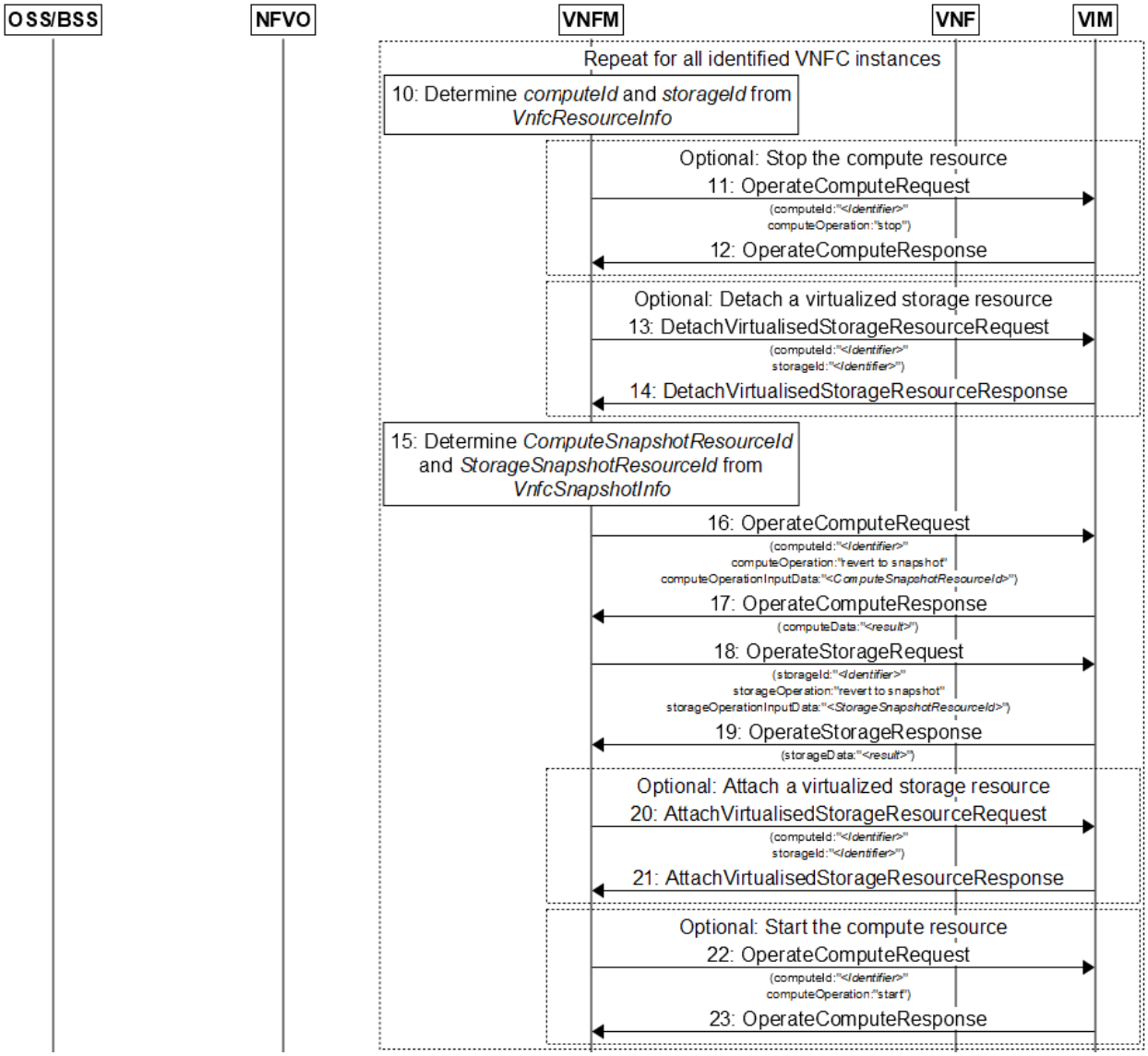

Figure F.3.4-2: Revert-To VNF Snapshot triggered from OSS/BSS, part 2

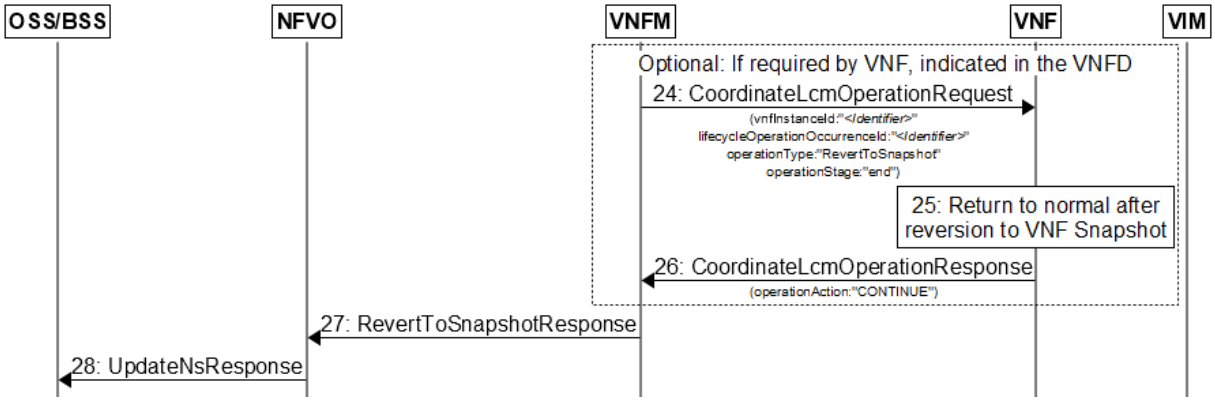

Figure F.3.4-3: Revert-To VNF Snapshot triggered from OSS/BSS, part 3

F.3.5 Delete VNF Snapshot information procedure

The procedure to delete VNF Snapshot information comprises of the following steps as depicted in Figure F.3.5-1:

Step 1: The OSS/BSS initiates the DeleteSnapshotInfo operation and sends a request to update an NS to the NFVO, including the update type, the identifier of the VNF Snapshot information object to be deleted and the identifier of the corresponding VNF instance.

Step 2: The NFVO determines the responsible VNFM maintaining the VNF Snapshot information of the VNF instance, utilizing the indicated VNF instance identifier.

Step 3: The NFVO sends a request to delete the VNF Snapshot information to the VNFM, including the identifier of the VNF Snapshot information object to be deleted.

Step 4: The VNFM determines the identifiers of the VnfcSnapshotInfo objects to be deleted from the VnfSnapshotInfo object.

The steps 5 to 12 are repeated for all identified VNFC Snapshot information objects to be deleted:

Step 5: The VNFM determines the identifier of the VnfcInfo object from the VnfcSnapshotInfo.

Step 6: The VNFM determines the identifiers of the virtualised compute and virtualised storage resources of the VNFC instance from the VnfcResourceInfo.

Step 7: The VNFM determines the identifiers of the virtualised compute and storage resource snapshots from the VnfcSnapshotInfo.

Step 8: The VNFM sends a request to the VIM to delete the indicated virtualised compute resource snapshot for the indicated virtualised compute resource.

Step 9: The VIM returns a response to the VNFM upon completion of the deletion of the indicated virtualised compute resource snapshot, including an indication of the result of the operation.

Step 10: The VNFM sends a request to the VIM to delete the indicated virtualised storage resource snapshot for the indicated virtualised storage resource.

Step 11: The VIM returns a response to the VNFM upon completion of the deletion of the indicated virtualised storage resource snapshot, including an indication of the result of the operation.

Step 12: The VNFM deletes the VnfcSnapshotInfo object.

Step 13: The VNFM deletes the VnfSnapshotInfo object.

Step 14: The VNFM completes the DeleteSnapshotInfo operation and sends the response to the NFVO.

Step 15: The NFVO sends the response to the originating request to the OSS/BSS.

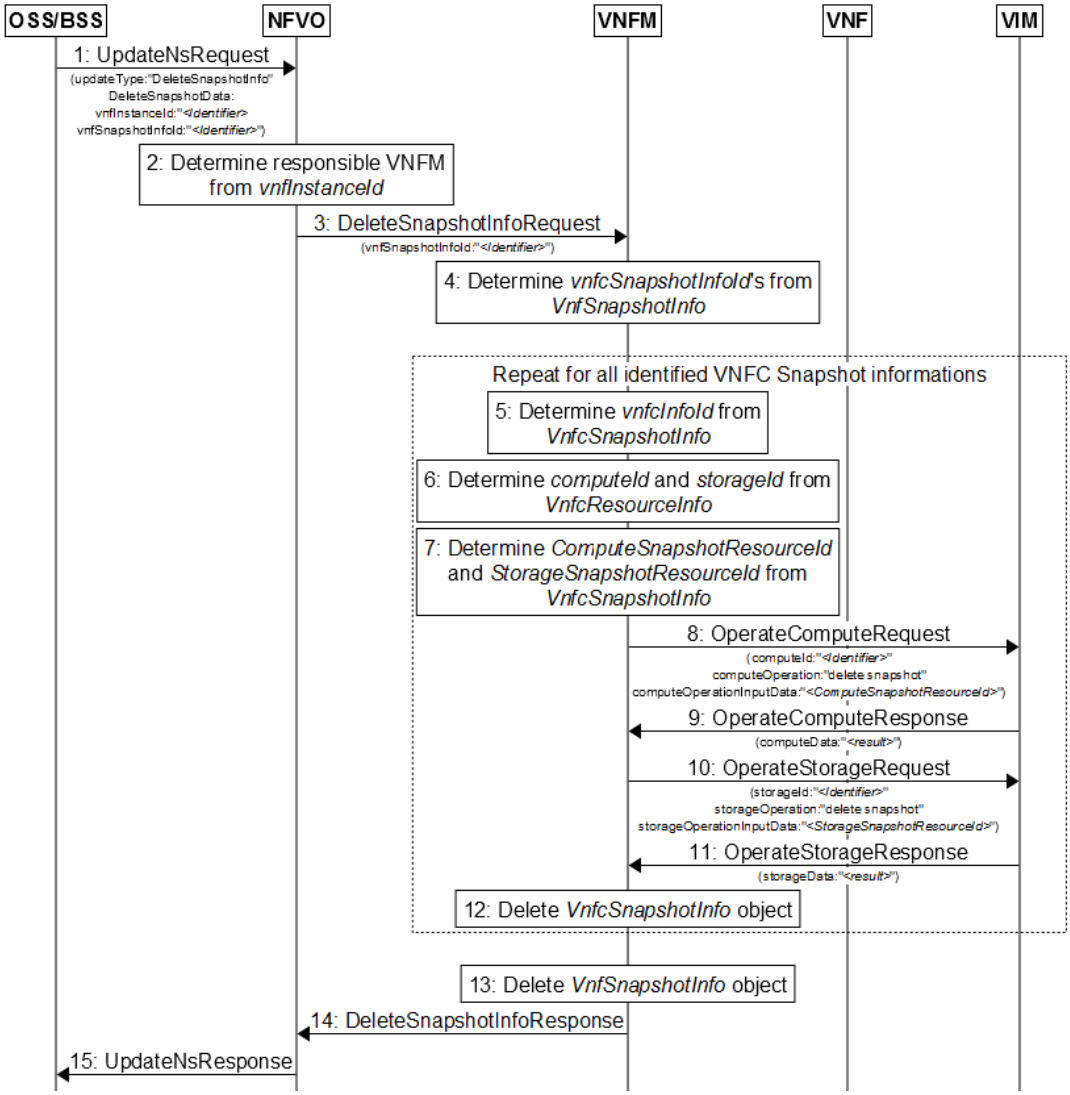

Figure F.3.5-1: Delete VNF Snapshot information triggered from OSS/BSS

Annex G (informative): NFV-MANO and integration of management and connectivity for Multi-Site services

G.1 Introduction

The present annex introduces the architecture options for the placement of WAN infrastructure management functional entity with respect to the NFV-MANO architecture for supporting multi-site network services.

There are two architecture options:

a) Architecture option #A: WIM integration into NFV-MANO framework as specialized VIM:

- In this option, the WIM is introduced as a specialized VIM. The WIM exposes the interfaces with Network Controllers of WAN infrastructure and is responsible for controlling and managing network connectivity of WAN between endpoints in different NFVI-PoPs.

b) Architecture option #B: WIM integration as external entity to the NFV-MANO framework:

- In this option, the WIM functionality is external to the NFV-MANO framework and integrated or controlled by other OSS/BSS. In this model WAN resources are envisioned not to be reconfigured regularly, e.g. for static provisioning, or when such WAN resources are pre-provisioned. The WIM functionality is out of scope of NFV-MANO but the interactions over the Os-Ma-nfvo reference point need to be considered.

NOTE: In this option, the NFVO may be allowed to trigger resources of WAN via the OSS/BSS, but such a case is regarded to be similar to option #A, with the difference that WIM functionality is not interfaced directly by the NFVO.

The option #B can be suitable in cases where early NFV deployments cannot make use of network controllers with programmatic/open interfaces for all network segments.

G.2 Architecture options

G.2.1 Architecture option #A: WIM integration into NFV-MANO framework as specialized VIM

In this option, Or-Wi reference point between NFVO and WIM is as a subset of Or-Vi and it is specified to manage WAN connectivity as shown in Figure G.2.1-1. The WIM function block is responsible for the management of virtualised network resources of WAN to support the deployment of network services that extend across multiple NFVI-PoPs.

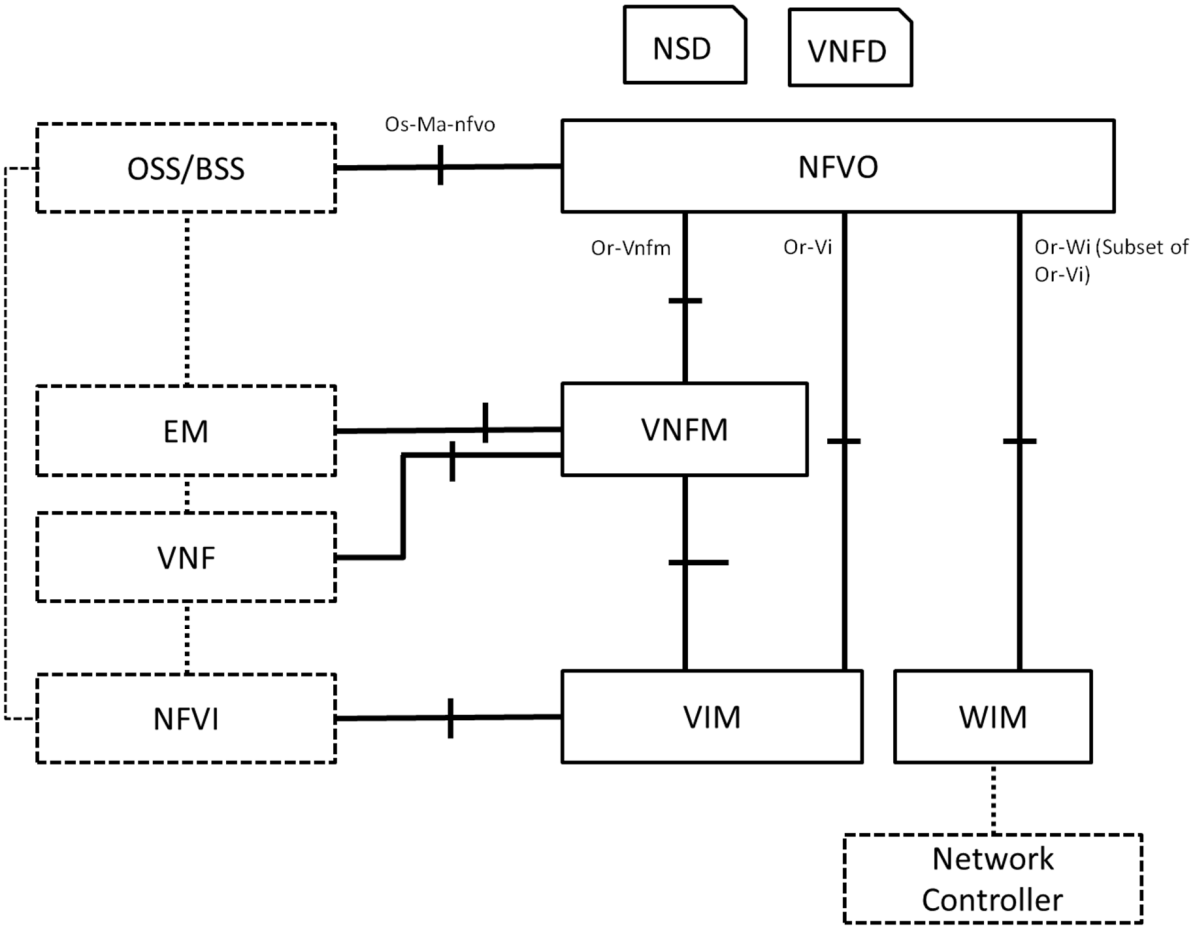

Figure G.2.1-1: Managing WIM function blocks using Or-Wi reference point

G.2.2 Architecture option #B: WIM integration as external entity to the NFV-MANO framework managing WIM functionality of OSS/BSS with Os-Ma-nfvo reference points

In this option, the Os-Ma-nfvo reference point supports the required WAN management functions, as shown in Figure G.2.2-1. The NFV-MANO does not have the responsibility for the management of the virtualised network resources inside the WAN. If WAN connectivity is pre-provisioned, the NFVO can be provided information about the relevant connectivity that spans across the multiple NFVI-PoPs.

NOTE: In this option, the NFVO can request the management of virtualised network resource of WAN for NFV services with OSS/BSS that span across multiple NFVI-PoPs, but such a case is regarded to be similar to option #A, with the difference that WIM functionality is not interfaced directly by the NFVO.

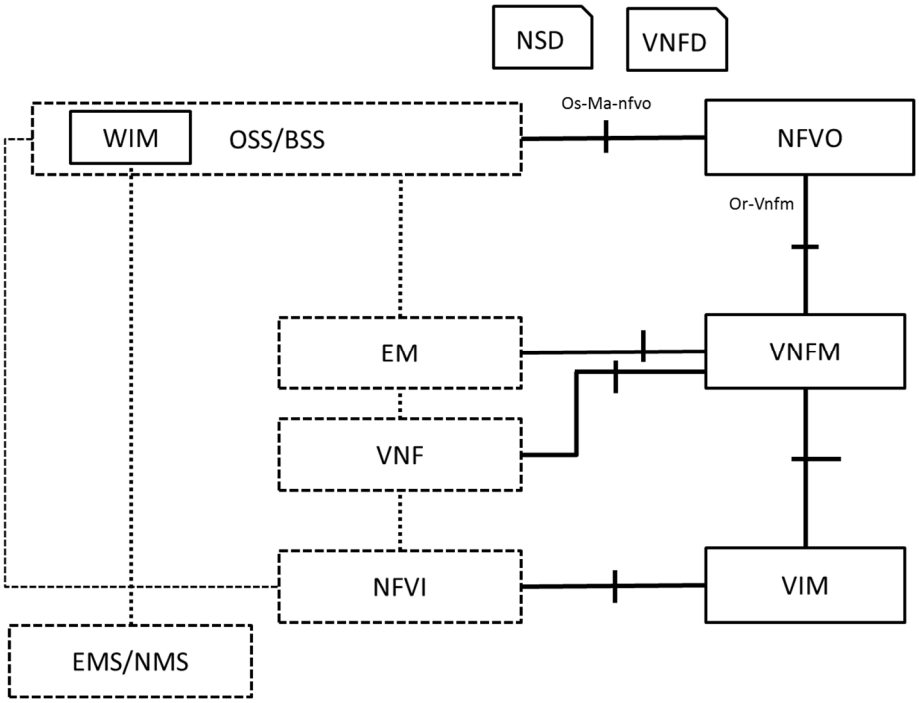

Figure G.2.2-1: Managing WIM function blocks using Os-Ma-nfvo reference point

Annex H (informative): NFVI operation and maintenance

H.1 Procedures related to NFVI operation and maintenance

H.1.1 Introduction

This clause describes end-to-end examples of policy transfer and coordination procedures enabling the mitigation of NFVI operation and maintenance impacts.

The operation and maintenance performed in an NFVI can impact all or several virtualised resources used by a VNF hosted by this NFVI. If these virtualised resource impacts occur simultaneously, they could have negative effect on the availability, reliability and continuity of the services provided by that VNF. To mitigate such impacts, it is desirable to install policies in the VIM that put constraints on the virtualised resources and their groups used by the VNF that would regulate the NFVI operation and maintenance procedures in such a way, which ensures that the virtualised resources are taken away at a pace tolerable by the VNF and the VNF is given enough time to prepare for each such upcoming impact.

The constraints applicable at NFVI operation and maintenance are those specifically defined for such operations in clause 5.6.4.2. These constraints can originate from different sources and at different times. Namely:

- The VNF provider can provide some or all the NFVI operation and maintenance constraints in the VNFD together with the information on the virtualised resources needed for the VNF and their affinity/anti-affinity rules. This information can be used by the VNFM to provide the NFVI operation and maintenance constraints in an appropriate form to the VIM e.g. at the time of the allocation of the virtualised resources or at the time of the creation of the affinity/anti-affinity groups. Subsequently, the VNFM can use the VNFD information also to update the constraints as necessary. For example, after scaling the VNF, the VNFM can set a new constraint value appropriate for the new group size of an anti-affinity group.

- The EM on behalf of the VNF can provide NFVI operation and maintenance constraints at runtime, or update existing ones based on the current status of the VNF and the traffic it handles. For example, if the VNF is involved in a long-running operation (e.g. healing) that should not be interrupted (based on, e.g. service provider policies), it can require a longer impactNotificationLeadTime for any upcoming NFVI operation and maintenance.

To guarantee the consistency of the NFVI operation and maintenance constraints applicable to groups of virtualised resources, they need to be aligned with the current placement and the placement constraints of the virtualised resources.

The placement constraints of virtualised resources are negotiated based on the affinity/anti-affinity rules of the VNFD, the availability of resources, and any applicable additional policies (e.g. fallbackBestEffort) between the NFVO and the VNFM as part of the virtualised resource granting procedure.

Once the VNF has been deployed, there might be inconsistencies between the NFVI operation and maintenance constraints and the current placement and/or placement constraints of the virtualised resources they apply to.

To handle such inconsistencies, the constraints are converted into policies and propagated using the policy management interface of the appropriate reference point. The NFV-MANO component obtaining NFVI operation and maintenance constraints from the VNFD or receiving NFVI operation and maintenance policies with constraints from the EM or another NFV-MANO component needs to ensure the consistency of the constraints and policies they are aware of and accordingly accept or reject any newly received policies.

Once the consistent NFVI operation and maintenance constraints have been delivered to the VIM in the form of policies, the VIM applies them to the NFVI operation and maintenance as appropriate.

Whenever the NFVI operation and maintenance is initiated, the VIM provides notifications about upcoming impacts in advance as requested by the hosted VNFs allowing the VNFs to prepare for the impacts. The VNFs can explicitly confirm their readiness for the impact or the VIM can proceed with the NFVI maintenance and operations after the applicable time has expired.

H.1.2 VNFD-based transfer of NFVI operation and maintenance policies

In this scenario shown in Figure H.1.2-1, the NFVI operation and maintenance constraints are specified in the VNFD as nfviMaintenanceInfo of the virtualised resources and their groups of a VNF. For a new instance of the VNF these are all the applicable placement constraints. Therefore, based on the constraints in the VNFD the VNFM creates the appropriate NFVI operation and maintenance policies and installs these policies as part of the VNF instantiation procedures.

The scenario comprises of the following steps:

Step 1: The NFVO requests the VNFM to instantiate a VNF. The receiving VNFM performs all the procedures necessary before deployment (e.g. collecting the information on the needed resources and their groups with their nfviMaintenanceInfo from the VNFD, obtaining the grants from the NFVO, etc.).

Steps 2, 3 and the loop 4-7 can be executed in different orders as appropriate for the interacting VNFM and the VIM.

Step 2: The VNFM converts the constraints of the nfviMaintenanceInfo (see clause 7.1.8.17 in ETSI GS NFV-IFA 011 [i.19]) into NFVI operation and maintenance policies and transfers them to the VIM.

Step 3: The VNFM requests the VIM to create an anti-affinity group (AAG), named AvailabilityG. The request may reference applicable policies installed in step 2.

In a loop for each i^{th} ^{VNF}C instance ($VNFC_{i)\ of}$ the AvailabilityG AAG steps 4-7 are executed:

Step 4: The VNFM initiates the procedure to instantiate $VNFC_{i.}$

Step 5: The VNFM requests the VIM to allocate a virtual compute resource, named $VR_{i,}$ as part of the AvailabilityG AAG. The request may reference applicable policies installed in step 2.

Step 6: The VIM allocates VR_i on host $H_{i.}$

Step 7: The VIM responds to the VNFM confirming the allocation of VR_i on host $H_{i.}$

Upon completion of the above steps, the procedure continues as follows:

Step 8: The VNFM subscribes with the VIM for virtualised resource change notifications for the individual virtualised resources (VR_i) and for the AvailabilityG AAG.

Step 9: The VNF (or on its behalf the EM) subscribes with the VNFM for alarms at the severity level of warning. These report upcoming changes in the virtualised resources of the VNF.

At step 9 all the resources necessary for the instantiation of the VNF have been created. In case of an upcoming impact due to some NFVI operation and maintenance the VNF (or on its behalf its EM) will receive an alarm of severity warning.

In addition, at runtime the NFVI maintenance policies may be queried and updated as needed. For example, if new virtual compute resources were allocated according to steps 4-7 and they were added to the AvailabilityG AAG the maximum number of impacted resources constraint for AAG may need to be changed.

If the VNFM wants to retrieve the current set of active policies in the VIM for the resources used by a VNF (e.g. maximum number of impacted resources for the AvailabilityG), the following step is executed:

Step 10: The VNFM queries the VIM of the currently installed NFVI operation and maintenance policies for the resources used by the VNF.

If the VNFM needs to transfer an updated set of NFVI operation and maintenance policies to the VIM (e.g. the new maximum number of impacted resources for the AvailabilityG), the following step is executed:

Step 11: The VNFM transfers to the VIM the updated NFVI operation and maintenance policies.

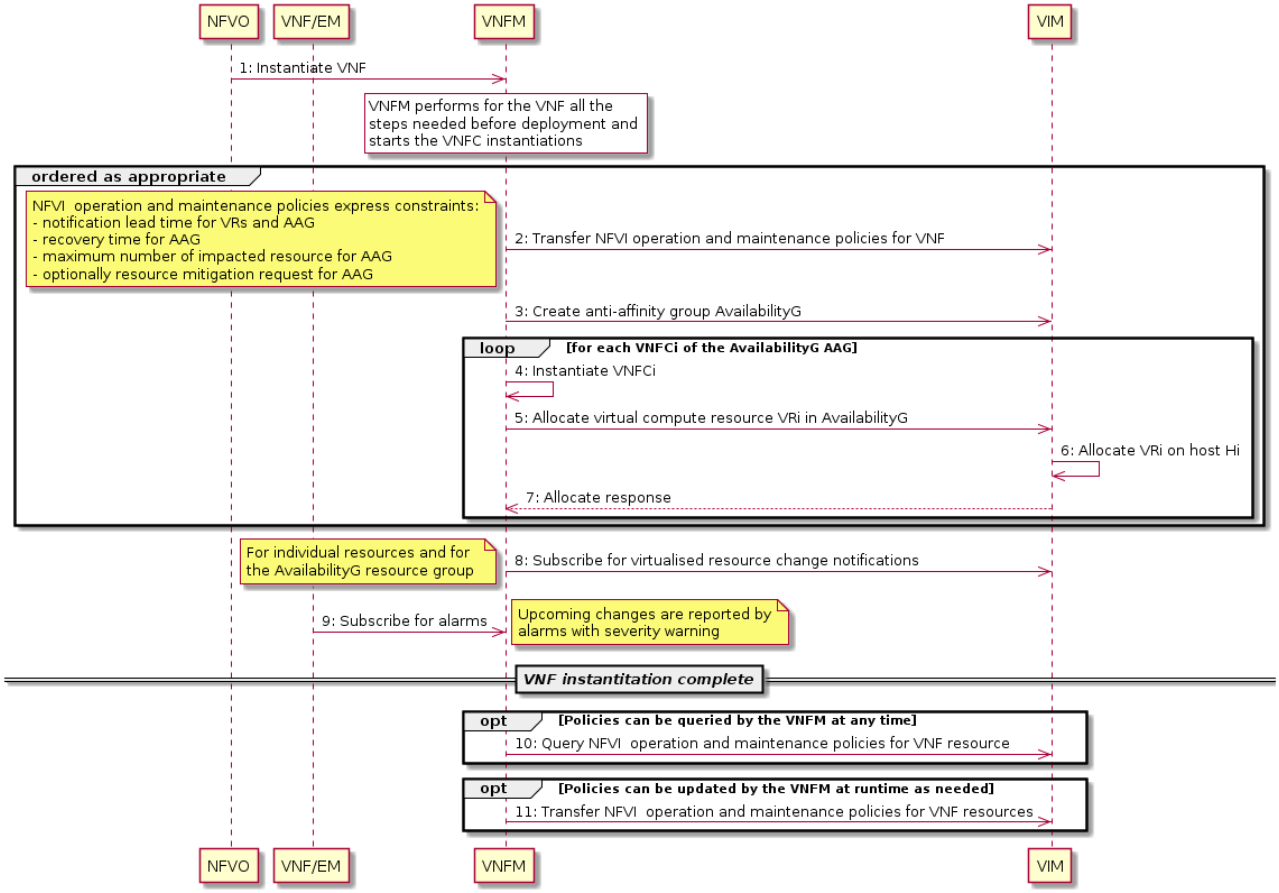

Figure H.1.2-1: Installing policies for constraints provided in the VNFD

H.1.3 NFVI operation and maintenance coordination for group impact

In the scenario shown in Figure H.1.3-1, the NFVI operation and maintenance constraints have been specified for an anti-affinity group of a VNF instance, for example, according to the scenario of clause H.1.2. In particular, it was requested to send virtualised resource change notifications in advance of an impact on the AvailabilityG anti-affinity group (AAG) with a specified lead time.

EXAMPLE: The reason for this could be to have enough time for the VNF instance to redirect the traffic it is handling to a geo-redundant pair. That is, the VNF is deployed as a geo-redundant pair of VNF instances, whereby one VNF instance is the VNF instance considered in this scenario. It is worth mentioning that no switch-over needs to be performed by the VNF pair if the upcoming impact is only for a single virtualised resource (or less than the maxNumberOfImpactedInstances).

The scenario comprises of the following steps:

Step 1: The VIM (or the NFVI Software Modification Module) determines that it is going to upgrade a set of resources, which host AAGs. Among these AAGs is the AvailabilityG AAG, whose virtualised resources are notified by virtualised resource change notifications in advance with a lead time.

Step 2: The VIM sends to the VNFM a virtualised resource change start notification indicating that the virtualised resources of the AvailabilityG AAG will be impacted.

Step 3: The VIM starts also a timer with the lead time specified for the AvailabilityG together with the notification sent in step 2 as required by the policies installed and applicable to the AvailabilityG AAG.

Step 4: Based on the notification received in step 2, the VNFM sends an alarm at the warning level to the VNF (or its EM) about the upcoming impacts on the group of VNFC instances (i.e. for a given VNFC a.k.a VDU) due to NFVI operation and maintenance.

Step 5: The VNF performs action(s) to mitigate the upcoming impacts.

NOTE: In the above example, to mitigate the upcoming impacts the VNF performs a switch-over to the geo-redundant pair.

Step 6: At the expiration of the AvailabilityG AAG lead time, the VIM proceeds with the upgrade of the resources, which among others, host the virtualised resources in AvailabilityG AAG.

Next, in a loop the VIM performs the upgrade/maintenance of the resources hosting the AvailabilityG AAG as appropriate.

Step 7: The VIM sends to the VNFM a virtualised resource change end notification indicating that the impact on the virtualised resources of the AvailabilityG AAG has ended.

Step 8: The VNFM clears with the VNF (or its EM) the alarm about the group impact of VNFC instances.

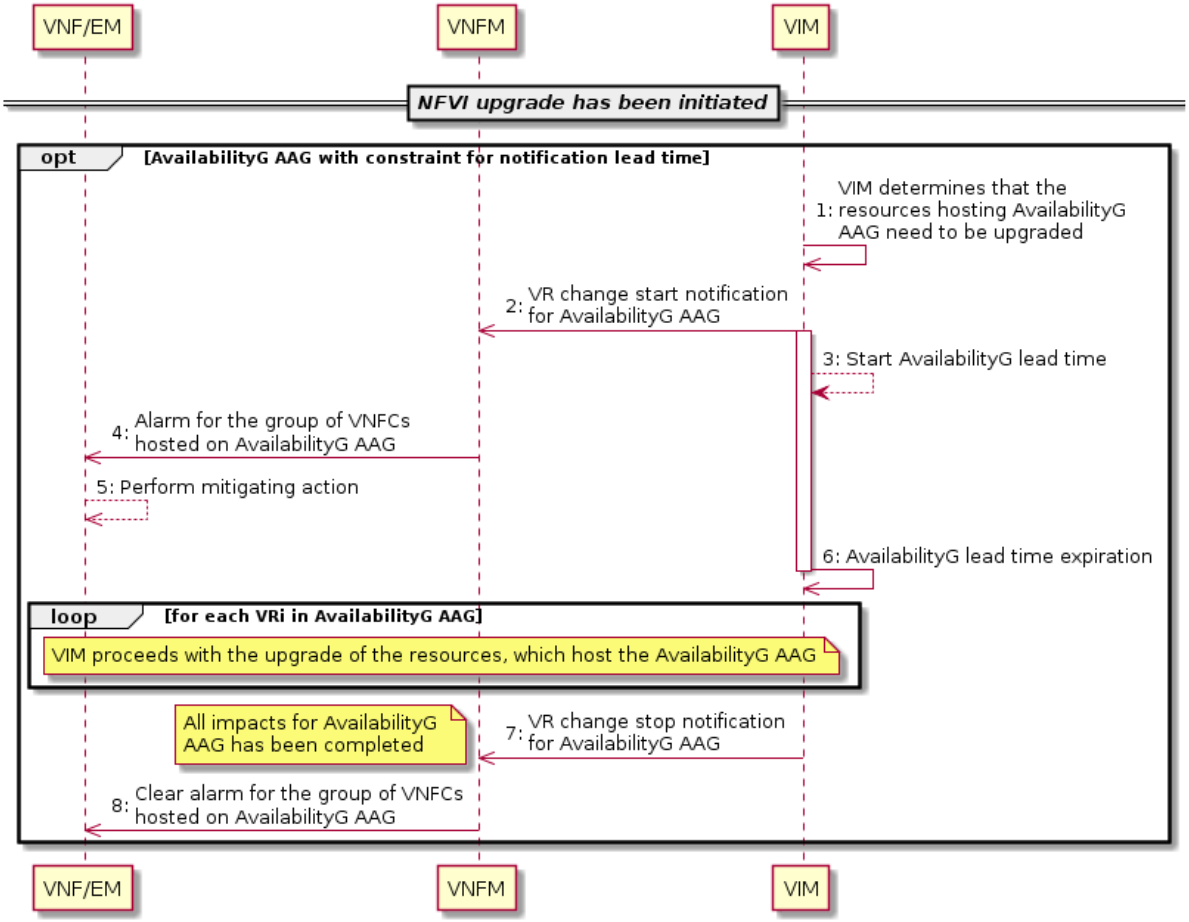

Figure H.1.3-1: NFVI operation and maintenance coordination for group impact

H.1.4 NFVI operation and maintenance coordination for virtualised resource impact

In the scenario shown in Figure H.1.4-1, the NFVI operation and maintenance constraints have been specified for a virtualised resource of a VNF before the initiation of the NFVI upgrade operation, for example, as a result the scenario discussed in clause H.1.2. In particular, it was requested to send virtualised resource change notifications in advance of an impact on the virtualised resource with a specified lead time and with resource mitigation.

The scenario comprises of the following steps:

Step 1: The VIM (or the NFVI Software Modification Module) determines that it is going to upgrade next host H_i, which hosts, among others, virtualised resource VRi of the VNF.

Step 2: The VIM sends to the VNFM a virtualised resource change notification indicating that virtualised resource VR_i will be impacted.

Step 3: Together with the notification sent in step 2, the VIM starts a timer with the lead time specified for VR_i.

Step 4: To mitigate the removal of VR_i used by $VNFC_i$ of the VNF, the VNFM initiates an auto-scaling out of the VNF to add $VNFC_x$.

Step 5: The VNFM requests the VIM to allocate VRx.

Step 6: The VIM allocates VRx on host Hx.

Step 7: The VIM confirms the allocation of VRx.

Step 8: Based on the notification received in step 2, the VNFM sends to the VNF (or its EM) an alarm with severity warning about the upcoming impact of $VNFC_i$ due to NFVI operation and maintenance and indicates that $VNFC_x$ is provided to mitigate the impact.

Step 9: The VNF redirects the traffic from $VNFC_i$ to $VNFC_x$.

Step 10: The lead time for VR_i expires indicating to the VIM that the maintenance of host H_i can proceed.

Step 11: The VIM removes VR_i from host H_i to prepare H_i for the maintenance.

Step 12: The VIM performs the maintenance of host H_i.

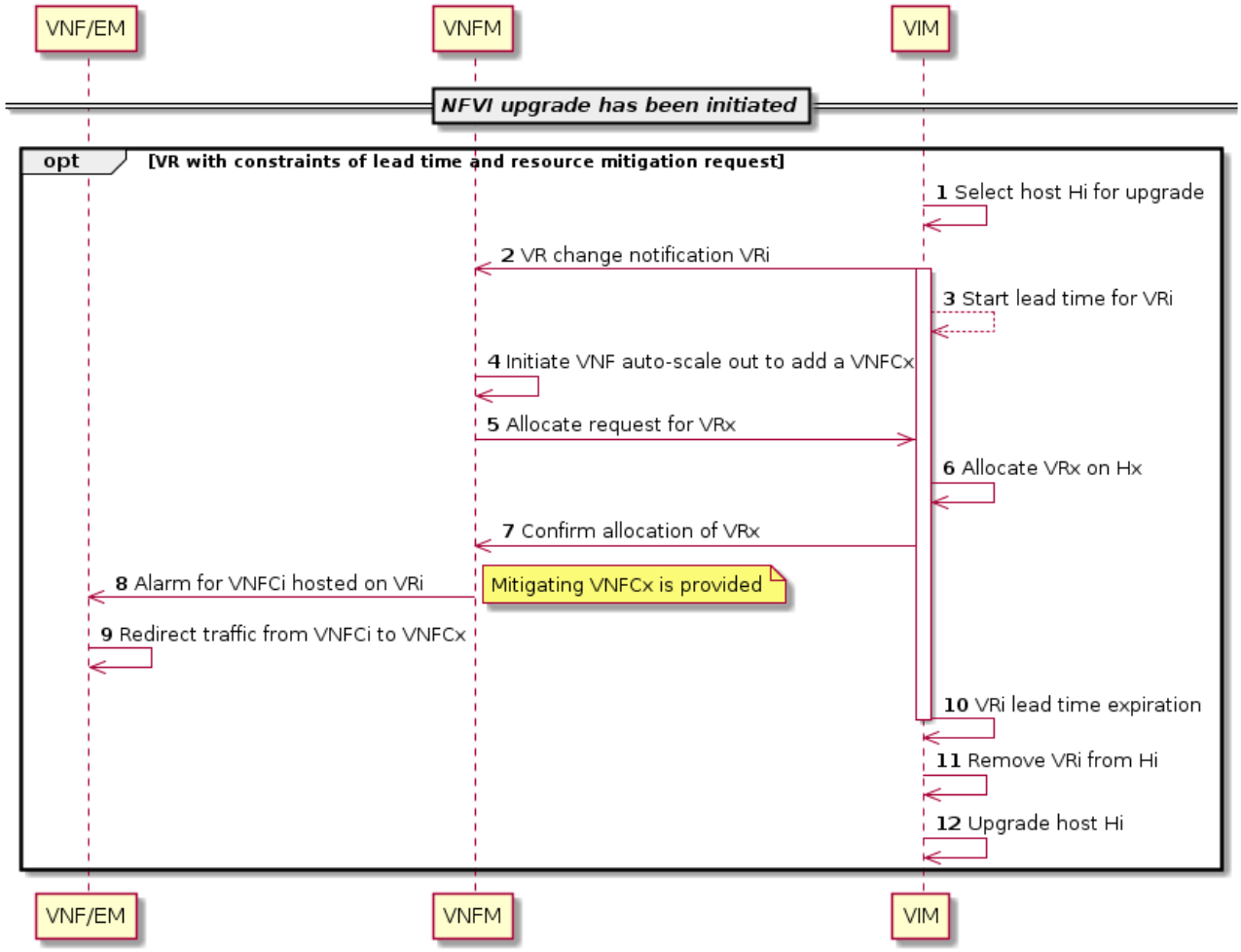

Figure H.1.4-1: NFVI operation and maintenance coordination for virtualised resource impact

Annex I (informative): Change History

Date	Version	Information about changes
2018-02-28	V2.4.2	Started 2018H1 maintenance CR NFVIFA(18)000092 Remove some abbreviation (approved during IFA#89 Sophia Antipolis) CR NFVIFA(18)000096r1 Remove definitions covered in NFV003 (approved during IFA#89 Sophia Antipolis)
2018-05-02	V2.4.3	History corrected CR NFVIFA(17)0001135r4 Make VnfPkgChangeNotification reception optional for VNFM (approved during IFA#89 Sophia Antipolis) CR NFVIFA(18)000195 Remove unused reference and an abbreviation (approved IFA#91)
2018-05-09	V3.0.0	Base Line for Release 3 created from draft v2.4.3 as agreed in IFA#98
2018-05-24	V3.0.1	Implements FEAT04 Compute Host Reservation Mega CR NFVIFA(18)000419r3 See also NFVIFA(18)000475 for a list of CRs to other specifications associated with FEAT04
2018-06-11	V3.0.2	Correct Title line, correct numbering in clause A.2.8 Implements FEAT07 Composite NS across multi domain Mega CR NFVIFA(18)000455r3 See also NFVIFA(18)000492r1 for a list of CRs to other specifications associated with FEAT07 Implements FEAT08 NS across multiple administrative domains Mega CR NFVIFA(18)000425r1 See also NFVIFA(18)000515r1 for a list of CRs to other specifications associated with FEAT08
2018-06-15	V3.0.3	Implements FEAT11 NFV-MANO management NFVIFA(18)000576 Implements FEAT15 VNF Snapshot NFVIFA(18)000539 See also NFVIFA(18)000577r1 for a list of CRs to other specifications associated with FEAT15
2018-06-27	V3.0.4	Correct wording in history Implements CR NFVIFA(18)000661 Clarifications on Compute Host Reservation
2018-09-21	V3.1.2	Base Line for Release 3 Drop 2 created from published version 3.1.1 (not considering changes by EditHelp between v2.4.3 and v2.5.1, since v3.1.1 was created by editHelp) Implements CR NFVIFA(18)000760r1 IFA010ed321 Enhance policy management requirements related to multi-domain NS provisioning Implements CR NFVIFA(18)000798r2 IFA010ed321 Add policy management requirements related to support the capability for consuming operations in NFVO Implements CR NFVIFA(18)000799r2 IFA010ed321 Add policy management requirements related to support the capability for consuming operations in VNFM Implements CR NFVIFA(18)000800r2 IFA010ed321 Add policy management requirements related to support the capability for consuming operations in VIM Implements CR NFVIFA(18)000802r3 IFA010ed321 Add functional requirements for software image management
2018-11-09	V3.1.3	Implements following CRs: NFVIFA(18)000879r1 Restructuring annex of IFA010 (FEAT05 proposal) NFVIFA(18)000861 IFA010 - New annex for VNF Snapshots NFVIFA(18)000880 IFA010 - New annex Create VNF Snapshot procedure NFVIFA(18)000906r1 IFA010 - New annex Query VNF Snapshot information procedure NFVIFA(18)000907r1 IFA010 - New annex Revert-To VNF Snapshot procedure NFVIFA(18)000908r1 IFA010 - New annex Delete VNF Snapshot information procedure
2019-01-04	V3.1.4	Implements the following CRs: NFVIFA(18)0001012 - IFA010ed321 CR add policy associate disassociate operations NFVIFA(18)0001107r2 - IFA010 MegaCR FEAT010 General and functional requirements for Multi-Site Service
2019-01-18	V3.1.5	Implements the following CRs: NFVIFA(19)000018 IFA010ed321 - Annex F.3.2 Create VNF Snapshot procedure NFVIFA(19)000019 IFA010ed321 - Annex F.3.4 Revert-to VNF Snapshot procedure NFVIFA(18)000844r7 IFA010 MegaCR FEAT05 Slicing NFVIFA(19)000084 - IFA010 harmonize use of Assign and Allocate NFVIFA(19)000085 - IFA010 move informative reference
2019-02-06	V3.1.6	Implements the following CRs: NFVIFA(19)000060 FEAT02 IFA010 MegaCR
2019-02-23	V3.1.7	Implements the following CR: NFVIFA(19)000169r1 IFA010ed321 requirements for PNFD archive support

Date	Version	Information about changes
2019-05-10	V3.2.2	Base Line for Release 3 Drop 3 created from published version 3.2.1
2019-06-27	V3.2.3	Implements the following CR: NFVIFA(19)000074r2 IFA010ed321 Update of PM requirements for aligning with IFA027 Rapporteur's action: reference to IFA027 should be informative. NFVIFA(19)000421 IFA010ed331 7.2.5 Terminology correction NFVIFA(19)000488r1 IFA010 - Support for pods NFVIFA(19)000510r5 IFA010 General requirements for the software modification of NFV-MANO functional entities NFVIFA(19)000529 IFA010ed331 Change external VNF connectivity requirements
2019-07-16	V3.2.4	NFVIFA(19)000555 IFA010ed331 General requirements for change current Vnf Package NFVIFA(19)000556 IFA010ed331 5.X General requirements for software modification - bugfix NFVIFA(19)000038r7 IFA010 MegaCR FEAT03 NFVI MOD NFVIFA(19)000705 IFA010ed331_Update of PM requirements for aligning with IFA027-Rel-3 NFVIFA(19)000480r8 IFA010 MegaCR FEAT16 SAL Editorial corrections
2019-10-02	V3.3.2	Initial version for maintenance
2019-12-10	V3.3.3	NFVIFA(19)000891 IFA010-Remove Annex -Authors and contributors NFVIFA(19)000978r1 IFA010ed341 Snapshot Package Management interface requirements
2020-04-06	V3.3.4	NFVIFA(20)000234 IFA010ed341 FEAT15 Moving VNF snapshot package API Rapporteur-action to consistently use NFV-MANO and NFV-IFA
2020-06-19	V4.0.1	Initial version for Release 4
2020-09-03	V4.0.2	NFVIFA(20)000478r2 IFA010ed411 MegaCR FEAT17 Cloud-native VNFs NFVIFA(20)000533 IFA010ed411 Mirror of NFVIFA(20)000532 Fix usage of NFV003 and EM
2020-12-11	V4.1.2	Initial version for Release 4 drop 2
2021-03-05	V4.1.3	NFVIFA(21)000020r1 IFA010ed421 Rel-4 mirror of 007, SW image integrity NFVIFA(21)000119 IFA010ed421 (mirror of 076r1) Align NFV-MANO management to FEAT10 multi-site connectivity NFVIFA(20)000397r2 ENH02.02-IFA010ed411 New functional requirements
2021-03-18	V4.1.4	NFVIFA(21)000201r1 IFA010ed421 Mirror of 199 Avoid Reference to MAN001 and others NFVIFA(21)000030r3 Enh02.01 IFA010_release_4_megaCR NFVIFA(20)000835r2 Enh02.04-IFA010ed421 Use of invariant identifications for NSD constituents
2021-03-24	V4.1.5	NFVIFA(21)000197r2 IFA010ed421 MegaCR FEAT17 Cloud-native VNFs

History

Document history		
V4.1.1	November 2020	Publication
V4.2.1	May 2021	Publication