ETSI GS NFV-REL 006 V3.1.1 (2018-02)

Network Functions Virtualisation (NFV) Release 3; Reliability; Maintaining Service Availability and Continuity Upon Software Modification

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
DGS/NFV-REL006

Keywords
NFV, reliability, software

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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 IPR Policy, no investigation, 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.

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 requirements for the purpose of Software Modifications, such that NFV service availability and continuity is maintained. All types of software related to Network Function Virtualisation (NFV) - e.g. Virtual Network Functions (VNF), Management and Orchestration (MANO) and Network Function Virtualisation Infrastructure (NFVI) as well as required controlling and supporting functionality will be addressed. Where applicable, external specifications may be referenced to avoid duplication of work. The present document contains normative provisions.

2 References

2.1 Normative references

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

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

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

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

[1] ETSI GS NFV 002 (V1.2.1): "Network Functions Virtualisation (NFV); Architectural Framework".

[2] ETSI GS NFV-MAN 001 (V1.1.1): "Network Functions Virtualisation (NFV); Management and Orchestration".

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

[4] ETSI GS NFV-IFA 011 (V2.1.1): "Network Functions Virtualisation (NFV); Management and Orchestration; VNF Packaging Specification".

[5] ETSI GS NFV-IFA 013 (V2.1.1): "Network Functions Virtualisation (NFV); Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification".

[6] ETSI GS NFV-IFA 007 (V2.1.1): "Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification".

[7] ETSI GS NFV-IFA 005 (V2.1.1): "Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification".

[8] ETSI GS NFV-IFA 008 (V2.1.1): "Network Functions Virtualisation (NFV); Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specification".

[9] ETSI GS NFV-SOL 004 (V2.3.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification".

[10] ETSI GS NFV-SOL 005 (V2.4.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point".

[11] ETSI GS NFV-SOL 003 (V2.3.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point".

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] CRN website: "The 10 Biggest Cloud Outages Of 2013".

NOTE: Available at http://www.crn.com/slide-shows/cloud/240165024/the-10-biggest-cloud-outages-of-2013.htm.

[i.2] ETSI GR NFV-REL 007 (V1.1.1): "Network Functions Virtualisation (NFV); Reliability; Report on the resilience of NFV-MANO critical capabilities".

[i.3] ETSI GR NFV-IFA 021 (V0.11.0): "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on management of NFV-MANO and automated deployment of EM and other OSS functions".

[i.4] ETSI GS NFV-TST 001 (V1.1.1): "Network Functions Virtualisation (NFV); Pre-deployment Testing; Report on Validation of NFV Environments and Services".

[i.5] ETSI GS NFV 001 (V1.1.1): "Network Functions Virtualisation (NFV); Use Cases".

[i.6] Iulian Meamtiu and Tudor Dumistras: "Cloud Software Upgrades: Challenges and Opportunities", IEEE conferenceInternational Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2011.

NOTE: Available at https://www.umiacs.umd.edu/~tdumitra/papers/MESOCA-2011.pdf.

[i.7] ETSI GS NFV-IFA 018: "Network Functions Virtualisation (NFV); Acceleration Technologies; Network Acceleration Interface Specification".

[i.8] ETSI GS NFV-REL 003 (V1.1.2): "Network Functions Virtualisation (NFV); Reliability; Report on Models and Features for End-to-End Reliability".

[i.9] ETSI GR NFV-TST 006 (V0.0.8): "Network Functions Virtualisation (NFV); Testing; Report on NFV CICD and Devops".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

software rollback: software modification process that reverts the system from the newly deployed software version to the previously deployed software version

software update: software modification process for bug fixes or enhancements without adding, modifying or removing functionality, interfaces or protocols

software upgrade: software modification process aimed at adding, modifying or removing functionality, interfaces or protocols

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

BSS	Business Support System
CompHost	Compute Host
DF	Deployment Flavor
DUT	Devices Under Test
EM	Element Manager
EMS	Element Management System
EPC	Evolved Packet Core
FUT	Functions Under Test
IE	Information Element
IMS	IP Multimedia System
KPI	Key Performance Indicator
LCM	Life Cycle Management
MANO	Management and Orchestration
NAT	Network Address Translation
NFV TST	ETSI NFV Test Working Group
NFV	Network Function Virtualisation
NFVI	NFV Infrastructure
NFVO	NFV orchestrator
NS	Network Service
NSD	Network Service Descriptor
O&M	Operation and Maintenance
OS	Operating System
OSS	Operations Support System
PNF	Physical Network Function
SDN	Software defined Network
SUT	System Under Test
VIM	Virtualised Infrastructure Manager
VNF	Virtual Network Function
VNFC	VNF Component
VNFD	VNF Descriptor
VNFFG	VNF Forwarding Graph
VNFM	VNF Manager
VPN	Virtual Private Network
VR	Virtualised Resource

4 Overview

The intent of the present document is to specify requirements for the purpose of Software Modifications, such that service availability and continuity in an NFV environment are maintained. All types of software related to NFV (e.g. VNFs, MANO and NFVI) are considered for the software modification process; in addition, required controlling and supporting functionality is addressed. Where applicable, external specifications may be referenced to avoid duplication of work.

It should be noted that supporting work on the test processes for Software Modification is addressed by the ETSI NFV Test Working Group (NFV TST). As appropriate, references to the TST work and/or brief descriptions of the developed test processes are provided. An overview of the overall document framework is as follows.

Software related to NFV may be of different types. Clause 5.1 refers to the three working domains identified for NFV in previous work [1] and associates specific software types to these domains. The purpose for delineating these individual software types is to align specific processes or methods that may apply to the modification of each type of software as well as the corresponding requirements to support such processes. The software types are as follows:

- Clause 5.1.2 - VNFs that support the full range of Network Services and Applications: note that components comprising a VNF are referred to as VNF Components (VNFC). Thus modifications may be done for individual VNFCs or VNFs. VNF Descriptors and Attributes are included as appropriate.

- Clause 5.1.3 - MANO Software: Software components that provide MANO functionality.

- Clause 5.1.4 - NFVI Software: Software components that support NFV Infrastructure.

Software Updates and Upgrades have been defined for VNFs as follows [3]:

- A VNF update does not introduce new functionality and/or new interfaces.
- A VNF upgrade might introduce new functionality and/or new interfaces.

Detailed definitions for these terms indicating precise intent for these actions are provided in clause 3.1. Note that they apply to VNFs, NFVI, and MANO software components.

Software modification implications are presented in clause 5.2.

The methods by which modifications are instituted are as follows:

- "NetOps" or Network Operations - This method involves significant manual involvement throughout the modification process followed by substantial levels of testing by the Network Operator. Typically, this method could be instituted when major changes in software are involved with careful management of resource utilization. Additionally, it is expected that major changes will not be very frequent; this may allow for a carefully planned modification and testing process.
- "DevOps" or Development Operations - This method is fully automated end-to-end. The magnitude of changes is expected to be small for each new version and allows for automated testing throughout the process delivery cycle. It is also expected that such changes could be fairly frequent (e.g. multiple changes on a daily basis). The DevOps method is out of scope for the present document.

Clause 6 is structured as follows

- Clause 6.1: This clause provides an overview and introduces the phases of the software modification process.
- Clause 6.2: This clause describes the overall software modification process.
- Clause 6.3: Software Modification Requirements. This clause provides detailed normative requirements for modification of VNFs, MANO components, and NFVI components.
- Clause 6.4: Rollback of Software Modifications. This clause provides requirements on how to deal with problem situations associated with the software modification process.
- Clause 6.5: Test Process Implications. The test processes developed in the ETSI NFV TST Working Group are referenced and briefly described as applicable.

Additional information that is relevant for the software modification process is provided in informative annexes A, B and C.

5 NFV Software

5.1 Types of NFV Software

5.1.1 Introduction

The types of NFV Software that are in scope for the software modification process are described in ETSI GS NFV 002 [1], clause 5.2 as follows:

> "Network Functions Virtualisation" envisages the implementation of NFs as software-only entities that run over the NFV Infrastructure (NFVI). Figure 1 illustrates the high-level NFV framework. As such, three main working domains are identified in NFV:
>
> - Virtualised Network Function, as the software implementation of a network function which is capable of running over the NFVI.
> - NFV Infrastructure (NFVI), including the diversity of physical resources and how these can be virtualised. NFVI supports the execution of the VNFs.

- NFV Management and Orchestration, which covers the orchestration and lifecycle management of physical and/or software resources that support the infrastructure virtualisation, and the lifecycle management of VNFs. NFV Management and Orchestration focuses on all virtualisation-specific management tasks necessary in the NFV framework.

Figure 1 from clause 5.2 of ETSI GS NFV 002 [1] depicts the three working domains; this is shown below. The VNF domain is supported by the NFVI domain and both of these domains are managed by the MANO domain.

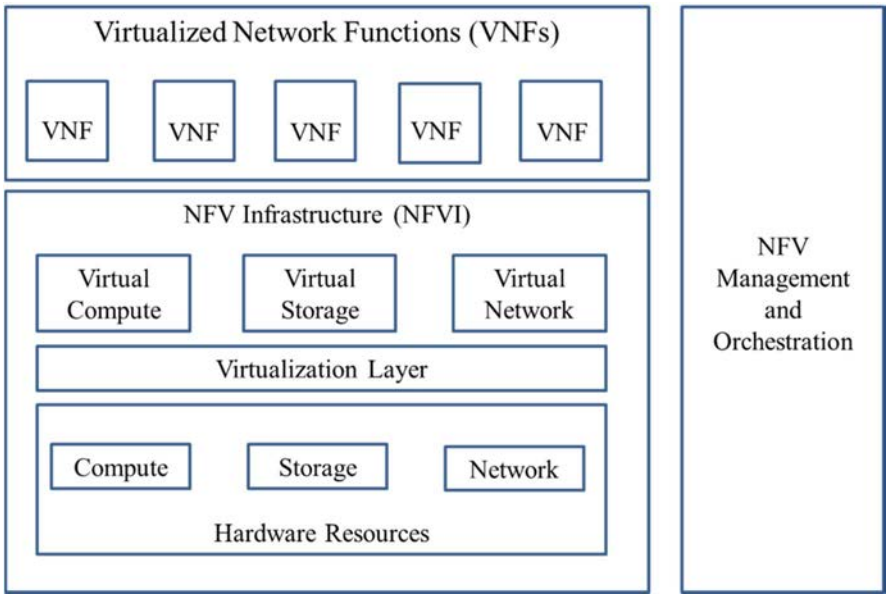


Figure 1: High Level NFV Framework as shown in clause 5.2 of ETSI GS NFV 002 [1]

The software in scope for the present document relates to these three domains.

5.1.2 VNF Domain Software

There are two types of VNF domain software:

1) Individual VNFs: Virtualised network functions that perform a given network task. Examples of individual VNFs include:

   a) Network Address Translation (NAT)

   b) Firewall

   c) Load Balancer

2) Network Services: An end-to-end Network Service is defined in ETSI GS NFV 002 [1] as a Forwarding Graph (FG) of the necessary Network Functions that support the desired service. This FG of Network Functions can include a combination of Physical Network Functions (PNF) as well as VNFs. From the perspective of the present document, only the portion of the FG that contains VNFs is in scope. Examples of services include the following:

   a) Mobile Voice/Data

   b) Internet Access

   c) Virtual Private Network (VPN)

5.1.3 MANO Domain Software

MANO domain software support the three MANO Functional Blocks in ETSI GS NFV-MAN 001[2]:

1) VNF Manager (VNFM).

2) Virtualised Infrastructure Manager (VIM).

3) NFV Orchestrator (NFVO).

It is assumed that the software for these Functional Blocks will need to be modified in their entirety; at the moment there is no breakdown available for splitting these blocks into smaller components.

5.1.4 NFVI Software

NFVI Software comprises the set of software that support all VNFs and Network Services. Examples of NFVI Software include:

- Operating System (OS),
- Hypervisor,
- Network Controller Software.

It is assumed that each of these software systems will need to be modified in their entirety unless individual software vendors create the software as a package of components.

5.2 Software Modification Implications

Network services do need to provide a defined level of availability. It is expected that service availability and service continuity will be maintained while modifying the software of any part of the NFV system. In DevOps environments in ETSI GR NFV-TST 006 [i.9], updates without functional changes may occur frequently and are deployed automatically. Therefore, the NFV system is expected to guarantee service continuity during software modifications at any time.

Especially when functional changes are introduced, the question of compatibility arises, e.g. data structure compatibility, internal and external interface compatibility, software functional compatibility, etc. In case of any incompatibility, the software modification needs to be carried out via upgrade procedures and different categories of incompatibilities need to be considered:

- Incompatibilities may arise among the entities to be modified for the duration of the software modification process, e.g. data structure incompatibility, internal interface incompatibility.
- As a result of software changes, incompatibilities may also occur between the modified entities and the rest of the system or the users of the modified entities, e.g. external interface incompatibility, software functional incompatibility.

The first category creates a reliability threat and needs to be considered in the solution design and resolved during the software modification process. The latter category cannot be resolved purely by the software modification process and the impact would become unavoidable for the environment and the users after the modifications if it was not considered in the software design. In order to avoid the outage from the latter category, the modified software needs to be backward compatible in external interfaces and software functions.

Incompatibilities may occur due to changes in the data structure, in the interfaces, their combinations, or even without interface and/or data structure changes due to behavioural changes. These changes need to be clearly indicated so that they can be taken into account in the design, planning and scheduling of the software modification process.

Even if data structures are changed in an incompatible manner, it is expected that the service continuity during the modification process will not be affected. This cannot be handled as an update. Instead an upgrade procedure has to be used to accomplish the software modification. It needs to be clearly indicated, if during the upgrade the reliability of network services is reduced. If so, then this upgrade can be scheduled at a proper time when the impact of reduced availability is (more) acceptable.

In addition, the software modification (e.g. upgrade) mechanism is expected to be implemented in such a way that the mechanism itself does not reduce the reliability of the running services.

6 Software Modification Process

6.1 Overview

The software modification process which is part of the overall Life Cycle Management process needs to consider the following tasks: downloading the new software to the network operator's domain, pre-deployment testing of the software in the network operator's lab environment, on-boarding the new software into the NFV system, preparing the deployment plans and any other prerequisite necessary for the software modification, deploying the new software, and the application of post-deployment field testing and follow up actions. These tasks can be grouped into two main phases:

- Software modification preparation phase: This phase includes:
  - the initial software download which is an interaction between the software vendor and the network operator for the purpose of software delivery to the network operator's domain;
  - testing the new software version in the network operator's lab environment;
  - on-boarding the new software into NFV system;
  - preparing the deployment plans and any other prerequisite necessary to carry out the software modification.
- Software deployment phase: This phase deploys the new software in the live NFV system. This means the instantiation of entities with the new software version to replace entities of the old software version. The switch over process may migrate the services to the new software instances gradually to ensure service continuity and/or to allow for post-deployment field testing and verification. To complete the software modification process follow up actions may be required as well as handling any failures that may occur in the process.

Further details on the overall software modification process are given in clause 6.2 while clause 6.3 elaborates on the software deployment phase for the different types of NFV software:

- Clause 6.3.1 - VNF software
- Clause 6.3.2 - Software related to NFV management and orchestration
- Clause 6.3.3 - NFVI software

Clause 6.4 discusses the considerations for handling failures during the software modification process and clause 6.5 provides a high-level overview of the pre-deployment testing process.

6.2 The Overall Software Modification Process

A high-level overview of the software modification process between the Software Vendor and Network Operator is as follows:

1) Software Vendor:
   a) New Software Version is complete.
   b) Vendor performs software version control testing that certifies accuracy and performance of the modified version.
   c) Vendor signals to Network Operator(s) about impending software update/upgrade delivery.

2) Network Operator:

a) Receives information of impending delivery of modified software:

- Type of Software includes:
  - VNF Software.
  - NFVI Software.
  - Management and Orchestration Software.
- Criticality of Modification:
  - Urgent: Critical bug fix; may require accelerated/instantaneous testing process followed by possible migration of existing traffic flows from old version to new version.
  - Normal: Can follow scheduled testing process with minimized level of migration of existing traffic flows.

b) Network Operator and Software Vendor engage in software download processes.

- Secure delivery of software from Software Vendor to Network Operator with following dependencies:
  - Protocol(s) used for download process.
  - Security of download (e.g. image authenticity).
- Network Operator informs Software Vendor of receipt of software.

c) Network Operator tests integration and performance of the new software in Lab.

- Type of testing:
  - Urgent.
  - Normal.
- Testing Phase:
  - Integration testing.
  - Test performance metrics.
  - Result of test:- certified or failed.
- Network Operator informs Software Vendor if pre-deployment testing failed otherwise proceeds to onboard the new software.

d) On boarding the new software.

- For VNF software the VNF package which includes its VNFD is on-boarded into NFV MANO system according to the related NFV specifications using the standard VNF package management operations.
- For MANO software if it is implemented as a VNF and delivered as VNF package, the on boarding is the same as for VNF software. Otherwise, the on-boarding procedure is MANO system or functional block specific.
- For NFVI software, the new software is on-boarded by the NFVI software management system which is not in the scope of NFV specification currently.

e) Planning of software deployment and verification of all prerequisites.

- Verifying the availability of resources, their configuration at the target deployment location(s) based on the following information:
  - Physical locations of the existing version of the software.
  - Available server capacity at physical locations in a global view (new version may or may not be installed at the location of existing software).
  - Processor capabilities at available servers (e.g. modified software may require same type of processor as existing software.
- Scheduling the software modification.
- Preparation of the software deployment plan, which includes:
  - The polices and strategies for the execution of the software modification.
  - Post-deployment testing and monitoring procedures to verify in-situ deployed instances of the new software.
  - The strategies for handling exception conditions, which may arise during the execution of the software modification or if post-deployment test procedure fails.
- Backup the necessary data and configurations of the currently running software.

3) Network Operator.

a) Deploys the new software according to the deployment plan prepared for the software modification:

- Software modification operations are initiated and post-deployment test and monitoring procedures are applied. The software modification operations depend on the type of the delivered software and they are discussed in the subsequent clauses.
- Service migration might follow different strategies, e.g. gradual migration during which the old and new software might provide services simultaneously.
- Post-deployment test and monitoring results on deployed instances might be evaluated periodically. If successful, the deployment of the new software continues until complete deployment at which point the old version is deactivated or removed as appropriate.
- In case of an exception during software deployment, if any post-deployment test procedure fails, or the monitored results are not satisfactory the modified software is removed from service according to the exception handling plans and old software continues to be used. More discussion will be in clause 6.4.

b) After the new software is deployed into the field, the Network Operator may monitor the software performance for some time (e.g. capacity in different traffic load) before the product certification results are submitted to the Software Vendor.

4) Software Vendor - Follow Up Action.

a) Product certification is issued - Success:

- The contracted performance requirements are fulfilled.
- No immediate action is required.
- Test results are examined for potential improvement of the software.

b) Product certification is not issued - Failure:

- The contracted performance requirements are not fulfilled.
- Rollback is initiated due to the final performance results of the monitored software.

- Test results are examined to determine cause of software fault/malfunction.
- Errors/bugs are fixed in the software.

c) Process cycle is repeated from Step 1.

6.3 Applicability for Software Types

6.3.1 Network Services/VNFs

6.3.1.1 VNF Software Modification Scenarios

It is expected that the download process between the software vendor and the network operator will be carried out by the network operator's Operations Support System/Element Management System (OSS/EMS). When the network operator has approved the new VNF software package for operating in NFV networks (after lab testing), on-boarding VNF packages into the network can be done via the management systems directly or via the operator's NFV Management and Orchestration System such as MANO. For the latter case, VNF packaging principles and a set of VNF Package management operations have been specified as follows:

- Principles of new software packages prior to release [4] and [9]:
  - The VNF Package contents, including the VNF descriptor, VNF binaries, configuration, scripts and software images, as well as manifest file, checksum, etc. as appropriate constitutes a single delivery unit from a distribution perspective.
  - Any changes to the constituency of this unit shall be considered as a change to the whole which shall be versioned, tracked and inventoried.
- On-boarding a VNF package to NFVO via the interface between OSS/BSS and NFVO [5] and [10].
- Adding images of a VNF package to the image repository managed by the VIM via the interface between NFVO and VIM [7].
- Subscription to and notification of on-boarding a VNF package, and fetching of VNF package/VNF package artefacts by VNFM via the interface between NFVO and VNFM [6] and [11].

A representative set of software modification scenarios are considered to ensure that service availability and continuity expectations are satisfactorily met:

1) **Software Modification Via Network Operator's Management System**: This scenario involves the onboarding of the new VNF packages directly by the OSS/BSS systems of the network operator and the initiation of the VNF software modification by the OSS/BSS or EM systems. The EM is the only VNF-aware sub-system capable of handling any VNF-specific coordination and management that may be required during the VNF software modification to ensure service availability and continuity.

2) **Software Modification for Different VNF Types**: As defined in clause 4, VNFs can be modified via software updates (minor changes) or as software upgrades (major changes). In case of software update, the VNF software modification operation is performed on an existing instance. However software upgrades often require the instantiation of new instances depending on the applicable method of upgrade.

3) **VNFC Software Modification**: Even though every new software package contains the complete VNF, the actual changes may involve only a subset of the VNFCs. Rather than update or upgrade the entire VNF instance by instantiating a new VNF instance with the new software version, it may be more efficient to modify only the affected VNFC instances. VNFC upgrades and updates become increasingly necessary due to the following reasons:

   a) As the market demand for new service introduction intensifies and agile development processes become more common (e.g. DevOps), there is a demand for frequent changes to the VNF software. Such rapid changes may best be accomplished by modifying individual VNFCs in a VNF.

   b) Research results [i.6] show that the release interval of cloud services is as short as one week or less.

c) Per [i.1], three of the 10 biggest cloud outages of 2013 and 2014 were caused by software upgrade related operations. It is important to enable upgrades of the smallest component of a VNF, which is the VNFC. This minimizes the following:

- risk of outages,
- negative service impacts,
- required resources for upgrade,
- duration of the VNF software upgrade process.

NOTE: The update process is the same for VNFs and VNFCs (see annex A).

4) **Simultaneous Operation of Old and New VNF or VNFC Software Instances**: Per clause 8.2 of [i.8], there may be occasions whereby instances of the new and old software versions simultaneously provide services during the software modification process. The instance(s) of the new software version only serve(s) selected new service requests dispatched by the traffic decider/load balancer based on some policies/rules, while the ongoing service requests and probably also some new service requests are still served by instance(s) of the old software version. The main benefits of this method are as follows:

a) Minimized risk of outage due to the modification by setting some policies or rules for the traffic decider/load balancer from the VNF management plane (for instance, EM or VNF operation and maintenance unit). Policies could be developed that permit the partial selection of specified service and/or user group traffic types for migration to the new software version in multiple stages. This permits software verification in a production environment.

b) Minimized migration of the state data from the old software instance to the new software instance. This is accomplished by allowing the majority of existing traffic flows to complete over the old software instance prior to its removal.

c) More reliable system because of the independence of the software modification procedure from the resiliency and scaling mechanisms associated with the old and new software instances.

Note that for the case of the simultaneous operation of old and new VNFC instances, dedicated scaling polices or rules might be necessary for the time of the software modification to replace the regular scaling policies.

6.3.1.2 VNF Software Modification Requirements

[**REQ.VNF.M.01**]: The VNF LCM shall support the VNF software modification process.

[REQ.VNF.M.02]: The VNF software modification shall support updates and upgrades of VNFs and VNFCs.

[**REQ.VNF.M.03]**: It shall be possible for the OSS/BSS to initiate the VNF software modification.

[**REQ.VNF.M.04]**: It shall be possible for the EM to initiate the VNF software modification.

[**REQ.VNF.M.05]**: The VNF package shall include the metadata for the VNF software modification.

[**REQ.VNF.M.06]**: The metadata for the VNF software modification shall provide fields to include the following parameters:

- The selectable policies/rules designed for the VNF software modification and their corresponding need of additional virtual resources (e.g. for instances of the new software version).
- The type of the VNF software modification, e.g. VNF software update.
- The VNF identifier and version identifier(s), e.g. the version number or name, to specify the software modified by the VNF package.
- Additional parameters will need to be addressed for the following cases:
  - For VNF or VNFC software update, identifiers of the modified software images and identifiers of the files or scripts for the software image modification.

- For VNFC software upgrade, identifiers of the modified software images of the old version and identifiers of their corresponding software images used in the new version package.

[**REQ.VNF.M.07]**: The VNF LCM process shall support instantiation of VNFC instances of the new software version within the VNF instance being upgraded during the VNF software modification.

[**REQ.VNF.M.8**]: The VNF LCM process shall support simultaneous service delivery over VNF or VNFC instance(s) of the old and new software versions during the VNF software modification.

[**REQ.VNF.M.9**]: The VNF LCM process shall support scaling out/up VNF or VNFC instance(s) of the new software version and scaling in/down of the VNF or VNFC instance(s) of the old software version during the VNF software modification as defined by policies/rules.

[**REQ.VNF.M.10]**: The VNF LCM process shall support termination of the old VNFC software instances during the VNF software modification on the interface to EM.

[**REQ.VNF.M.11**]: The VNF LCM process shall support interactions with EM to deliver coordination information needed for multiple stages of the VNF software modification process.

[**REQ.VNF.M.12**]: It shall be possible to determine the state of a VNF instance undergoing the VNF software modification.

[**REQ.VNF.M.13**]: When the deployment of the new VNF software version for a VNF instance is completed and the old software version has been replaced, the VNF information shall be updated to keep it in sync in the NFV system.

[REQ.VNF.M.14]: The VNF LCM process shall ensure the accessibility of the VNF software packages used during the VNF software modification operation. This may be needed for correct error handling, i.e. old VNFC instances are recovered from the old image while new VNFC instances are recovered from the new.

6.3.2 Management and Orchestration Software Modification Process

6.3.2.1 Introduction

In this clause, the term "Management and Orchestration components" refers to the three MANO components, the VIM, the VNFM, and the NFVO. The functionality associated with these three components and the critical roles they play in an NFV environment have been specified in ETSI GS NFV-MAN 001 [2].

From a reliability perspective, when the sub-system (i.e. VNF) managed by one of these components is confronted with failure or increased traffic load situations, the affected component may initiate and direct appropriate recovery processes such as VNF re-instantiation/healing, VNF scaling, and failure containment, while ensuring service availability and continuity. As discussed in ETSI GR NFV-REL 007 [i.2], the ability to deliver expected levels of service availability depends on a highly reliable set of Management and Orchestration components. To this end, it is critical that the software modification process for these components is carried out in a manner that continues to ensure desired levels of service availability and continuity.

The reasoning about the software modification process for these components is somewhat constrained as there are no definitive standards that state how these components are going to be constructed and developed. Hence, the modification process itself for these components would be speculative at this stage.

One potential way forward is to assume that at least some of these components will be developed as VNFs. If that proves to be the case, then their modification process and associated requirements are as described in clause 6.3.1 of the present document.

A more straightforward approach is to simply reference the appropriate Use Case developed in ETSI GR NFV-IFA 021 [i.3] clause 5.1.5 regarding the addition of individual Management and Orchestration components to the NFV system. As stated in this Use Case: "The goal is to add the NFV-MANO functional entity, e.g. adding a new VNFM to the list of VNFMs known to the NFVO (that NFVO may use for VNF LCM)". By accepting the software modification process based on this Use Case, the goal of the present document is then restricted to development of necessary architectural assumptions and associated requirements that can ensure maintenance of service availability and continuity expectations during the process.

6.3.2.2 Addition of Management and Orchestration Component - An Overview

The Use Case process for adding a Management and Orchestration component is described in clause 5.1.5 of ETSI GR NFV-IFA 021 [i.3]. This Use Case assumes the presence of a MANO Monitor for overseeing the modification. It also permits the modification of an individual component (VIM, VNFM, or NFVO) rather than modifying the entire Management and Orchestration system.

The flow is described see table 5.1.5.5-1 in ETSI GR NFV-IFA 021 [i.3] as follows.

Table 5.1.5.5-1: Addition of NFV-MANO functional entity flow description [i.3]

#	Actor/Role	Action/Description
Begins when	MANO Monitor	The MANO Monitor determines that a specific Added NFV-MANO functional entity needs to be added for Other NFV-MANO functional entity(ies).
Step 1	MANO Monitor ➔ Other NFV-MANO functional entity	The MANO Monitor requests the Other NFV-MANO functional entity(ies) to add the Added MANO functional entity as a known entity. The request includes necessary information for the Other NFV-MANO function to proceed with the addition, e.g. identification of the Added NFV-MANO function, interface reachability, etc.
Step 2	Other NFV-MANO functional entity	The Other NFV-MANO functional entity(ies) proceed()s with the addition of the Added NFV-MANO functional entity. This may require interaction between the entities, e.g. discovery of capacity, establishment of reachability, etc.
Step 3	Other NFV-MANO functional entity ➔ MANO Monitor	The Other NFV-MANO functional entity(ies) confirm(s) the addition of the Added NFV-MANO functional entity to the MANO Monitor.
Ends when	MANO Monitor	The MANO Monitor has been informed about the results of the addition of the Added NFV-MANO functional entity by the Other NFV-MANO functional entity(ies).

6.3.2.3 Critical Assumptions for Reliability

Given that a single component of the Management and Orchestration System is undergoing software modification, the following architectural assumptions need to be made from the perspective of reliability requirements.

Assumption 1: The architecture of the Management and Orchestration System has built-in redundancy for all potential failure as well as software modification conditions. The redundancy permits the management of critical entities to continue while the Management and Orchestration component under modification is undergoing software modification.

This permits the functionality to be shared equally between two versions of the same component. Thus for example, a VNF Manager is responsible for the following operations:

- Instantiation of VNFs.
- Scaling of VNFs.
- Updating and/or upgrading VNFs.
- Termination of VNFs.

This assumption requires some form of redundancy for the VNFM. Two possible scenarios can be described:

- The redundancy could be internal: Components within the VNFM may have built-in redundancy with efficient failover mechanisms that permit continuous operations for the VNFM. Under this scenario, the internal components of the VNFM can be modified directly by vendor defined methods.
- The redundancy could be external: There are two possibilities here:
  - Active-Standby Mode: Two VNFMs share the state of a common set of VNFs. In this case, one VNFM acts as a "Master" (or active) while the other acts as the "Backup" (or standby). Under this scenario, when the "Master" VNFM is modified, the "Backup" seizes control of the set of VNFs. Once the modification of the "Master" is completed (including in-field testing), the "Backup" is modified.

- Active-Active Mode: A pool of operational VNFMs may share the state information of a common set of VNFs. In this case, control of the impacted set of VNFs is taken over by a designated VNFM while the original VNFM that was in control is modified.

In both cases, the modification of the VNFM components can be accomplished using the procedure described in clause 6.3.2.2, i.e. by adding the new version of the component and removing the old, provided the internal redundancy is visible to the MANO Monitor. The architectural implementations for these types of redundancies are for further study.

Assumption 2: State information of the resources managed by any given Management and Orchestration component is immediately and readily available to the redundant Management and Orchestration component. After the software modification is completed, the Management and Orchestration component is still capable of interpreting the state information correctly.

Taking the example of the active-standby VNFMs from Assumption 1 above, the LCM state information of all VNFs that are managed by a VNFM is available to the standby VNFM. This ensures that the LCM state of all VNFs can be managed by the standby VNFM. The standby VNFM goes through the software modification first. When this is completed, the standby VNFM (of the new software version) reads and interprets the LCM state information of all VNFs, being thus ready to take over the active role. A switch-over is performed so that the other VNFM (formerly active, now standby) can go through the software modification.

Assumption 3: For the entities (VNFs and/or other NFVO-MANO components) managed by any given Management and Orchestration component, the modification process of the Management and Orchestration component is transparent. The backward compatibility of the management interface is maintained and the Management and Orchestration component after the software modification retains the same identity and accessibility from the perspective of its managed entities as before the software modification.

Taking the example of the active-standby VNFMs further, this means that the managed VNFs cannot detect that the VNFMs are undergoing software modification and the active role is being switched around during the process potentially between different versions of the VNFM. The VNFs continue to use the same management interface towards the active VNFM before, during and after the software modification in terms of version, way of access, identity, and authorization. For example, the VNFs receive notifications from the VNFM during and after the software modification according to their subscription made before the software modification has started. Similarly, the VNFs send their state indicators' value the same way before, during and after the VNFMs software modification process.

6.3.2.4 Requirements

The following requirements are necessary to ensure service availability and continuity during the software modification process of Management and Orchestration components.

[REQ.MANO.M-01]: All Management and Orchestration components shall be architected with acceptable levels of redundancy.

NOTE: The specific form of redundancy could be internal or external as stated above. The choice is up to the network operator and the selected component vendor.

[REQ.MANO.M-02]: Necessary state information shall be immediately made available to the redundant instances of the Management and Orchestration components.

[REQ.MANO.M-03]: While undergoing software modification all Management and Orchestration components shall maintain their identity, accessibility and backward compatibility towards their consumer entities to ensure the continuity of the management services.

6.3.3 NFVI Software

6.3.3.1 Introduction

The NFVI [1] is the totality of all hardware and software components which build up the environment in which VNFs are deployed, managed, and executed. It can be divided into physical hardware resources for computing, storage and networking, and software resources providing the virtualisation layer and the virtualised resources (e.g. hypervisors, VMs, VLANs, virtual disks, etc.).

The NFVI and its constituent components have a lifecycle independent of the lifecycles of the VNFs and MANO when hosted on that infrastructure. The layered nature of software, however, means that changes to the NFVI layer has the potential to have adverse effects on the VNF and MANO layers if care is not taken (as described in this clause) to avoid or minimize this impact. This potential impact affects the governance, precedence, and priority of management activities as a result of the software modification process.

An NFVI software modification procedure may be initiated after the successful completion of the initial software download procedure as described in clause 6.2.

6.3.3.2 Software Modification Precedence

The nature of layered architectures is that, through planning and technology, many software modifications of a given layer can proceed without impact on surrounding layers. This, however, is not universal, and there will be edge cases where a conflict arises between the need to make software modifications to the NFVI layer, and its potential to impact one or more workloads (VNF and/or MANO) hosted on that NFVI. In these cases, rules need to be defined for how to handle the conflict.

Since the purpose of NFVI is to support workloads it might seem essential to let those workloads dictate when a software modification to the NFVI can proceed. However, this approach, if unbounded, could indefinitely block NFVI software modifications from proceeding, with potentially significant impacts on stability, security, or new service deployment.

Conversely, limits in technology and impacts on services either through direct service disruption during a software modification, or indirect service disruption through pre-modification workload migration, may be deemed unacceptable for certain classes of workloads.

The balance between these extremes is to provide a limited facility for services to respond to a pending service disruption/outage at the NFVI layer, and take appropriate actions in a more graceful fashion than the actions that would be taken under failure conditions. This would generally take the form of a "notice period" of pending software modifications, and the time bound opportunity for services to respond on their own in preparation for such an NFVI software modification. This allows for more customized, fine-grained responses to predictable NFVI service disruptions.

It should be noted that no amount of testing can guarantee the success of an NFVI software modification. Thus, any notice period can serve only to reduce the potential for disruptive impact. A software modification might introduce a pathological behaviour, and so part of the software modification management process needs to be gradual, reversible as necessary, and risk controlled introduction of workloads onto the newly changed infrastructure, in order to further reduce the risk associated with software modifications.

6.3.3.3 Coordination of NFVI Software Modifications

A software modification will often affect far more than a single NFVI component, and might require the controlled deployment of software modifications across an entire resource pool. At the same time, VNFs and MANO components deployed on these NFVI components may need to take appropriate and potentially different actions to mitigate the potential impact. Some VNFs or MANO components may need to change their configuration, e.g. scale out to increase their redundancy, others may need to evacuate the virtualised resource being shut down for the time of the software modification, or block/reduce traffic directed to such an impacted virtualised resource.

Thus, the prerequisite for successful coordination is the ability to identify at the NFVI layer the constraints of the hosted VNFs and MANO with respect to the virtualised resources and their groups.

The grouping of virtualised resources relevant and known to both the NFVI and the VNF/MANO layers is the anti-affinity group. It is used typically to prevent single points of failure and therefore reflects redundancy used at the upper layers.

The constraints may be expressed when the virtualised resources and their groups are requested from or defined for the NFVI layer, for example, together with the manager entity who would need to be notified about the upcoming NFVI software modification. Alternatively, the manager entities could subscribe for such notifications.

The constraints among others might indicate the following:

- Whether the notification is requested before a virtualised resource or a group of virtualised resources (e.g. anti-affinity group) is impacted.

- Lead time requested for the notification as preparations may need time.
- Options such as an upper time limit at which virtual resource migration is feasible otherwise shutdown is preferred.
- For an anti-affinity group, the minimum number of virtual resources that need to be available or maximum number of virtual resources that can be impacted simultaneously.

Whenever an NFVI software modification is requested from the NFVI Software Modification Manager it needs to coordinate the software modification activities at the NFVI layer with the managers of the hosted entities by considering these constraints of virtualised resources and groups. The NFVI Software Modification Manager is responsible for managing the NFVI software modification procedure. Note however that at this time it has not been decided whether and which NFV functional block will implement this function. An exemplary coordination flow for the NFVI software modification can be found in annex C.

6.3.3.4 NFVI Resource Software Modification

A software modification of NFVI may include:

- Change of firmware of the physical equipment.
- Change of the host OS and/or hypervisor software, including virtual machines.
- Change of software providing virtual networks.
- Change of software providing virtual storage.

Note that similar consideration may apply to the replacement of physical computing, networking, and storage equipment.

The NFVI software modification is illustrated on the use case of an upgrade of a compute host presented in annex C.

In general, the software modification of individual resources need to be ordered in such a way that they do not impact the network services. In particular, VMs that participate in a redundancy schema at the VNF level are configured as an anti-affinity group, which needs to be taken into consideration. The software modification process of individual resources should impact at most an eligible number of VMs of an anti-affinity group at a time. The number could be constrained by the minimum number of virtualised resources that need to be available at any given time in the anti-affinity group (for example to maintain quorum) and/or the maximum number of virtualised resources that can be impacted simultaneously based on the replication schema used at the VNF or MANO layer.

With respect to virtualised network resources, the anti-affinity grouping of virtualised links requires similar consideration. Alternatively, the NFVI may provide reliable virtualised network resources, whose reliability needs to be maintained throughout the modification. The same approach is applicable to virtualised storage resources.

For a large data centre, it is probably not sufficient to perform the software modification of resources one by one as this would take considerable time. Careful planning of the NFVI software modification will be needed to optimize the time consumption. The entity that is responsible for planning the software modification should take appropriate steps for a smooth process including the following:

- Identify the resources that could be put in maintenance mode simultaneously.
- Identify satisfactory location to migrate resources in order to avoid repeating migration steps.
- Optimize resource relocation when the physical resources are limited.

Additional considerations may be required to accommodate potential NS and VNF LCM requests during the NFVI software modification as these operations may interfere with the software modifications and may be subject to SLAs. For example, VNF scaling in may remove just upgraded virtualised resources, while scaling out of the old version instead of the new version, prolongs the NFVI software modification process. Also, the software modification process may prevent scaling out or instantiation operations if too many resources are taken out from a resource pool for software modification, which in turn may cause performance degradation at the VNF and network service levels or prevent VNF healing after failure.

6.3.3.5 NFVI Software Modification Requirements

[**REQ.NFVI.M.01**]: A VNF instance or a manager acting on its behalf (e.g. EM/VNFM/OSS/BSS/NFVO) shall be able to receive an advanced notification about the software modification of an NFVI resource or group of NFVI resources as well as at the completion of the respective software modifications.

[**REQ.NFVI.M.02**]: The anti-affinity group or the resource group id assigned to a tenant shall be used to identify a group of NFVI resources with respect to NFVI software modifications.

[**REQ.NFVI.M.03**]: The NFVI software modification manager shall obtain the parameters of impact tolerance of hosted VNFs with respect to each of their virtualised resource(s), assigned resource group(s), and anti-affinity group(s) before executing any NFVI software modification.

These parameters include the following:

- The minimum lead time of the advanced notification.

NOTE 1: The lead time reflects the time the VNF needs to prepare for the potential disruption(s).

- In the case of resource groups, the minimum number of anti-affinity group members required to be available and/or the maximum number of anti-affinity group members that can be impacted simultaneously.

NOTE 2: The minimum number reflects the minimum number of virtualised resources required for the function/service to be provided (e.g. cluster membership requires quorum); the maximum number reflects the replication schema/redundancy.

- In the case of VM migrations, VM migration tolerance including at least the migration type and the maximum interruption time.

NOTE 3: A VM may tolerate live, offline or no migration at all. Considering the VNF level fault detection mechanisms, a maximum interruption time can be specified that can go unnoticed. The migration may also be constrained by affinity/anti-affinity groups and/or location constraints.

[**REQ.NFVI.M.04**]: It should be possible to specify whether the VNFM or NFVO should act on behalf of the VNF at NFVI software modification and the rules/policies to be used to prepare the VNF for it.

[REQ.NFVI.M.05]: During the NFVI software modification process the affinity/anti-affinity groups of virtualised resources shall be maintained according to the specified parameters. The NFVI software modification process shall not impact simultaneously more than the eligible number of virtualised resources of an anti-affinity group.

NOTE 4: The eligible number of resources depends on the currently available virtualised resources, the maximum number that can be taken out and the minimum number of members required in an anti-affinity group.

[**REQ.NFVI.M.06]**: The NFVI software modification process shall not impact the overall reliability and key performance indicators (KPIs) of the virtualised resources offered. SLAs shall not be violated if resource assignments are changed as a result of LCM operations during a software modification.

[REQ.NFVI.M.07]: During the NFVI software modification process, the compatibility requirements between virtualised and virtualisation resources shall be satisfied.

NOTE 5: For example, in case of the upgrade of the hypervisor or OS of virtualisation layer, the virtualised resource using the current VM image may become incompatible. The NFVI software modification needs to incorporate the VM image conversion process and ensure that VMs are migrated/failed over between compatible hosts and that reservations are adjusted appropriately during the upgrade, which means that if both old and new versions are being used simultaneously, both need to have access to reserved resources adequately.

6.4 Exception Handling of Problems during Software Modification Process

During the software modification process (clause 6.2 and 6.3), problems may be encountered such as:

- Lack of resources.

- Failure in the instantiation of new software.
- Unsatisfactory results from testing of new software.

Handling exceptions and other problems during the software modification process is crucial and it is obligatory when it comes to availability and reliability. From the perspective of service availability and continuity it is necessary to perform the removal of the newly deployed software in a graceful manner with minimal service impact.

Whenever a problem or an exception condition happens while the software modification is still in progress, if the root cause cannot be handled and the service impact is intolerable, it is necessary to roll back the software modification, i.e. revert back from the newly deployed software to the previously deployed old software to minimize any further impact. The details of the rollback process will depend on the criticality of the problem/exception. For example, losing the state for some services by an abrupt removal of the new software may be justified by the fact that this prevents fault propagation and further service degradation. In other cases, the new software can remain until the completion of or the migration of the services back to the old software as part of a graceful rollback process.

Since an exception (problem or failure) might happen in any of the procedures of the software modification phase, the software modification process needs to be designed for and include plans for exception handling. In addition, the rollback process may also need to be initiated by a management (e.g. administrative) request for certain conditions, such as degraded performance (beyond acceptable limits) even if the new software is operating correctly.

Considering the case of "Simultaneous Operation of Old and New VNF or VNFC Software Instances" of clause 6.3.1.1 an example software modification scenario may instantiate instances of the new software and migrate traffic to them from the instance(s) of the old software in three steps according to the following example schedule. Note that the migration would be typically managed by the EM as it is aware of the VNF behaviour, the traffic it serves and the related KPIs:

- Step 1: Transfer 10 % of the traffic to the instance(s) of the new software and monitor the KPIs for a period of time;
- Step 2: If step 1 is OK, transfer another 20 % of the traffic to the instance(s) of the new software and monitor the KPIs for another period of time;
- Step 3: If step 2 is OK, transfer all remaining traffic to the instances of the new software and monitor the KPIs for yet another time period before committing the software modification.

At any step, if there is a problem with the new software, if the monitored KPIs parameters are not satisfactory, or other verifications fail, the EM would at least stop the traffic migration to the new software instance by applying an appropriate traffic distribution policy and return an error message to the NFV MANO, which in turn delivers it to the initiator of the software modification process (e.g. OSS or EM). This initiator then would need to decide and issue the rollback request.

Once the software modification process has been completed and committed (e.g. Step 3 including the KPI monitoring phase completes with OK result), such a rollback procedure is not possible any more. Instances of the old software and even the package with the old software might have been removed from the NFV system. This may happen because the software deployment phase might be time constrained and the verifications of the new software performed during the software deployment phase might not be sufficient for a decision to approve/accept the new software. Thus, the decision to revert to the old software may occur after the completion of the deployment of the new software and the commitment of the software modification process. In such a case, the network operator is still able to revert to the old software version. However, this needs to be handled as an independent and new software modification operation, by which the old version of the software is designated as the "new/target" and the new version of the software is designated for removal. For this new software modification operation, the deployment phase needs to be designed and planned according to the features of the old version with respect to the new. For example, it is less likely that the old software can interact with the new version for state exchange purposes, or the new version may handle traffic not supported by the old.

In some cases, the problem/exception may be so critical that the only remedy is to restore an earlier version of the system (preferably the most recently backed up). Since such an operation loses the state of the services provided and restores an earlier state of the system and its services this is a last resort and should be considered carefully. Such a fallback operation should be fast and impact the minimum necessary scope. From this perspective, i.e. the necessary scope, the dependencies need to be considered between different parts of the system to identify independent subsystems, which need to be restored as a whole in order to preserve consistency at the service level. The prerequisite of such an operation is that an appropriate backup or snapshot is prepared for the given scope or subsystem at the initiation of the software modification, before the deployment phase.

Requirements:

[REQ. GEN. 001]: The software modification process plan shall include and be designed for handling exceptions and rolling back to the software version deployed at the initiation of the software modification process.

[REQ. GEN. 002]: The software modification process shall provide the capability of initiating a rollback any time before the completion and committing the software modification process.

[REQ. GEN. 003]: The software modification process shall include the ability to revert from the currently deployed version to a previously deployed version of the software.

NOTE: The reverting process should follow the requirements of the software modification process (e.g. maintaining service availability and continuity) regardless whether it is part of a rollback or it is a new software modification process.

[REQ. GEN. 004]: It shall be possible to restore the last known to be stable state of any independent subsystem, which has been stored/backed up before the start of the deployment phase of the software modification.

[REQ. GEN. 005]: The software modification process shall not impact the stored backup of the previously saved stable states of the system.

6.5 Test Process - High Level Overview

6.5.1 Introduction

This clause provides a high-level overview of pre-deployment test procedures for the modification of VNFs, Management and Orchestration Functional Blocks, and NFVI elements. This overview is based on ETSI GS NFV-TST 001 [i.4] wherein a "System Under Test" (SUT) is subjected to a set of test procedures to validate the new "modified system". The "system" in this case refers to the modified versions of VNFs, Management and Orchestration Functional Blocks, and NFVI elements. Detailed test procedures are described in ETSI GS NFV-TST 001 [i.4].

The relevant scope of ETSI GS NFV-TST 001 [i.4] is quoted from that document as follows:

The present document is an informative report on methods for pre-deployment testing of the functional components of an NFV environment. The NFV components addressed in the present document include Virtual Network Functions (VNFs), the NFV Infrastructure (NFVI) and the NFV Management and Orchestration (NFV MANO). The recommendations focus on lab testing and the following aspects of pre-deployment testing:

1) Assessing the performance of the NFVI and its ability to fulfil the performance and reliability requirements of the VNFs executing on the NFVI.

2) Data and control plane testing of VNFs and their interactions with the NFV Infrastructure and the NFV MANO.

The eligible SUTs for the recommended pre-deployment lab testing methods are provided in ETSI GS NFV-TST 001 [i.4] clause 4.2 as follows:

In the context of pre-deployment validation, the System Under Test (SUT) consists of one or more functions under test.

NOTE: The functions under test (FUT) are entities which are also commonly known as Devices Under Test (DUT) in the testing community. The term Device Under Test is not used in the present document in order to avoid ambiguities; devices are often considered to be physical entities which does not apply here.

In order to illustrate this concept, the functions under test could be, for example, implementations of functional blocks from the NFV architecture such as virtualisation layer or VNF. However, other physical or virtual components could as well be functions under test (FUT), like a virtual switch for example.

Each test specification validates one SUT where the SUT is one or more functional components of the NFV architecture. The SUTs considered for pre-deployment validation are the NFV Infrastructure (NFVI), a Virtualised Network Function (VNF), a Network Service (NS) or the Management and Orchestration (MANO).

It has to be noted that even though the MANO or parts of it are listed as potential SUTs, no direct pre-deployment validation methodologies of them are in the scope of this report. However they are required as supporting functional blocks for the validation of other entities and are listed for completeness and might be considered for further study.

Accordingly, these pre-deployment lab test methods apply to the modification of VNFs and NFVI elements.

6.5.2 Pre-deployment Testing for VNFs

The test environment for VNFs is quoted as follows from clause 4.6 in ETSI GS NFV-TST 001 [i.4].

For pre-deployment validation of a Virtualised Network Function (VNF), the SUT consists of one FUT which is the VNF Under Test, see figure 4.3.

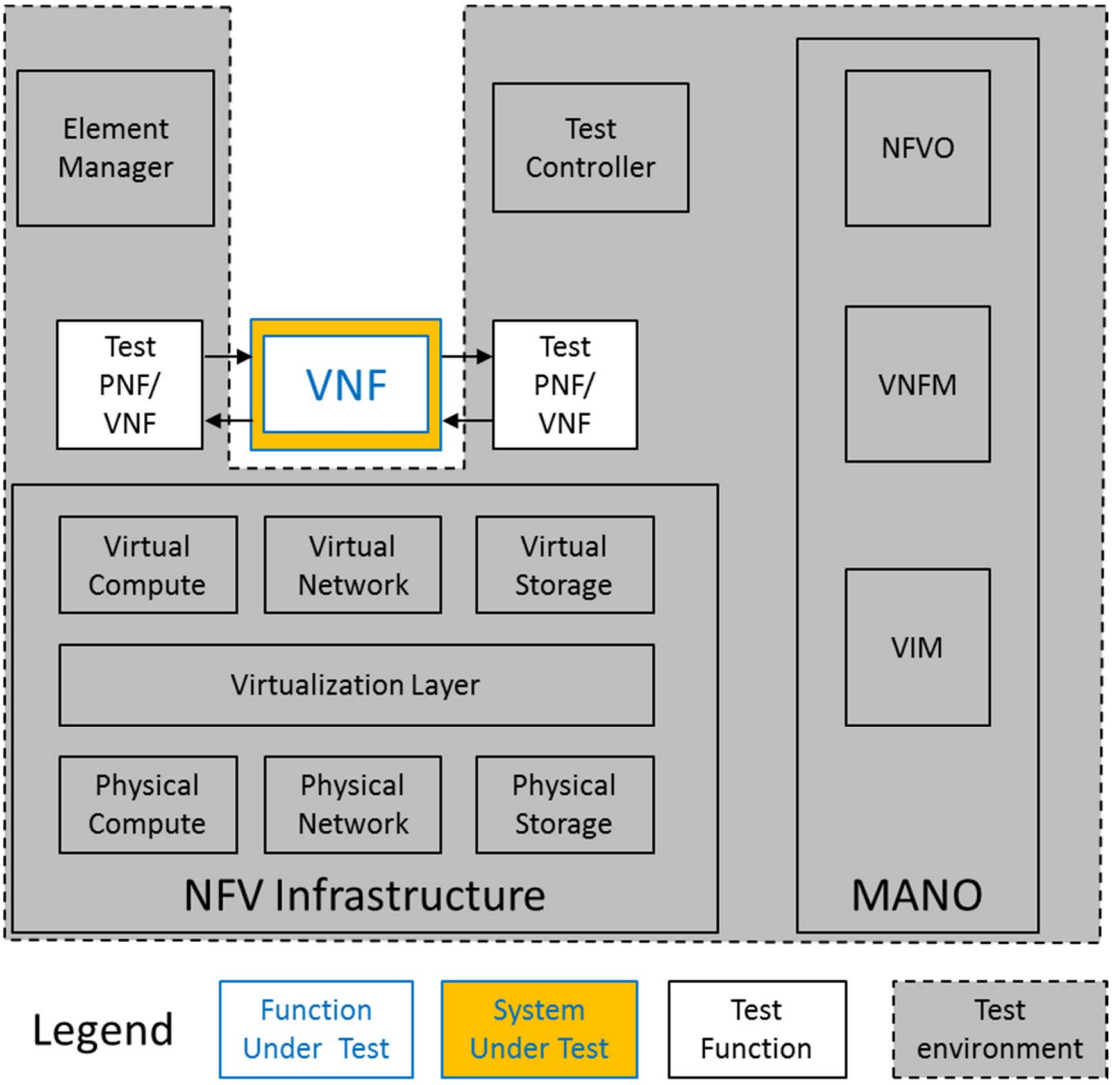


Figure 4.3: Functional architecture for VNF Under Test

The test environment consists of reference implementations of NFVI and NFV MANO functional components plus a Test Controller and Test PNFs/VNFs. In case required for maintaining the test PNFs/VNFs and the VNF Under Test, an optional Element Manager might be part of the test environment as well.

The Test PNFs/VNFs enable traffic scenarios towards the VNF Under Test and provide interfaces exposing access to functional and performance indicators.

The pre-deployment validations for VNFs are described in clause 7 of ETSI GS NFV-TST 001 [i.4] as follows:

- VNF Life Cycle Testing (Software Modification) is described in clause 7.1:

- VNF Instantiation Testing: clause 7.1.2.
- VNF Instantiation Testing in the Presence of Neighbouring VNFs: In this case, the VNF under test competes for NFVI resources with other VNFs that are adjacent to it. Test process for this case is described in clause 7.1.3.
- VNF Scaling Test Process: clause 7.1.4.
- VNF Termination Testing: clause 7.1.5.

- Data Plane Benchmarking for the New VNF Instance: clause 7.2.
- Control Plane Benchmarking for the New VNF Instance: clause 7.3.
- Control and User Plane Benchmarking for the New VNF Instance: clause 7.4.

6.5.3 Pre-deployment Testing for NFVI Elements

The test environment for NFVI elements is described as follows in clause 4.5 of ETSI GS NFV-TST 001 [i.4] as follows:

- For pre-deployment validation of the NFV Infrastructure (NFVI), the NFVI represents the SUT.

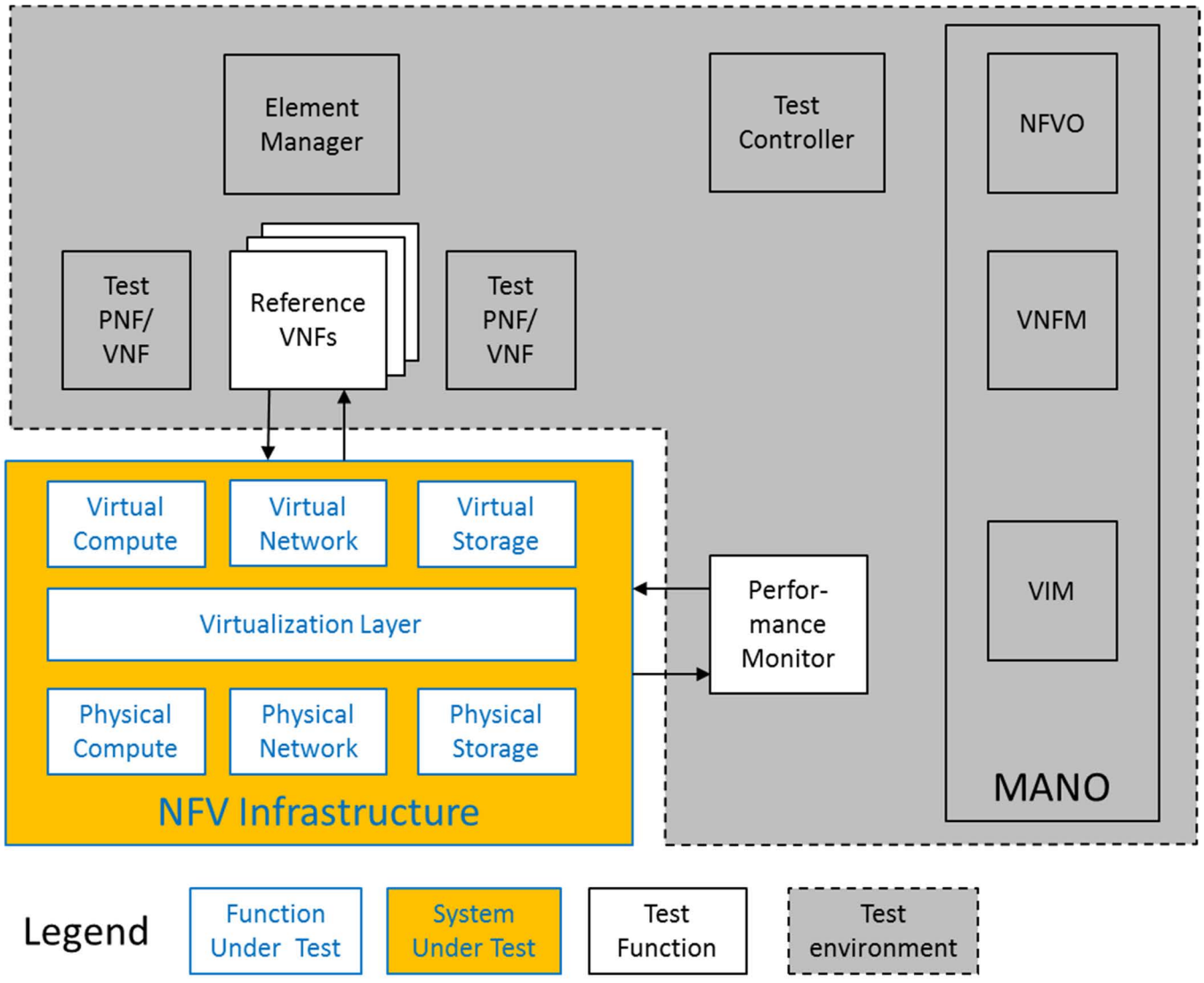


Figure 4.1: Functional architecture for NFVI under test

As illustrated in figure 4.1, the SUT comprises of the following functions under test (FUT):

- Physical Compute,
- Physical Network,
- Physical Storage,
- Virtualisation Layer,
- Virtual Compute,
- Virtual Network,
- Virtual Storage.

The test environment consists of a reference implementation of the NFV MANO functional components plus a Test Controller, Test PNFs/VNFs, Reference VNFs and a Performance Monitor. In case required for maintaining the test and Reference PNFs/VNFs, an optional Element Manager might be part of the test environment as well.

Different Reference VNFs as test functions are required to cover all aspects concerning different VNF types. The Reference VNFs are expected to be of the types described in ETSI GS NFV-SWA 001 [3], annex B, and shown in figure 4.2.

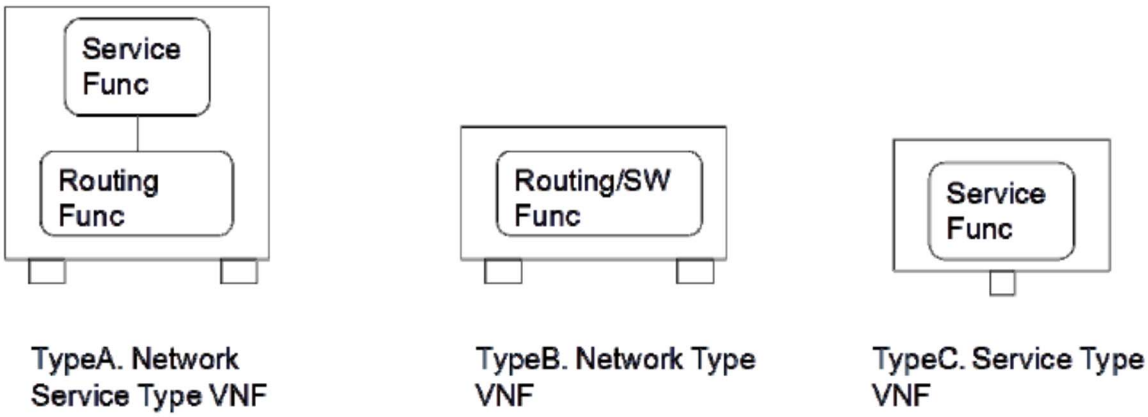


Figure 4.2: Reference VNF types (ETSI GS NFV-SWA 001 [3])

A Performance Monitor as test function is required to measure the performance indicators from the NFVI.

Optional test PNFs/VNFs might be required for certain test methods to enable traffic scenarios towards the Reference VNFs.

The pre-deployment validations for NFVI elements are described in clause 6 of ETSI GS NFV-TST 001 [i.4] as follows:

- Infrastructure Characteristics: clause 6.2.
- Scenario Validation: clause 6.3.
- Reference VNF Modeling: clause 6.4.
- Test Case Composition: clause 6.5.
- Validation Method: clause 6.6.

Annex A (informative): Analysis of VNF Software Modifications

A.1 Introduction

It is desirable that the VNF lifecycle management (LCM) operations ensure service availability and continuity during the software modification process. The purpose of this informative annex is to analyse the current VNF LCM process as it does not explicitly describe the VNF software modification process. In addition, since the automated operation of VNF lifecycle management is an essential innovation of NFV, it is desirable to automate VNF software modification operations.

In the software lifecycle, software updates (e.g. bug fixes) or software upgrades (e.g. introduction of new software functionality) cannot be avoided. Per generic VNF packaging rules [4], any change in the contents of a VNF package results in a new VNF package. However, this requirement does not dictate how the VNF software modification is executed.

A.2 Different VNF design and deployment patterns

The design of a VNF software modification depends on the VNF software architecture. These architectures could be classified as stateless or stateful. In particular, for a stateful VNF, the session data and the affinity of session connections need to be taken into consideration during the software modification process.

The following use cases examine software upgrades as well as software updates from the perspective of ensuring service availability and continuity during the modification process.

A.3 Use Case 1: Stateless VNF upgrade

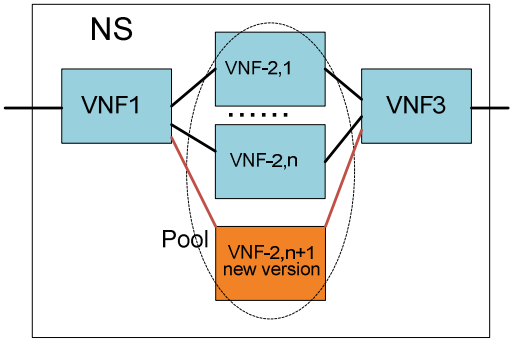
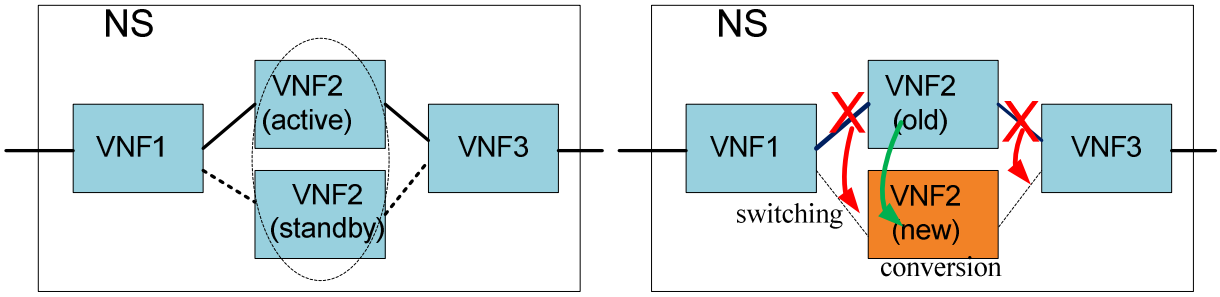


Figure A.1: Diagram of a stateless VNF software upgrade

For a stateless VNF, deploying a stateless VNF pool can provide resiliency. Figure A.1 shows a stateless VNF instance pool used to support a network service. Upgrading the software of a stateless VNF instance pool could be done by deploying new software version instances while gracefully shutting down and removing the old software version instances. Therefore, for stateless VNF instances, the network operator is able to use the VNF LCM operations of VNF instantiation and termination for the software upgrade while maintaining service availability and continuity. Theoretically, if the connected VNFs, e.g. VNF1 and VNF3 in figure A.1, support the functionality of configuring traffic policies, e.g. the amount of traffic sent to the instances of VNF-2 of the new version, the network operator can use it for directing the traffic with policies to minimize the risk of traffic disruption in case of a failure due to a fault in the new version of the VNF software.

A.4 Use Case 2: Stateful VNF upgrade

A.4.1 Deployment option 1: Stateful VNF with active-standby redundancy



(a) (b)

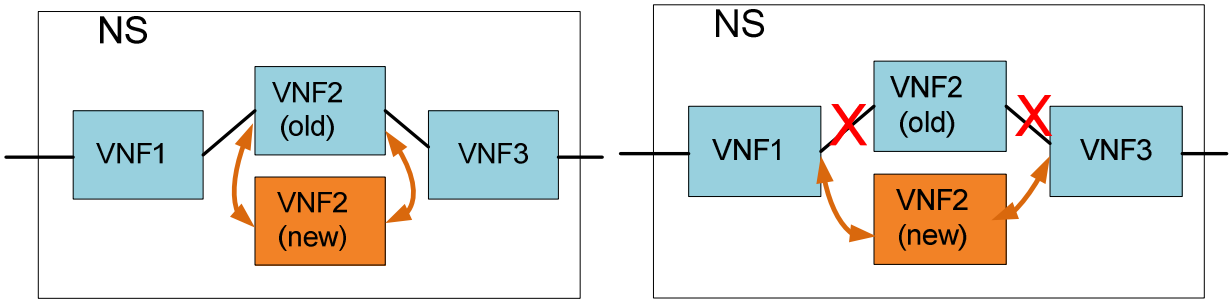
NOTE: (a) in an active-standby configuration (b) and their rolling upgrade by upgrading one VNF instance at a time.

Figure A.2: Diagram of stateful VNFs

Figure A.2 (a) shows the active-standby redundancy mode of a stateful VNF pair. Since a VM checkpoint technology based VNF redundancy is not currently a viable solution for most VNF applications, an application level active-standby VNF redundancy is used for such a resiliency deployment. A traditional rolling upgrade method discussed in ETSI GS NFV-REL 003 [i.8] may be utilized for performing the software modification by upgrading the standby VNF first. If applicable, the stateful data structure of the old VNF software version instance is converted into the corresponding data structure of the new software version before switching connections from the old VNF version instance to that of the new software version (figure A.2 (b). The drawback of this upgrade method is the lack of VNF redundancy in case of a failure while the upgrade process is under way. In addition, an outage might happen if the new software version or new configuration is not functioning properly.

A.4.2 Deployment option 2: Simultaneous operation of the old and new software version instances

Another method for upgrading the software of a stateful VNF instance has been given in ETSI GS NFV-REL 003 [i.8]. The method is to deploy a new software version instance and to use the old and new software version instances simultaneously for providing services without any state data conversion (figure A.3). The new software version instance only serves selected new service requests while the ongoing service requests, and probably also some new service requests, are still served by the old software version instance. The forwarding of new service requests to the new software version instance may be done by the traffic decider or by a load balancer/distributor as applicable. Policies/rules used by the traffic decider/load balancer/distributor determine how service request traffic is distributed into the old and new software version instances and the procedure of the traffic migration. The main benefit is a reduced risk of outage due to upgrade by setting some policies, or rules, in the traffic decider/load balancer/distributor from the VNF management plane (e.g. EM or VNF operation and maintenance unit).



(a) (b)

NOTE: (a) Selected new service requests are forwarded to the new software version instance while ongoing service requests and possibly some new service requests, are still served by the old software version instance. (b) As the ongoing services in the old software version instance decrease to a predefined value, the external connections to the old software version instance are switched to the new software version instance.

Figure A.3: Software upgrade of a stateful VNF without any data conversion

Implications for current NFV VNF LCM

Deployment options 1 and 2 show that a proper control of the service traffic migration from the old VNF version instance to that of the new version, is needed in the VNF upgrade process to maintain the service availability and continuity. The control functionality (e.g. traffic decider in ETSI GS NFV-REL 003 [i.8]) of service traffic can only be set and managed by the EM or the VNF operation and maintenance (O&M) if the traffic distribution is based on application level.

Both the OSS-NFVO and the EM are able to request the VNFM to instantiate a new VNF software version instance [6] and [8]. If the EM wants to upgrade a VNF instance, it can initiate a request to VMFM for instantiating the new software version and control the service traffic transfer between the two VNF instances. Therefore, the VNF software upgrade process could be initiated by the EM using the VNF instantiation and termination operations. However, since the EM does not have the NS information related to the updated VNF, when the VNFM sends the subsequent grant request, the NFVO will not know the related NS instance and the network service descriptor (NSD) that might require the applicable VNF level affinity and anti-affinity rules. Consequently, the NFVO might not grant the virtual resources needed for the software upgrade or might place the VNF instance of the new software version far from optimal and might even violate the affinity and anti-affinity rules defined in the NSD for the old VNF software version instance. In addition, once the software upgrade has been completed, the corresponding VNF information of the NS needs to be updated. Such an update operation, which has been specified as the function of NFVO [5], cannot be done by the EM directly.

Another issue of concern for this case is software image management. Software images are managed through the VIM via NFVO. Since the EM might not have the information of the new software images, it might request virtual resources from VNFM with the instantiation operation for the software upgrade without triggering the upload of the software images to VMs. The necessary upload of these software images to the VMs will have to be done by other means.

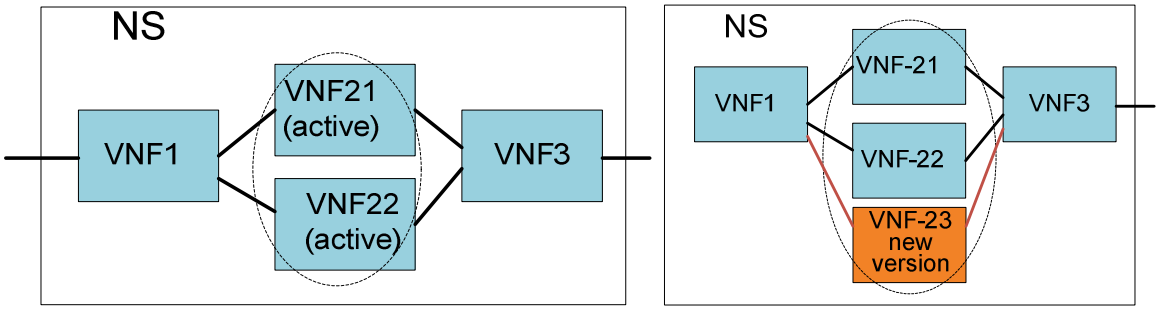
Creating a new VNF instance using the path OSS-NFVO-VNFM does not encounter the issues described above (i.e. regarding the software images and placement policies). However, the software upgrade cannot also be completed this way because the EM may not be aware of the newly instantiated VNF (by the NFVO-VNFM path) as part of the software upgrade process, and it also cannot get the information needed for controlling the new software version instance, which is necessary for maintaining service availability and continuity.

A.4.3 Deployment option 3: Stateful VNF with load sharing

Unlike the software upgrade of a stateless VNF (figure A.1), software upgrade of a stateful VNF with load sharing (figure A.4) cannot rely only on the VNF instantiation and termination operations due to the following reason.

If a new service request from a user or session is forwarded to VNF-22, the subsequent service requests from the same user or session need to be delivered to VNF-22 also in order to maintain the service affinity. Thus VNF-22 maintains the service state data for the same user/session and the user/session remains actively connected. Only the EM can override this. Thus, the EM needs to control the migration of the traffic to a new software instance, VNF-23. Hence, without the EM, VNF-22 cannot ever terminate its service functions.

Therefore, if service continuity needs to be maintained during the software upgrade process, it needs some additional processing software from the vendor which is managed from the VNF management plane, e.g. EM, to control the software instance upgrade.



(a) (b)

NOTE: (a) with load sharing (b) and adding a VNF instance of the new software version into the NS.

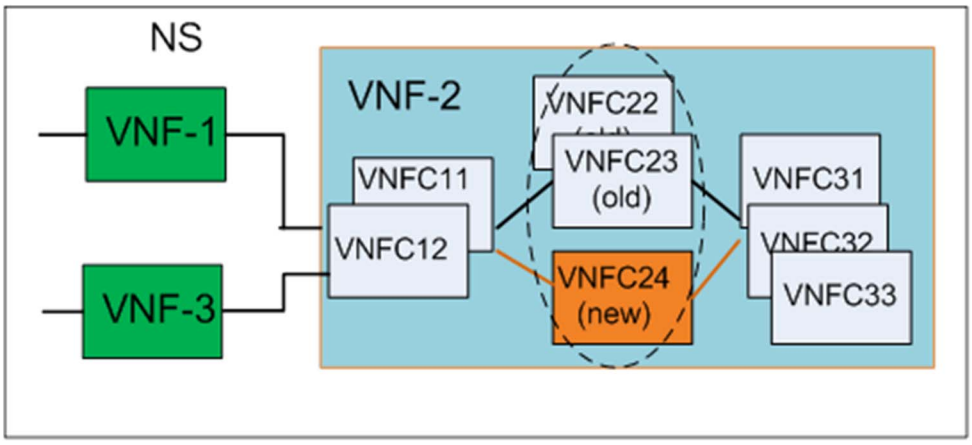
Figure A.4: Stateful VNFs

For this deployment option, if the VNF does not have any internal resiliency mechanisms, the VNF software upgrade process could use a method similar to the one discussed in the previous deployment options 1 and 2 (e.g. instantiating the new software version and forwarding only new service requests to such instances). The implication of current NFV VNF LCM for deployment options 1 and 2 is also applicable to this deployment option. However, for a reliable system, the stateful VNF in a load sharing configuration is normally combined with VNF internal resiliency mechanisms. Without such mechanisms, the affected services cannot be recovered in case of a failure in the VNF. Therefore, VNF load sharing configurations without any VNF internal resiliency mechanism are not commonly used in telecommunications networks.

If a VNF has internal resiliency mechanisms in this deployment option, the upgrade of the VNF software could be done at the level of the VNFC software upgrade and will be discussed in the next case.

A.4.4 Deployment option 4: Stateful VNF with an internal resiliency mechanism

For a stateful VNF in a network service, the typical resiliency implementation is to combine the VNF internal resiliency mechanism with network resiliency techniques. The active-standby VNF deployment in the previous deployment option 1 would rarely be used when the VNF application is complex. For example, if a VNFC instance is not functioning properly in the active VNF instance, the problem cannot be detected and the failure cannot be recovered by the standby VNF instance. The VNF internal resiliency is a vendor specific solution and the mechanism could be implemented in different ways. For example, VNFC resiliency could be implemented by using active-standby redundancy, or a load sharing model, or each VNFC might have its own resiliency mechanism. Figure A.5 shows an example of the mixture of active-standby and load sharing VNFCs in a VNF.



NOTE: Type 1 VNFC: VNFC11 and VNFC12 as redundancy pairs; Type 2 VNFC: VNFC22 and VNFC23 as a load sharing pool; Type3 VNFC: VNFC31, VNFC 32 and VNFC33 as another load sharing pool.

Figure A.5: An example using the mixture of active-standby and load sharing VNFCs in a VNF

In order to introduce a new functionality in a VNF, the software upgrade might entail the whole VNF or one or more types (e.g. figure A.5) of VNFCs. The VNF software upgrade depends on the software architecture and functionality implemented. If the change of the VNF software only involves one type of VNFC, it is not desirable to upgrade the whole VNF software. Only upgrading the changed parts of the software would have the benefits of minimizing the service impact, minimizing the risk of outage, minimizing the required resources and the duration of the upgrade process. An example (figure A.5) of upgrading the software of the type 2 VNFC (figure A.5) to introduce a new functionality is to use the upgrade methods discussed in deployment option 2:

- The type 2 VNFC instances of the new VNFC software version are instantiated to serve selected new service requests while ongoing service requests, and probably also some new service requests are continued to be served by the old VNFC software version instances.
- Appropriate policies or rules by the VNF management plane will have to be created in the traffic decider, to support traffic migration as described in option 2.

Implication for current NFV VNF LCM

Only the impacted VNFCs within a VNF are considered for upgrade. In addition, the VNFC software upgrade will need to deploy different versions of the VNFC software for a given type of VNFC in a VNF. However, current specifications neither support the instantiation/termination of a VNFC instance, nor support the deployment of VNFC instances of different software versions within one VNF instance. Therefore, the current specifications need to be evolved to support the VNFC software upgrade.

A.5 Use Case 3: VNF software update

A VNF software update does not result in the introduction of a new functionality, interface, or protocol but typically provides some bug fixes or enhancements. This is normally done by modifying a software module of the VNF software. From the NFV perspective, the smallest identifiable component of a VNF instance is a VNFC instance which is hosted in a VM. The VNF software update process could be done the following way:

- The VNF software vendor releases the executable files or scripts for updating the software images, or the binaries executing in NFV systems. Such executable files or scripts are normally called "patches".
- When a new VNF package containing the patches and other update information is delivered to the environment of a network operator, the on-boarding of new VNF packages is done.
- The VNF software update process includes loading the patches into the VMs and executing the "patches".

Since the software update results in changes of the VNF software source code, after on boarding a new VNF package, it needs to trigger the update procedure on the targeted components of the related VNF instance. However, the following issues have not been covered in current NFV specifications:

- The VNF software update related information has not been defined in VNF packaging format.
- The VNF software update operation has not been defined in current specifications of VNF LCM.

A.6 Use Case 4: NS update with simultaneous software upgrade of multiple VNF instances

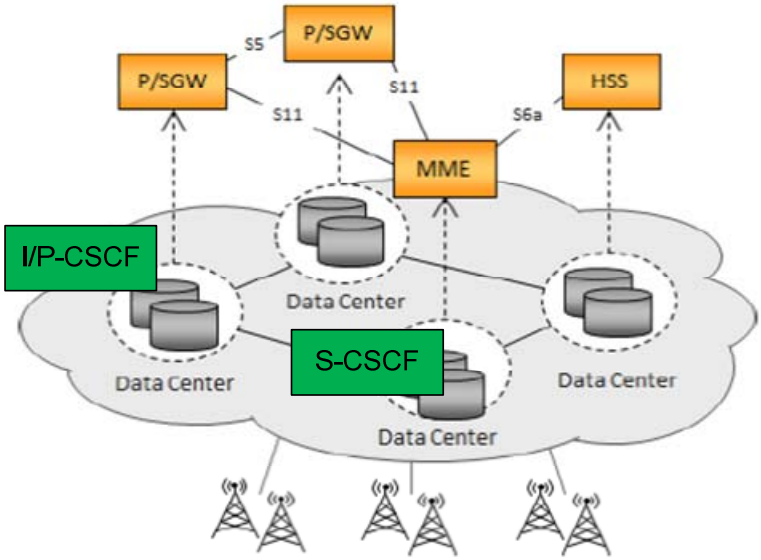


Figure A.6: An example of a virtualised EPC and IMS [i.5]

The example of a virtualised EPC and IMS in figure A.6 [i.5] shows that upgrading the VNFs' software or/and updating the VNFs' configuration in the data centre might be necessary for introducing a new network functionality, e.g. VoLTE.

This use case shows that more than one VNF instance of a NS may need to be upgraded for enabling a new network feature. Whatever method of software upgrade is used, the operating network services are expected to be always available and disruption of ongoing connections is not expected.

According to current NFV NS LCM, the entities of NS management, e.g. OSS/BSS/NFVO, cannot initiate the operation of VNF instance software update/upgrade.

A.7 Summary

From the preceding use case analysis, the following main issues have been identified and some of these issues need to be considered in the further development of specifications to support/enable the VNF software modification process:

- For some configurations, if the EM wants to upgrade a VNF instance, it can instantiate a new VNF instance with the new software version and control the transfer of service traffic migration from the old software version instance to the new VNF version instance. Therefore, the EM could handle the VNF software upgrade via the VNF instantiation and termination operations while maintaining service availability and continuity. However, when the VNFM sends the subsequent grant request to the NFVO, the related NS instance and therefore the NSD that provides the VNF level affinity and anti-affinity rules are not available to the NFVO. Consequently, the request of granting virtual resources might be rejected, or the placement of the new VNF instance may be far from optimal or might even violate the affinity and anti-affinity rules of the NSD. In addition, the EM cannot update the necessary information about the new VNF directly. Hence the EM by itself cannot upgrade the VNF.

- The issues related to the EM do not apply to the creation of a new VNF instance using the path OSS-NFVO-VNFM. However, OSS-NFVO is unable to use the VNF instantiation and termination operations for software upgrade because the upgrade related information cannot be delivered to the EM to control the service migration during the upgrade process.
- Neither the EM nor the OSS-NFVO can perform the VNFC software upgrade.
- In the specified VNF package contents, there is no information element defined for VNF software modification process which is nevertheless needed for performing the VNF software modification in a multiple vendor NFV environment.

Annex B (informative): Potential solutions for VNF software modification

B.1 Introduction

Clause 6.3.1.2 has specified the requirements for supporting the VNF software modification in VNF LCM. The analysis of the VNF software modification presented in clause A.1 has shown that some VNF software modification use cases can while other cannot be carried out with the current VNF LCM. This annex presents some ideas on potential solutions to support VNF software modifications. First in clause B.2 the alternatives requiring no or minimal changes to the VNF LCM are discussed and the circumstances under which they can be applied. Then those alternatives are presented that require further development of the VNF LCM to support VNF software modifications with the purpose of general applicability.

B.2 Alternative Solutions for Supporting VNF Software Modification with Existing LCM Operations

B.2.1 General

This clause describes solutions that do not require specifying new LCM operations although new parameters might be required in some cases. It identifies two software modification methods depending on whether new VNF instances of the VNF have to be created. Regardless of the method used, a pre-requisite is to on-board the new VNF package that contains the new software images. The OSS determines from the metadata in this VNF package the method to use for performing VNF software modification. Further applicability constraints are outlined for each of the cases as required.

B.2.2 Alternative Method #1

This method requires creating new instances of the VNF whose software is to be modified. It is typically used when there is a need for the old and the new version to exist simultaneously for a period of time (e.g. to ensure service continuity in case of stateful VNFs). It is more likely to be used for software upgrades but can be used in case of software update as well.

The OSS coordinates the overall process and proceeds as follows:

a) The OSS sends an "Update NSD" operation of the NSD management interface to the NFVO according to ETSI GS NFV-IFA 013 [5], in order to reference the new version of the VNFD.

b) The OSS sends an "Update NS operation" of the NS LCM management interface to the NFVO according to ETSI GS NFV-IFA 013 [5], with the "updateType" set to "AssocNewNsdVersion" to associated the NS instance with the updated NSD.

c) The OSS sends an "Update NS operation" of the NS LCM management interface to the NFVO according to ETSI GS NFV-IFA 013 [5], with the "updateType" set to "instantiateVnf". The new vnfd identifier referencing the new software images is included in the InstantiationVnfData IE.

d) If the current NS deployment flavour does not permit adding a new VNF instance (e.g. because the number of VNF instances for the current DF has been reached), the OSS sends an "Update NS operation" with "updateType" set to "ChangeNsDf" before requesting the creation of the new instance. It is assumed that a suitable NS flavor has been designed by the network operator for that purpose.

NOTE: An alternative would be to include an override flag in the "Update NS" operation instructing the NFVO to ignore VNF instance boundaries associated with the current NS deployment flavour. This approach would require adding a new information element to this operation.

The NFVO and the VNFM proceed with the VNF instantiation according to the procedures defined for this purpose in ETSI GS NFV-IFA 013 [5] and ETSI GS NFV-IFA 007 [6].

Once the OSS is made aware that the NS has been updated with a new VNF instance, it proceeds according to one of the following approaches so as the traffic (or a portion of the traffic) can be redirected to the new VNF instance:

a) If traffic redirection is to be handled by the NFVI, it issues another "Update NS operation" with updateType set to "UpdateVnffg" to update the list of network forwarding paths. Alternatively, the OSS can request an SDN controller in the NFVI to modify traffic forwarding rules directly, if the OSS has direct access to this controller.

b) If traffic redirection is to be handled at the NS level (e.g. the NS includes a Load Balancer VNF or the old VNF redirects the traffic), it informs the VNF (or its EM) in charge of traffic redirection by means outside of NFV standardization.

The traffic redirecting process may proceed with multiple iterations for ensuring service continuity while scaling out of the new VNF instance; scaling in the old VNF instance then can be done during the modification process. Once the software modification has been carried out and all traffic redirected to the new instance, the OSS sends an "Update NS operation" with "updateType" set to "ChangeNsDf" in order to return to the NS deployment flavour equivalent to the original.

B.2.3 Alternative Method #2

B.2.3.1 General

This method does not require creating new instances of the VNF whose software is to be modified. There are two variants depending on whether new VNFC instances have to be created within existing VNF instances. The assumptions are that:

- the virtual resources of the currently deployed version are suitable for the new version including the defined deployment flavors,
- the EM (or VNF instance itself) has access and is capable of managing the software images within the virtual containers whether this is necessary for the purpose of software modification, scaling, or failure handling,
- the EM is subscribing to the VnfLcmOperationOccurrenceNotifications to be aware of the VNFC instance IDs.

B.2.3.2 Variant A

This variant does not require creating new instances of the VNFCs whose software is to be modified. It is more likely to be applicable to VNFC software update but can be used in case of VNFC software upgrade as well as long as the software modification does not require changes to the virtualised resources and topology and there is no need to for the new and old versions to coexist.

The OSS connects to the VNF instance (directly or via an EM) to modify the software of one or more of VNFC instances. How this is achieved is outside the scope of NFV standardization. It is assumed that the new software images are those referenced in a new version of the corresponding VNF Package/VNFD. In other words, the resulting VNF instance is identical to a VNF instance that would have been created based on the new version of the VNF package. Once the new software has been uploaded, the OSS/BSS sends an "Update NS" operation with the "updateType" set to "ModifyVnfInformation" in order to associate the new VNF package identifier to the VNF instance that has been updated.

B.2.3.3 Variant B

This variant requires creating new instances of the VNFCs whose software is to be modified. It is typically used when there is a need for the old and the new version of these VNFCs to exist simultaneously for a period of time (e.g. to ensure service continuity in case of stateful VNFCs). It is more likely to be used for to VNFC software upgrade but can be used in case of VNFC software update.

The OSS sends a VNF software modification request to the EM. The contents of this request are outside the scope of NFV standardization. The EM issues a "Scale VNF" operation with the "type" parameter set to "scale-out" to the VNFM in charge of the VNF instance whose software is to be modified. The number of steps and the aspect are selected based on the VNFC(s) to be updated. For optimization purposes a new value of the "type" parameter could be defined to specify that no software image should be uploaded in the new VNFC instances or that specific software images other than those associated with the current package should be uploaded, in which case a reference to these software images is included in the request.

The VNFM then proceeds with the VNF scaling according to the procedures defined for this purpose in ETSI GS NFV-IFA 007 [6] and ETSI GS NFV-IFA 008 [8]. This includes sending a Grant request to the NFVO.

If the current VNF deployment flavour does not permit adding a new VNFC instance, the EM sends a "Change Deployment Flavour" operation to the VNFM before requesting the creation of the new instance. It is assumed that a suitable VNF flavour has been designed by the VNF provider for that purpose.

NOTE 1: An alternative would be to include an override flag in the "Scale VNF" operation instructing the VNFM to ignore VNF instance boundaries associated to the current VNF deployment flavour. This approach would require adding a new information element to this operation. A further alternative would be to add a "purpose" parameter to the "Scale request" indication whether the scaling procedure is used for the purpose of resource capacity management or for the purpose of software modification.

Once the EM is made aware that the scaling procedure has taken place successfully, it connects to the newly created VNFC instances and uploads the new software version (if the new software image was not referenced in the Scale request).

Depending on the type of VNF, traffic redirection to the new VNFC instances can be managed by the VNF alone (e.g. in case there is an in-line load balancer) or the VNF can request the NFVI to modify forwarding rules if a dedicated switch has been assigned to this VNF instance (e.g. using the procedures defined in ETSI GS NFV-IFA 018 [i.8].

Once all the traffic has been redirected to the new instances, the EM issues another "Scale Vnf" operation with the "type" parameter set to "scale-in" and a new parameter providing the list of VNFC instances to be removed.

Once the software modification has been carried out and all traffic redirected to the new instance, the EM sends another "Change Deployment Flavour" operation to the VNFM in order to return to the VNF deployment flavour equivalent to the original. The EM also reports the completion to the OSS so it can send an "Update NS" operation with the "updateType" set to "ModifyVnfInformation" in order to associate the new VNF package identifier to the VNF instance that has been updated.

NOTE 2: Since during the software modification process still the old VNF package and its images are associated with the VNF instance, non-EM initiated LCM operations such as auto-scaling or VNFM initiated healing of failed VNFC instances will use those (old) images. Hence the EM may need to reload the appropriate upgrade image into those VNFC instances. It is desirable to disable, using the ModifyVNFInformation operation defined in ETSI GS NFV IFA 008 [8], non-EM/VNF initiated scaling operations to minimize the interference with the ongoing software modification procedure (e.g. removal of upgraded VNFC instances).

B.3 Alternative Solutions for Supporting VNF Software Modification with New LCM Operations

B.3.0 Introduction

In clause 6.3.1.2, there is a requirement that the VNF LCM supports the VNF software modification and that it can be initiated from the OSS/BSS or the EM. This is to allow the OSS/BSS and the EM explicitly coordinate the software modification process with the NFV MANO components, i.e. make the NFV MANO components aware of an ongoing software modification, and to ensure the general applicability of the process.

It is understood that the solution for the VNF software modification should be consistent with the principles of the current VNF LCM as much as possible. The following guidelines are taken into consideration in this potential solution for the VNF software modification:

- The principles of the current VNF LCM should be reused as much as possible in the design of the VNF software modification operation.
- The VNFD format should not be modified. The needed information for the VNF software modification can be included in the VNF package as a separated file with the metadata of the VNF software modification.
- The principles of the VNF software packaging and the VNF package management of the current specifications should be reused.

B.3.1 Overview of the VNF Instantiation Operation

There are three possible ways to trigger the VNF instantiation process in the current VNF LCM:

- According to ETSI GS NFV-IFA 008 [8], the process begins when the EM sends a ***CreateVnfIdentifierRequest*** to the VNFM with the appropriate parameters, e.g. VNFD ID, etc., to create a VNF identifier (clause 7.2.2) of [8]) for a new VNF instance. When the EM receives the response from the VNFM with the identifier of the VNF instance, it sends an ***InstantiateVnfRequest*** to the VNFM to instantiate the VNF instance (clause 7.2.3 of [8]).
- According to ETSI GS NFV-IFA 013 [5], the OSS/BSS sends to the NFVO an ***UpdateNsRequest*** with the required parameters, such as nsInstanceId, updateType = "InstantiateVnf", instantiateVnfData, etc., to request the instantiation of a VNF instance (clause 7.3.5.2 of [5]) for the appropriate NS. The instantiateVnfData contains the parameters that are needed for the VNF instantiation. The NFVO uses those parameters to begin the VNF instantiation process. The process completes with the instantiation of the VNF as described in the next bullet.
- According to ETSI GS NFV-IFA 007 [6], the NFVO sends a ***CreateVnfIdentifierRequest*** with the appropriate parameters, e.g. VNFD ID, etc., to create a VNF identifier (clause 7.2.2 of [6]) for a new VNF instance. When the NFVO receives the response from the VNFM with the identifier of the VNF instance, it sends the ***InstantiateVnfRequest*** to the VNFM to instantiate the VNF instance (clause 7.2.3 of [6]).

The VNFM's response procedure to the "VNF instantiation request" consists of three steps [5] and [6]:

- first it returns the "lifecycleOperationOccurrenceId",
- second it sends a "start" Lifecycle Change Notification,
- finally it sends the "result" Lifecycle Change Notification.

The VNF LCM principles listed for the instantiation of a VNF instance should be applied also to the design of the VNF software modification.

B.3.2 VNF Software Modification Initiated from the OSS/BSS

Figure B.1 presents a potential flow diagram for the VNF software modification operation initiated from the OSS/BSS.

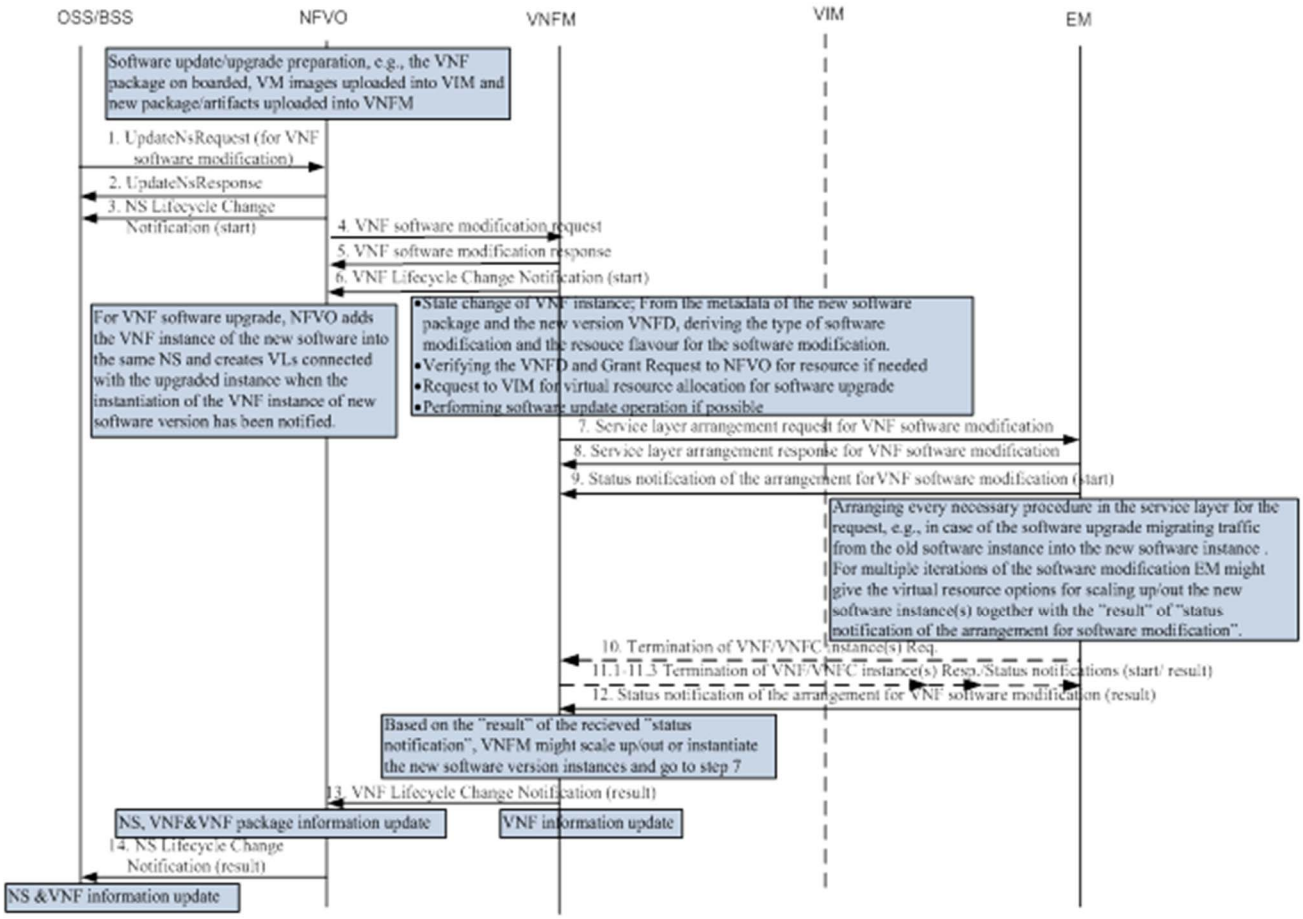


Figure B.1: Flow diagram for the operation of the VNF software modification initiated from the OSS/BSS

Before the initiation of the VNF software modification process, some preliminary operations need to be performed by the OSS/BSS, e.g. verification of the VNF package related to the VNF function, on-boarding the VNF package into NFVO, etc.

1) Once the preliminary operations are completed, the OSS/BSS initiates the VNF software modification toward the NFVO by an "Update NS operation" of the NS LCM management interface according to [5]. Note that a stateless VNF software upgrade could also be done by creating a new VNFFG and instantiating a VNF instance with the new version using the current LCM operations]:

   - In the "UpdateNsRequest" message the following option and information elements (IEs) need to be added:

     - Update type of "VNF software modification": The option of VNF software modification needs to be added to the supported list of update types in the updateType IE.

     - "ModifyVnfInstanceData" IE including (note that within a NS multiple VNF instances' software modification could be supported):

       - "Identifier(s) of the targeted VNF instance(s)": The IE is used to identify the targeted VNF instance(s) of the software modification (within an NS, software modification of multiple VNF instances could be supported by this request).

       - "Identifier(s) of VNF package (s)": The IE is used to identify the VNF package(s) for the VNF software modification (within an NS, software modification of multiple VNF instances could be supported by this request).

       - "Policies/rules" for the software modification (optional IE): The IE provides the corresponding policies/rules selected by the network operator for the software modification.

- "Instantiation Parameters": Parameters for instantiating instance(s) with the new software version (optional IE).

2) When the NFVO receives the "UpdateNsRequest" for the VNF software modification from the OSS/BSS, the NFVO returns the "lifecycleOperationOccurrenceId" by using "UpdateNsResponse" to the OSS/BSS, before triggering any further the operation. It also verifies the "request" including verifying the accessibility of the VNF packages used for the VNF software modification.

3) NFVO sends the "start" NS Lifecycle Change Notification. In case the verification in step 2 fails, the NFVO also sends the "result" NS Lifecycle Change Notification with the unsuccessful result (see step 14) and stops the modification process.

4) In case the verification of the "request" is successful, the NFVO triggers the procedure of the VNF software modification including:

   - Changing the state of the targeted VNF instance(s) to the "software modification" state, sending the *"VNF software modification request"* to the VNFMs.

   - The operation "VNF software modification request" needs to be added to the current VNF LCM. At least the following IEs (some additional IEs might need also) are needed for this request message:

     - Identifier of the (targeted/modified) VNF instance.

     - Identifier of the VNF package used for the VNF software modification, i.e. vnfdID.

     - Policies/rules selected by the network operator for the software modification (optional).

     - Instantiation parameters: Parameters for instantiating instance(s) with the new software version (optional).

5) When the VNFM receives the "VNF software modification request" from the NFVO, it derives the parameters from the received message such as the identifier of the targeted VNF instance (with the old software version), the identifier of the VNF package to be used for the VNF software modification, and the policies/rules selected for the software modification, or the instantiation parameters, etc. and performs the following procedures:

   - Verifies the "request" including verifying the access to the VNF packages/artefacts used for the VNF software modification.

   - Verifies the software version of the targeted VNF instance based on the VNF software modification metadata of the VNF package if the VNF package is applicable to it:

     - In case the previous verification results are "true", and if the type of the software modification is "VNF software upgrade", the VNFM creates a new VNF instance ID for the instance of the new software version.

   - Returns the "lifecycleOperationOccurrenceId" by using "VNF software modification response" to the NFVO, before triggering any the further procedure for the VNF software modification.

6) The VNFM sends the "start" VNF Lifecycle Change Notification to the NFVO. If any of the verifications in step 5 fails, after sending the "start" notification the VNFM sends the "result" VNF Lifecycle Change Notification to NFVO with "unsuccessful" result (see step 13) and stops the software modification process. The Lifecycle Change Notification might include the following parameters:

   - Identifier of the targeted VNF instance.

   - Type of the software modification.

   - Identifier of the VNF-instance to be instantiated with the new software.

In case the verification in step 5 is successful, the VNFM proceeds further with the procedure of the VNF software modification for the (targeted) VNF instance including:

- Changing the state of the targeted VNF instance to the "software modification" state.

- Based on the type of the VNF software modification defined in the metadata, the VNFM performs different procedures.

- For the modification type of VNF software upgrade, the VNFM instantiates a VNF instance with the new software version according to the following steps:

  - Determine the virtual resources needed or the flavor based on one of the follow ways:

    - If the message of "VNF software modification request" has given the instantiation parameters, the derivation procedure could be similar as that of current specified VNF instantiation.

    - Otherwise, the virtual resources are determined based on the received policies/rules in the message of "VNF software modification request", the VNF software modification metadata of the VNF package and VNFD.

NOTE: The metadata may include the selectable policies/rules and their corresponding values of additional virtual resources for the new software instance.

  - For the virtual resources needed, the VNFM requests the resource grant from the NFVO and virtual resources allocations from the VIM (note: the virtual resource granting interface between the VNFM and the NFVO might need to be modified to support the VNF software modification operation, e.g. LCM operation type, VNF instance identifiers of the old version and the new version, etc. The NFVO might check the resource quotas for the targeted VNFs in order to update them in case the resource quotas are not enough for granting extra resources for the software modification process).

  - Instantiating a VNF instance with the new software version (note: VNFM might use VNF life change notification for notifying the NFVO of the newly instantiated VNF instance. The NFVO might add the new VNF instance of the new software version to the NS of the modified instance and create the VLs to interconnect the new VNF instance of the new software version. Those steps are not shown in figure B.1).

- For the modification type of VNF software update, an initial determination is made based on the metadata defined e.g. the identifiers (e.g. versions, name, and storage addresses) of the software images modified and their corresponding identifiers of executable files or scripts used for updating. Accordingly, the VNFM identifies the VNFC instance(s) to be updated and the corresponding executable files or scripts to be used for updating. Then the VNFM further identifies the virtual container(s) (e.g. VMs) hosting the VNFC instance(s). If the identified virtual containers have external interfaces accessible by the VNFM it might perform the updates by uploading and executing the update files or scripts in the virtual container(s). Otherwise, the VNFM requests the EM to perform the update (see step 7).

- For the modification type of VNFC software upgrade, VNFM instantiates the VNFC instance(s) with the new software version in the targeted VNF instance (old software version) via the following steps:

  - The VNFM identifies the VNFC instance(s) to be upgraded based on the metadata of the VNF package which defines applicability of the VNFC software upgrade by the identifier(s) of the modified software image(s), and VNF version numbers. It also provides the identifier(s) of the corresponding software image(s) to be used for the modification.

  - The VNFM determines the virtual resources needed based on one of the following ways:

    - If the message of "VNF software modification request" has given the instantiation parameters, the derivation procedure could be similar to other LCM operations. Note that the current LCM does not support instantiation at the VNFC level, but clause 6.2.1.2 requires such support.

    - Otherwise, determine the virtual resources needed based on the received policies/rules in the message of "VNF software modification request", the metadata of the VNF software modification, and the VDU/VNFD in the new VNF package.

  - For the virtual resources needed requesting the resource grant from the NFVO and the allocation from the VIM.

  - Instantiating a VNFC instance(s) with the new software version within the targeted VNF instance (old software version).

7) In previous steps, instance(s) for the software modification might have been instantiated (for VNF or VNFC software upgrade) or identified (for VNF update). Since the VNFM does not have any VNF application level information and it is not aware of the VNF internal architecture it is not able to perform further operations needed for the software modification procedure, e.g. traffic migration from an old software instance to a new software instance. There might be a need for the EM to manage the traffic migration for the software modification. Therefore the VNFM needs to deliver to the EM the information of instance(s), virtual resources and options of policies/rules, etc., to continue with the VNF software modification procedure for service layer arrangements. Since the current specification of the reference point between the VNFM and the EM does not have such an operation, a new message needs to be added to the interface from the VNFM to the EM such as "Service layer arrangement for VNF software modification request". The message might include the following information:

   - Identifier of the VNF instance being modified.
   - Identifier of the VNF package used for the software modification.
   - The type of the VNF software modification.
   - Information elements for the newly instantiated instance(s) (of the new software version) or for the VNFC instance(s) to be updated including the images used for the update (in case of software update).
   - The options of policies/rules selected by the network operator for the software modification (optional).

8) The EM sends the "Service layer arrangement for VNF software modification response" as a response to the "Service layer arrangement for VNF software modification request". The message of "Service layer arrangement VNF software modification response" might include:

   - Identifier of the VNF instance being modified and identifier(s) of the instance(s) of the new software version.
   - "SoftwareModificationOccurrenceId" (for identifying the request to the EM because it takes time to migrate the traffic between instances of the old and the new software versions).

9) After sending the "Service layer arrangement response for VNF software modification", the EM sends a "start" with the "Status notification of the arrangement for VNF software modification" for notifying the VNFM (about "Service layer arrangement request for VNF software modification"). Then, it makes all necessary arrangements for the request procedure at the service layer, e.g. in case of the software upgrade, it migrates the traffic from the old software instance to the new software instance.

10) The EM may initiate the termination of VNF or VNFC instance(s) with the old software if this(these instance(s) has (have) completed the service migration to the instance(s) with the new software version.

11) VNFM may send the response and the notifications of "start" and "result" for the termination of the requested VNF or VNFC instance(s).

12) The EM sends the "result" notification with the "Status notification of the service layer arrangement for VNF software modification" when the current iteration of the software modification completes. For multiple iterations of the software modification if additional resources are needed for new software instances (depending on policies/rules), then in each iteration the EM might request the virtual resource options for scaling up/out/instantiating the new software instance(s) for the next iteration together with the "result" of "status notification of the software modification" of the current iteration. For VNFC software upgrade, there are two options:

a) Embedding the scaling up/out/instantiating request in the notification,

b) Define a new operation of "VNFC instantiation request".

13) Based on the "result" of the "status notification of the software modification", the VNFM decides if it should proceed to step 13 (i.e. final result) or request the grant and allocation of virtual resources for the next iteration of the software modification and go to step 7.

14) When the VNFM receives the final "result" notification for the completion of the software modification in the "Status notification of the arrangement for VNF software modification", it triggers the VNF information update and sends the "result" notification (for the "VNF software modification request") with the VNF Lifecycle Change Notification to the NFVO.

15) When the NFVO receives the "result" "VNF Lifecycle Change Notification" from the VNFM(s) to which the NFVO has sent a "VNF software modification request", it triggers the VNF information and VNF package state update and sends the "result" NS Lifecycle Change Notification for the "*NS Update Request*" to the OSS/BSS.

In steps 6 through 12, another alternative is that the VNFM does not interpret the virtual resource options from the metadata of VNF software modification. Instead it passes on the VNF software modification request to the EM in step 7. The EM interprets the metadata of the VNF software modification and manages its execution, which may include (in addition to the traffic migration) requesting the instantiation/termination of VNFC instances toward the VNFM if necessary in multiple iterations as appropriate at the service layer and according to the metadata for the VNF software modification. As a result, in step 12 the EM only returns the result of the service layer arrangements to the VNFM when it is finalized.

B.3.3 VNF Software Modification Initiated from the EM

Figure B.2 presents the potential flow diagram for the VNF software modification operation initiated from EM and the steps are similar to steps 7-12 in figure B.1 except for the difference in the initiator.

Before the initiation of the VNF software modification process, some preliminary operations need to be completed, e.g. verification of the VNF package including the VNF function, on-boarding the new VNF package by the OSS/BSS to the NFVO, etc. The OSS and the EM need to coordinate outside the NFV scope the preparation for the VNF software modification:

1) After the new VNF package has been on-boarded into NFV environment, the EM initiates the operation of VNF software modification towards the VNFM by the message of "VNF software modification request" which has been discussed in step 4 of figure B.1.

2) The VNFM responds to the EM concerning "VNF software modification request".

3) For steps 3 to 8, the operations of VNFM and EM are the exact same as those in steps 7 to 12 in figure B.1.

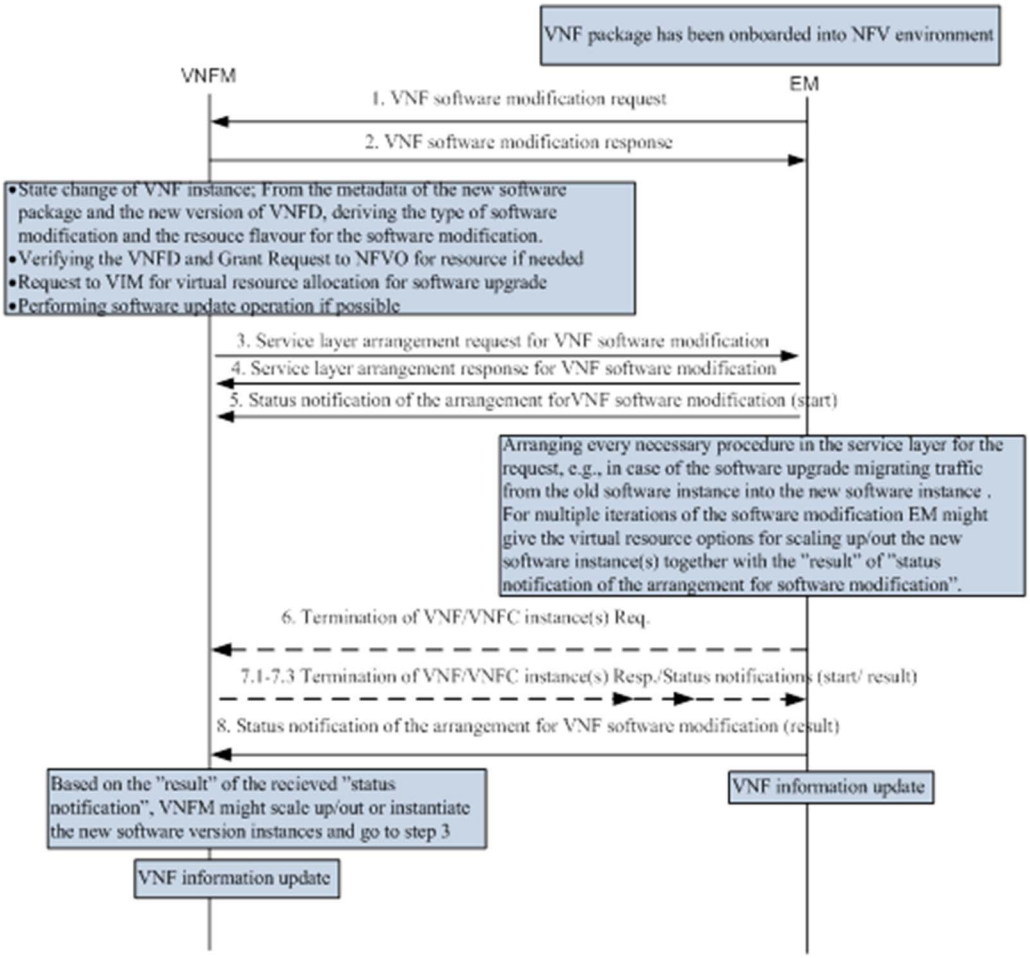


Figure B.2: Flow diagram for the operation of the VNF software modification initiated from EM

Annex C (informative): NFVI Software Modification Flows

C.1 Exemplary Coordination of NFVI Software Modifications

The general coordination flow for NFVI software modification could look like the example illustrated in figure C.1. The diagram shows the NFVI Software Modification Manager function and a VNF-level Manager entity, which could represent, for example, the VNFM, the EM, the OSS/BSS or possibly another functional block responsible for the coordination on behalf of the hosted VNF(s) and MANO.

According to figure C.1 the exemplary flow is as follows:

1) As prerequisite, the VNF-level Managers inform the NFVI Software Modification Manager whether coordination of NFVI software modifications is necessary for a virtualised resource (VR) or a VR group (such as an anti-affinity group) as well as the applicable constraints. This may be subscription based or part of the VR/VR group creation process.

2) When an NFVI software modification is requested (2.a), the NFVI Software Modification Manager identifies the impacted VRs and VR groups and the order in which the software modifications of NFVI resources can be performed considering the constraints imposed by these impacted VRs and VR groups (2.b).

3) The NFVI Software Modification Manager notifies the VNF-level Manager that an NFVI software modification procedure is about to start, which may impact the VR group (3.a) or the VR (3.c) for which it coordinates the process. For a VR such as a virtual machine, it may further specify whether the VR is going to be live-migrated or shut down. At the same time as the notification is sent, the NFVI Software Modification Manager starts a timer with the determined lead time. The lead time may be determined, for example, as the maximum lead time imposed by the constraints. The NFVI Software Modification Manager waits at most the lead time before proceeding with the software modifications.

4) The VNF-level Manager initiates the necessary preparations for the potential disruption(s) caused by the NFVI software modification(s). In case of a VR group (4.a) this may mean for example a scale out to increase the VNF's redundancy, or switching the active role from the impacted VNF to its geo-redundant pair. In case of a VR (4.c) this may also mean a switch over of the active role of the VNFC instance hosted on the impacted VR, or redirecting part or the entire traffic to another set of redundant VNFC instances. Once the preparations are completed the VNF-level Manager may inform the NFVI Software Modification Manager about its readiness (4.b, 4.d), however this is not necessary. Such a response can be used by the NFVI Software Modification Manager to cancel the lead timer.

5) Once the lead time expires or all responses have been received the NFVI Software Modification Manager proceeds with the software modification of the NFVI resources (5.a) or resource (5.b) as applicable. In case of a VR group (5.a), this may mean multiple iterations until all NFVI resources supporting the VR groups have been modified as requested. Doing so the NFVI Software Modification Manager needs to respect the applicable constraints. For an anti-affinity group, for example, the VRs impacted simultaneously in the group may not exceed the maximum number specified for the anti-affinity group and at least the minimum number of VRs may need to be kept available at all times.

6) Once the NFVI software modification procedure is completed for all required resources, the NFVI Software Modification Manager notifies the VNF-level Manager about the completion of the process (6.a, 6.c), which in turn can initiate any wrap-up actions (6.b, 6.d), such as reversing the configuration changes made in preparation to the impact or workload rebalancing.

7) Finally, the NFVI Software Modification Manager sends a notification that the requested NFVI software modification has been completed.

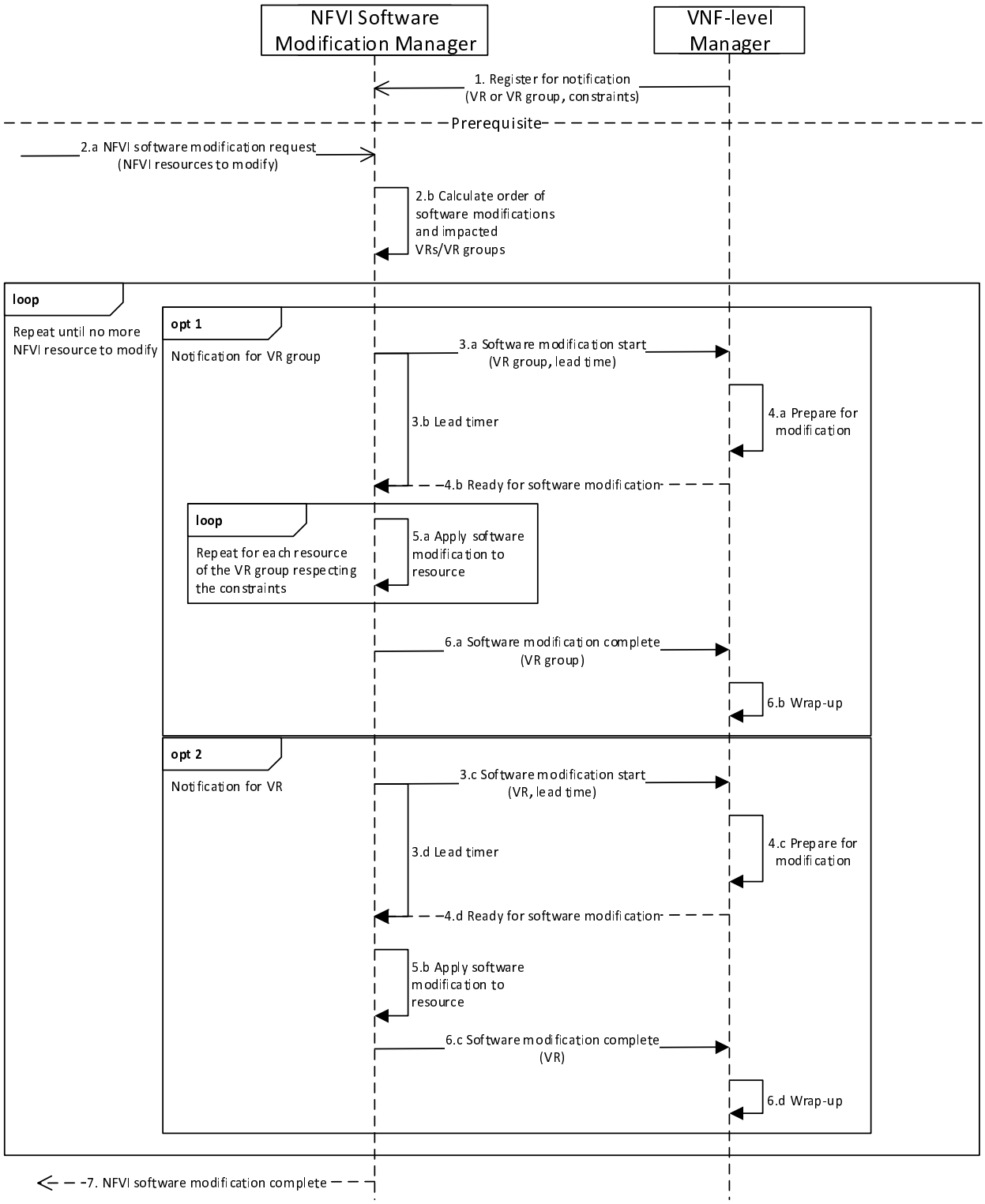


Figure C.1: Exemplary coordination flow for NFVI software modification

C.2 Illustrative Example of NFVI Resource Software Modification

The sequence diagram of figure C.2 gives an overview of how these NFVI resource software modifications could be handled. It is the elaboration of step 5 of the exemplary coordination flow using as example the upgrade of a compute host (CompHost), which impacts a hosted VR, namely a VM. Note that this is an illustrative example demonstrating the interaction that could take place for this process.

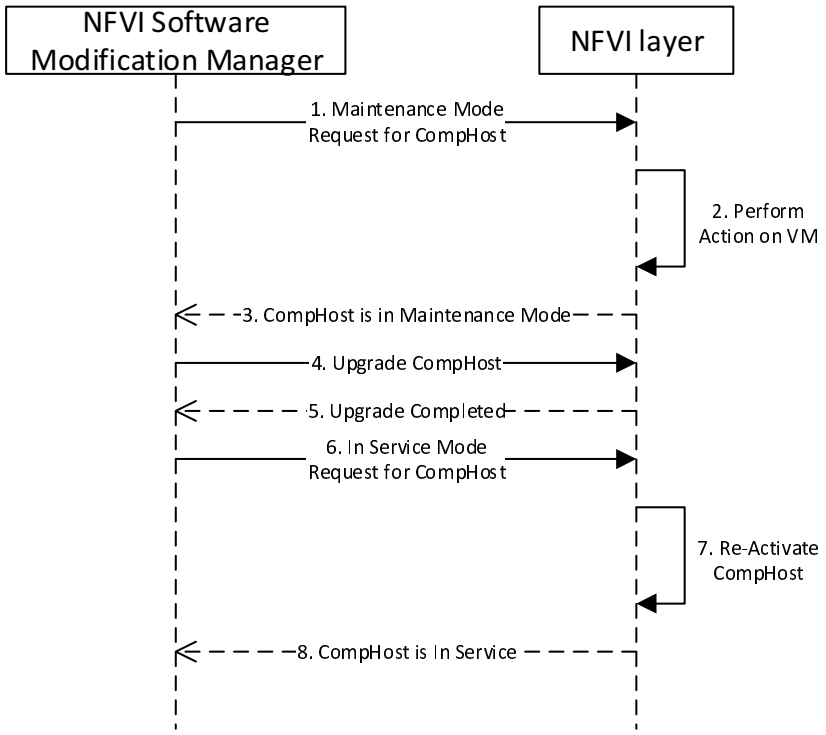


Figure C.2: Example Handling of NFVI Resource Software Modification

1) NFVI Software Modification Manager requests that a compute host (e.g. CompHost) is put in maintenance mode. This means that some preparations need to be done so that the resource can be taken out of service.

2) Depending on the applicable policies, an appropriate action is performed on the impacted VM: The VM may be shut down or it might be migrated. For the latter case, it is assumed that a compatible host is available for migrating the impacted VM.

3) Once the action is completed and the resource does not serve any VM anymore, it is in maintenance mode and the NFVI Software Modification Manager is informed.

4) The NFVI Software Modification Manager initiates the upgrade of the CompHost resource. The upgrade method is not essential as the resource is out of service. The duration of the upgrade method may be of importance.

5) The NFVI Software Modification Manager is informed about the completion of the resource upgrade.

6) Since the upgrade has finished the NFVI Software Modification Manager requests to take the CompHost resource back into service.

7) The actions necessary to bring the resource back into service are performed.

8) NFVI Software Modification Manager request is confirmed that the CompHost resource is back in service again.

The diagram of figure C.2 focuses on the upgrade of an individual resource and the same process needs to be repeated for each NFVI resource or group of resources to be upgraded as it has been indicated in the step 5 of the exemplary coordination flow. Some of these may be performed in parallel, while other may need to be in sequence.

Annex D (informative):

Authors & Contributors

The following people have contributed to the present document:

Rapporteur:

Percy S. Tarapore, AT&T

Contributors:

Stefan Arntzen, Huawei

Bruno Chatras, Orange

LAC Chidung, Orange

Geoff Halprin, Huawei

Tommy Lindgren, Ericsson

Shaoji Ni, Huawei

Maria Toeroe, Ericsson

History

Document history		
V3.1.1	February 2018	Publication