ETSI GR NFV-REL 011 V4.1.1 (2020-11)

Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Report on NFV-MANO 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
RGR/NFV-REL011ed411

Keywords
availability, MANO, NFV, service

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

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

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

Modal verbs terminology

In the present document "**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 collects and describes use cases for modifying NFV-MANO software while maintaining service availability and continuity, irrespective of the technologies being used to deploy this software. As a result, detailed recommendations for the requirements of the NFV-MANO software modification process are derived.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

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

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

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

[i.1] ETSI GS NFV-MAN 001 (V1.1.1): "Network Functions Virtualisation (NFV); Management and Orchestration".

[i.2] ETSI GS NFV-IFA 031 (V3.3.1): "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Requirements and interfaces specification for management of NFV-MANO".

[i.3] ETSI GS NFV-IFA 009 (V1.1.1): "Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options".

[i.4] ETSI GR NFV 003 (V1.5.1): "Network Functions Virtualisation (NFV); Terminology for main concepts in NFV".

[i.5] ETSI GR NFV-IFA 029 (V3.3.1): "Network Functions Virtualisation (NFV) Release 3; Architecture; Report on the Enhancements of the NFV architecture towards "Cloud-native" and "PaaS"".

[i.6] M. Toeroe and F. Tam (Eds.): "Service Availability: Principles and Practice", J. Wiley (2012).

[i.7] ETSI GS NFV-IFA 010 (V3.3.1): "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Functional requirements specification".

[i.8] ETSI GR NFV-REL 007 (V1.1.2): "Network Functions Virtualisation (NFV); Reliability; Report on the resilience of NFV-MANO critical capabilities".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GR NFV 003 [i.4] and the following apply:

NOTE 1: A term defined in the present document takes precedence over the definition of the same term, if any, in ETSI GR NFV 003 [i.4].

NOTE 2: In the scope of the present document, for any entity which is part of the NFV architectural framework, the term "entity" is used.

fallback: recovery procedure of forcefully restoring an entity using a specific restoration point

NOTE 1: A successful fallback allows the entity to continue normal operation.

NOTE 2: A fallback may restore simultaneously more than one entity depending on the content of the restoration point.

NOTE 3: Depending on the content of the restoration point, the fallback operation does not ensure that the NFV-MANO functional entity continues to provide its services during the fallback procedure.

redundancy unit: part of an NFV-MANO functional entity which can have the option to be deployed redundantly to improve service availability

NOTE 1: When the redundancy unit(s) of an NFV-MANO functional entity is(are) deployed redundantly, they together with the rest of the NFV-MANO functional entity form a single instance of the NFV-MANO functional entity and, accordingly, act together to deliver better availability of the services provided by the NFV-MANO functional entity.

NOTE 2: The redundancy unit(s) of an NFV-MANO functional entity is(are) visible to the operator who takes the decision to deploy it(them) redundantly, or not, considering the supplier supported features.

restoration point: backup containing all data that is necessary to revert to an entity's software and state this entity had at the time the backup was captured

NOTE 1: A restoration point may contain data to restore more than one entity.

NOTE 2: A restoration point may be created by any techniques for capturing the software and the actual state of the entity. This can be a snapshot.

retry: graceful recovery procedure of re-entering the flow of the software modification process after the last successfully executed step prior to the error/failure occurrence

NOTE: A successful retry allows the continuation of the software modification process in progress as if no error/failure had occurred.

software modification: main part of the software modification process deploying the target software version

NOTE: Depending on the type of changes in the software deployed by a software modification, this part (or a subset) may be referred to as software upgrade or software update as defined in ETSI GR NFV 003 [i.4].

software modification process: process of replacing or modifying the software of one or more deployed functional entities

NOTE 1: The goal of the software modification process can be bug fixes or enhancements with or without adding, modifying or removing functionalities, interfaces or protocols.

NOTE 2: The software modification process may have two parts: software modification - (mandatory) main part deploying the intended target software version; and, if needed, software rollback - (conditional) recovery part restoring the software to the version deployed at the beginning of the software modification process if needed.

software rollback: graceful recovery procedure reversing a software modification

NOTE 1: A software rollback is undoing step by step the operations executed as part of the software modification. It can be triggered even after an operation resulted in an error.

NOTE 2: A software rollback restores the software to the version deployed at the beginning of the software modification, thereby ensuring that the involved NFV-MANO functional entities (i.e. targeted and impacted NFV-MANO functional entities) are in a consistent state (state synchronization might happen as part of this process).

NOTE 3: The intention is that during a software rollback, the NFV-MANO functional entity continues to provide its services.

3.2 Symbols

Void.

3.3 Abbreviations

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

API	APplication Interface
CIS	Container Infrastructure Service
CISM	Container Infrastructure Service Management
EM	Element Manage
GS	Group Specification
LCM	Life Cycle Management
MCIO	Managed Container Infrastructure Object
NFVI	Network Functions Virtualisation Infrastructure
NFVO	Network Function Virtualisation Orchestrator
NS	Network Service
NSD	Network Service Descriptor
NSO	Network Service Orchestration
OS	Operating System
OSS	Operation Support System
RO	Resource Orchestration
SDN	Software Defined Networking
SMM	Software Modification Manager
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VNFC	Virtual Network Function Component
VNFM	Virtual Network Function Manager
VRM	Virtual Resource Management
WAN	Wide Area Network
WIM	WAN Infrastructure Manager

4 Overview of NFV-MANO software modification

4.1 NFV-MANO software modification process

4.1.1 Introduction

As NFV-MANO is composed of different NFV-MANO functional entities, the software modification process consists of modifying the software of one or more of such functional entities. Two approaches for such modification are possible: online modification and offline modification. The online approach allows for in-service modification of the NFV-MANO functional entity(ies) software. As for the offline approach, it requires some downtime during the software modification. As service continuity is a prerequisite (see ETSI GR NFV-REL 007 [i.8]), the present document will only handle the online approach for NFV-MANO software modification process.

In addition to the online/offline distinction, the nature of the NFV-MANO functional entity(ies) has to be taken into consideration. Actually, the NFV-MANO functional entity(ies) can be deployed as VNF(s): in such case, its(their) software modification procedure is similar to the one considered for, e.g. network function VNFs. Another case is illustrated by NFV-MANO functional entity(ies) not deployed as VNF(s). Therefore, only the latter case will be discussed in the present document.

NOTE: Not being deployed as a VNF does not restrict that the NFV-MANO functional entity cannot be deployed as a virtualised entity; it only implies that it is not managed as a VNF.

Internal support of seamless software modification means that the creation of new NFV-MANO functional entity(ies) is not needed thanks to mechanisms embedded in the NFV-MANO functional entity(ies), e.g. redundant elements used for hosting the new software version. As such process is internal to the NFV-MANO functional entity(ies), the change of interface endpoints to the post-modification NFV-MANO functional entity(ies) is not necessary. Two approaches are thus possible: with or without internal support of seamless software modification. Both cases will also be handled in the present document.

Finally, it is noteworthy that two situations can occur: NFV-MANO functional entity running without redundancy, or cluster-like NFV-MANO functional entity based on an N+K architecture. In the latter case, the redundancy can be deployed locally in a single site or geographically across multiple sites.

4.1.2 Main concepts

As described in clause 4.1.1, the **software modification process** is the process of modifying or replacing the software of one or more functional entities deployed in the NFV-MANO, while the NFV-MANO continues to provide its services.

The goal of the software modification process is usually to introduce bug fixes or enhancements to the system or its subsystems with, or without, adding, modifying or removing functionalities, interfaces or protocols. To achieve these goals, the software modification process starts out with its main part - the **software modification**. During this software modification part, the software modification process takes actions progressively moving forward on the path towards deploying the intended or target software version. Depending on the type of changes in the software compared to the currently deployed software version, this part (or a subset) may be referred to as software upgrade or software update as defined in ETSI GR NFV 003 [i.4].

The software modification process itself is managed by an entity referred to as the Software Modification Manager (SMM). The SMM is responsible to coordinate the software modification process at the NFV-MANO system level by triggering the appropriate actions on potentially multiple different entities. These entities include the target NFV-MANO functional entities, but also other NFV-MANO functional entities and impacted entities, which might be managed by, or interacting with, the target NFV-MANO functional entities. Thus, the role of SMM can be fulfilled by an administrator or OSS, or it could be a new functionality offered by NFV-MANO. Considering this last option, it is important to point out the difference between the SMM and the internal support of an NFV-MANO functional entity for software modification.

The internal software modification support of an NFV-MANO functional entity is provided by its supplier and it manages the software modification of this NFV-MANO functional entity within its scope. The internal software modification support of an NFV-MANO functional entity is fully aware of the internal structure of this NFV-MANO functional entity. However, for the SMM, this internal software modification support is visible only as a single operation, which it can trigger on this NFV-MANO functional entity as part of the overall software modification process that can include other entities as well. When using the internal software modification, the SMM does not need to be aware of the internal structure of the NFV-MANO functional entity.

If the software modification of an NFV-MANO functional entity needs to be supported by the SMM, the SMM needs to be aware of parts of this NFV-MANO functional entity providing redundancy, that is, the **redundancy units** of the NFV-MANO functional entity, when the SMM needs to use this redundancy in the software modification process.

The assumption is that when redundancy units of an NFV-MANO functional entity are deployed redundantly, they act together as a single instance of the NFV-MANO functional entity to ensure better availability of the services provided by the NFV-MANO functional entity. The SMM can take advantage of such redundant deployment to improve service availability during the software modification process.

The actions the SMM typically orchestrates during the main software modification part are:

1) Preparation for the software modification, that is, performing different checks and operations to ensure as much as possible that the software modification will be successful, or if the software modification fails, then to ensure that a recovery is possible. The preparation includes checking and reserving resources, selecting or creating a **restoration point**, collecting information, starting special logs, etc.

2) One or more lifecycle management operations on the target NFV-MANO functional entities, which are necessary to deploy the target software. This can include triggering the internal software modification support of the target NFV-MANO functional entity, or the instantiation and termination of the target NFV-MANO functional entities or their redundancy units. Additional actions can be performed as necessary to manage the services provided by the target entities to their consumers.

3) Testing at different levels that the newly deployed software performs as expected and the software modification was successful. An NFV-MANO functional entity with internal support for software modification can perform testing within its scope, while the SMM is responsible to ensure that the software modification was successful at the system level.

4) Committing the changes introduced by the successful software modification.

It is envisioned that the details of the software modification process are provided to SMM by the operator as an execution plan, which is a workflow that the SMM can execute to achieve the intended software modification.

Unfortunately, there is no hundred percent guarantee that the software modification can be accomplished successfully, e.g. an action can fail persistently even after retries, or the target NFV-MANO functional entities with the newly deployed software may not pass all the required tests at the system level. This means that a software modification process also needs to include a conditional recovery part - the **software rollback**. A software rollback reverts the changes made by the software modification by undoing step by step the operations executed as part of the software modification. As a result, a software rollback restores the software to the source versions deployed at the beginning of the software modification, thereby ensuring that the involved entities (i.e. targeted and impacted entities) are in a consistent state (state synchronization process might happen as part of this process). The software rollback is a graceful recovery procedure as the goal is that the target NFV-MANO functional entities continue to provide their services during the procedure.

The software rollback part is also managed by the SMM using the execution plan in the same way as in the software modification part. It can also use information collected as part of the preparation to the software modification as well as during its execution, e.g. special logs supporting the software rollback.

The recovery part consists of actions similar to the software modification part:

1) Preparation for the software rollback, that is, performing different checks and collecting information to ensure that the software rollback is possible and will be successful.

2) One or more lifecycle management operations on target NFV-MANO functional entities necessary to revert them to their source software that was deployed at the beginning of the software modification. Additional actions can be performed as necessary to manage the services provided by the targeted entities to their consumers.

3) Testing that the re-deployed source software performs as before and the software rollback was successful.

4) Committing the changes made by the successful software rollback.

A software rollback can be triggered at any of the actions described in the software modification part which was performed, even after an erroneous operation, provided the system and specifically the target NFV-MANO functional entity are in a known state, i.e. a state from which graceful recovery is possible.

Depending on the target NFV-MANO functional entity and/or the action at which the problem occurred, it might be still possible to complete the software modification by re-trying the erroneous operation, i.e. re-entering the flow of the software modification process after the last successfully executed step prior to the error/failure occurrence. Moreover, **retry** can also be an option after an erroneous operation during a software rollback, which can avoid the need for a more drastic recovery procedure.

If neither retry nor rollback is possible, for example, because the system is in an unknown state, the last resort is a forceful recovery procedure, i.e. a **fallback**. The prerequisite of a fallback is that there is a restoration point available to which the target NFV-MANO functional entity and possibly other entities can be reverted so that normal operation can be continued.

The fallback procedure is not part of the software modification process, as failures or any other situation requiring such measures can occur outside of the scope of the software modifications process. However, the SMM can trigger a fallback to a given restoration point as a recovery procedure from a failed software modification process.

This **restoration point** is a backup containing all the data necessary to revert the software and the state of an entity (NFV-MANO functional entities and other entities) that this entity had at the time the backup was captured. The same restoration point can contain data to restore more than one entity, e.g. the target NFV-MANO functional entity, impacted entity, etc. Restoration points may be created by using different techniques for capturing the software and the actual state of entities such as a snapshot.

Figure 4.1.2-1 illustrates the relation between the different concepts related to the software modification process, while the next clause 4.1.3 elaborates further on the different recovery procedures introduced in this clause.

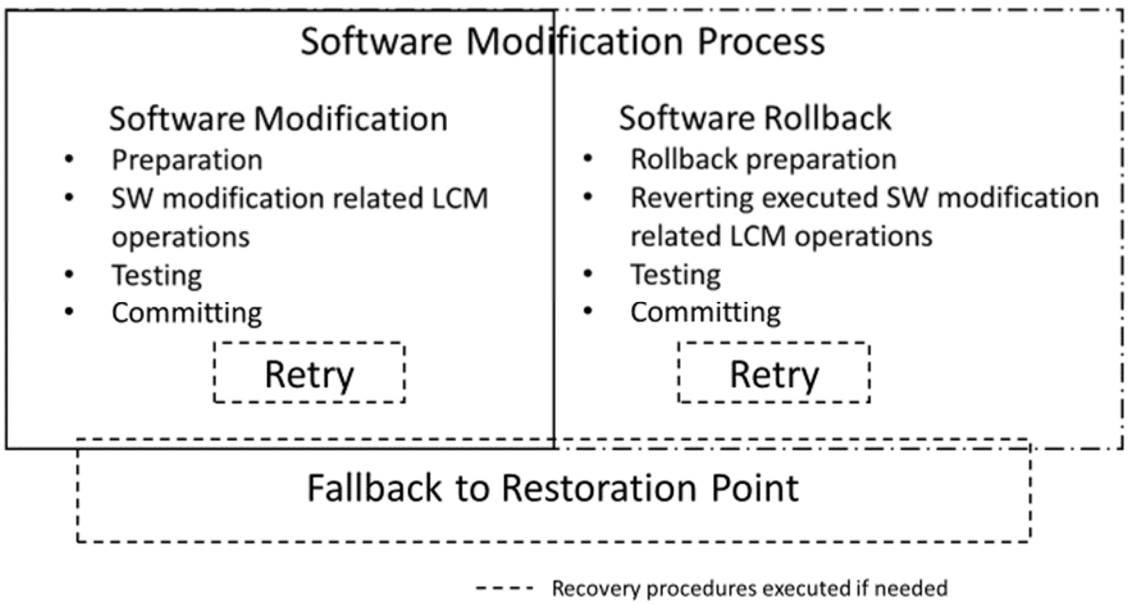

Figure 4.1.2-1: Interrelation of the software modification related concepts

4.1.3 Recovery procedures

From a resiliency perspective, the goal of the NFV-MANO software modification process is to modify the software of one or more NFV-MANO functional entities without impacting the services the NFV-MANO functional entities provide. Unfortunately, failures can occur at any time in the system including during, and as part of, the NFV-MANO software modification process. This clause introduces the different recovery procedures that can be applied during the software modification process.

M. Toeroe et al. [i.6] identify four artefacts determining the system state at any time. They apply to the NFV-MANO as follows:

1) The NFV-MANO configuration in terms of the NFV-MANO functional entities and their components.

2) The software executable associated with each of the NFV-MANO functional entity and their components.

3) The runtime status of each NFV-MANO functional entity and their components with respect to the services they provide.

4) The state of the service instances provided by the different NFV-MANO functional entities and their components.

At normal operation, when the NFV-MANO functional entities provide their service to the NFV system, typically only the last two artefacts (items 3 and 4) change, i.e. the runtime status of the NFV-MANO functional entities and the state of the service instances they provide. While at software modification the first two also change. Namely, the software of NFV-MANO functional entities is modified through manipulating the NFV-MANO configuration in such a way that it changes the software executable associated with the different NFV-MANO functional entities and their components, e.g. adding new NFV-MANO functional entities running the new software and removing old ones running the old software.

Accordingly, changes in items 1 and 2 together can be referred to as "software modification changes" and changes in items 3 and 4 referred to as "NFV-MANO service changes". The software modification process can also be defined as progressing with the software modification changes until they are successfully completed while not impacting the progress of the NFV-MANO service changes reflecting the services provided by the NFV-MANO. This is shown in chart A: Success of figure 4.1.3-1.

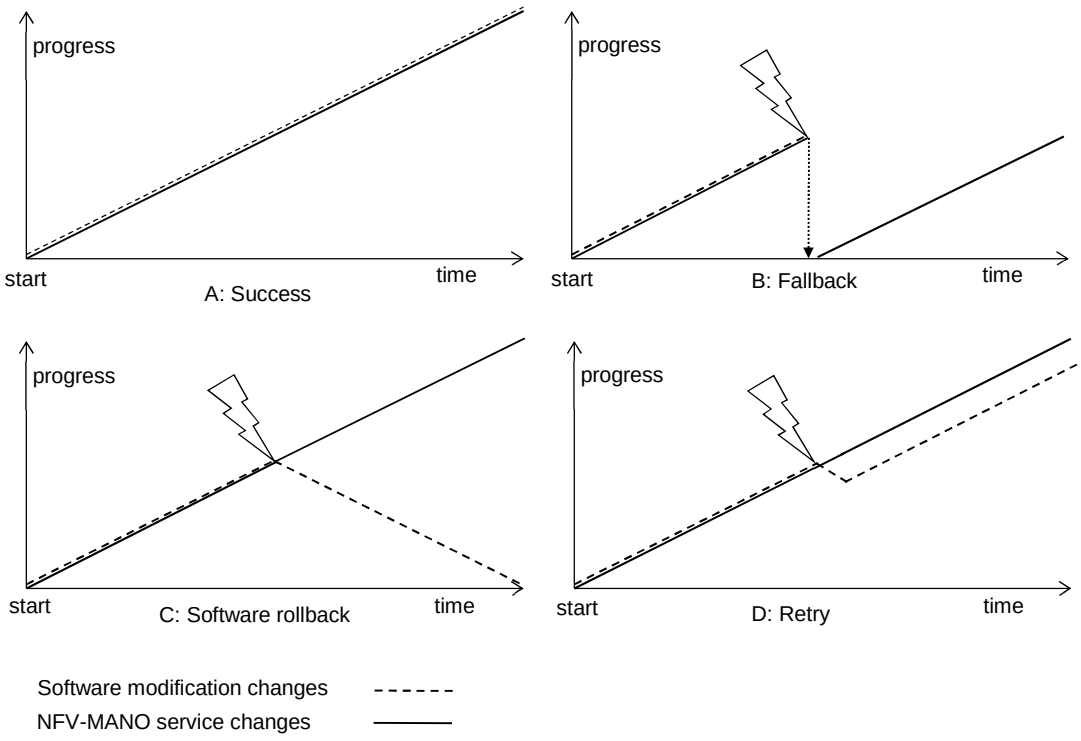

Figure 4.1.3-1: Fallback, rollback and retry of the software modification process

That is, the NFV-MANO software modification process starts out with a certain configuration of the NFV-MANO functional entities (and their components), which are associated with certain software executables. This configuration of NFV-MANO functional entities (and their components) and the associated software executable s change as the software modification process proceeds and the software modification changes are deployed and progress to complete the software modification procedure, which is shown in figure 4.1.3-1 by the dashed line.

At the same time (i.e. at the moment the software modification process starts) and independent of the software modification process, the NFV-MANO functional entities (and their components) have a given status and provide certain service instances in the NFV system as part of their normal operation. As the NFV-MANO functional entities perform their normal operation even during the software modification process, the NFV-MANO services also progress with their changes ideally without any disturbance from the software modification process, which is shown by the solid line in figure 4.1.3-1.

Compared to this successful case of the software modification process, charts B-D show cases when the NFV-MANO software modification process fails. In case of a failure, the goal is at least to be able to restore the status of the NFV-MANO functional entities and their service state to what they started out with at the beginning of the software modification process. This case is shown in chart B: Fallback of figure 4.1.3-1.

In this case, at start the status of NFV-MANO functional entities and the NFV-MANO service state information are stored as a backup - i.e. a restoration point is created - and when the failure occurs, the NFV-MANO functional entities are restarted from this backup. As a result, the status of NFV-MANO functional entities and the states of service instances provided by these NFV-MANO functional entities are also restored to where they were at the start of the software modification process. That is, the NFV-MANO service changes are back at the level they were at start, which is shown by the solid line dropping to the level it was at start and starting to progress from this level again.

Fallback is the procedure of restoring from the backup the configuration of NFV-MANO functional entities and associated software executables at start and removing all the software modification changes. After the failure, it is advisable to wait with the re-initiation of the software modification process itself until it is better understood how to avoid the failure to repeat again. As the software modification changes do not restart after handling the failure, there is no dashed line after the fallback.

The fallback procedure means that all the NFV-MANO service changes that have been performed from the start of the software modification process to the failure are lost, which could mean a rather significant service impact. For example, if the NFV-MANO has deployed a new NS after the start of the software modification process, since this NS instance is not part of the state information stored at the start in the backup, it cannot be restored from this backup, which could cause discrepancies and/or impact not only on the NFV-MANO services, but also on the NS instance.

NOTE: To avoid such situations, it is possible that a system will not allow certain LCM operations during a software modification process, and/or other measures are taken to enable roll-forward to restore such changes after a fallback.

A more graceful handling of failures is intended by the procedure shown in chart C: Software rollback of figure 4.1.3-1. Ideally, if a failure happens during the software modification process which prevents its completion successfully, it is still possible to isolate the problem and undo step by step the software modification changes that were deployed from the beginning of the process, while the NFV-MANO functional entities can continue to operate normally and provide their services. For example, if the new NFV-MANO functional entity running the new software does not pass the tests verifying its correct behaviour, the software modification cannot proceed and the new faulty NFV-MANO functional entity needs to be removed, while the rest of the system operates normally rather than restoring everything from the restoration point created at the start.

Accordingly, chart C of figure 4.1.3-1. shows that, after the failure, the NFV-MANO service changes shown by the solid line continue to progress as in the successful case without any impact. As for the software modification changes, they are gradually undone, i.e. regressed back to the state they were at the start, that is, to the level where no software modification changes were applied yet, as shown by the dashed line. Thus, the software rollback procedure typically has no impact on the services provided by the NFV-MANO functional entities.

In certain cases, it is also possible that the problem causing the failure of the software modification process is temporary. For example, an operation issued as part of the software modification process, such as taking a VM snapshot, does not succeed and the operation itself (i.e. snapshot) is rolled back. Obviously, in this case the software modification process cannot proceed without a completed snapshot. Nevertheless, it might not be desirable to roll back the entire software modification process because of this problem. Chart D of figure 4.1.3-1 presents the retry procedure. In this case, as a result of its failure, only a single operation of the software modification process is undone by rolling back the operation itself. This operation rollback could be performed automatically by the manager controlling the operation. Thus, during this operation rollback, the software modification process makes a small step back to the state at the beginning of the operation which is rolled back as shown in chart D by the dashed line. Meanwhile, the NFV-MANO functional entities continue to provide their services, so the NFV-MANO service changes progress as normal as shown by the solid line.

If the failure, which caused the operation rollback is deemed temporary (e.g. the error code received suggests a retry), the operation can be retried and, if it is successful, the software modification process can continue with its changes as normal although with a slight delay.

These different failure handling procedures of the software modification process typically apply to different failures. Sometimes, it is clear which one needs to be applied after a failure; other times, this is not clear. In the latter case, an escalation procedure can be considered which starts by applying the retry procedure first, and if that fails, then escalates to a software rollback, and if that also fails, then to a fallback. Note that this means that even though the creation of the restoration point is only mentioned in the case of the fallback procedure, it is necessary in all cases for the escalation procedure to work. More details on the different procedures will be provided in the subsequent clauses.

4.2 Architecture addressed by the NFV-MANO software modification

4.2.1 Architectural refinement of NFV-MANO functional blocks

The NFV-MANO architectural framework has been defined in the ETSI GS NFV-MAN 001 [i.1] specification as the following NFV-MANO functional blocks:

- NFV Orchestrator (NFVO);
- VNF Manager (VNFM); and
- Virtualised Infrastructure Manager (VIM).

Later on, Release 3 further specified the WAN Infrastructure Manager (WIM) as part of the NFV-MANO, of which its functional requirements are defined in ETSI GS NFV-IFA 010 [i.7]. The ETSI GS NFV-IFA 031 [i.2] specifies the framework for managing NFV-MANO, including the capabilities to distinguish the NFV-MANO services defined for the different NFV-MANO functional blocks and the exposed NFV-MANO service interfaces. Furthermore, ETSI GS NFV-IFA 031 [i.2] introduces the concept of NFV-MANO functional entities as the abstract modelling of the NFV-MANO functional blocks when being also considered as "managed objects".

Accordingly, the abstract NFV-MANO functional blocks correspond to NFV-MANO functional entities (NFV_MANO_Functional_Entity) shown in figure 4.2.1-1. An NFV-MANO functional entity can offer one or more capabilities as an NFV-MANO service (NFV_MANO_Service), which can be invoked using a defined interface referred to as NFV-MANO service interface (NFV_MANO_Service_Interface).

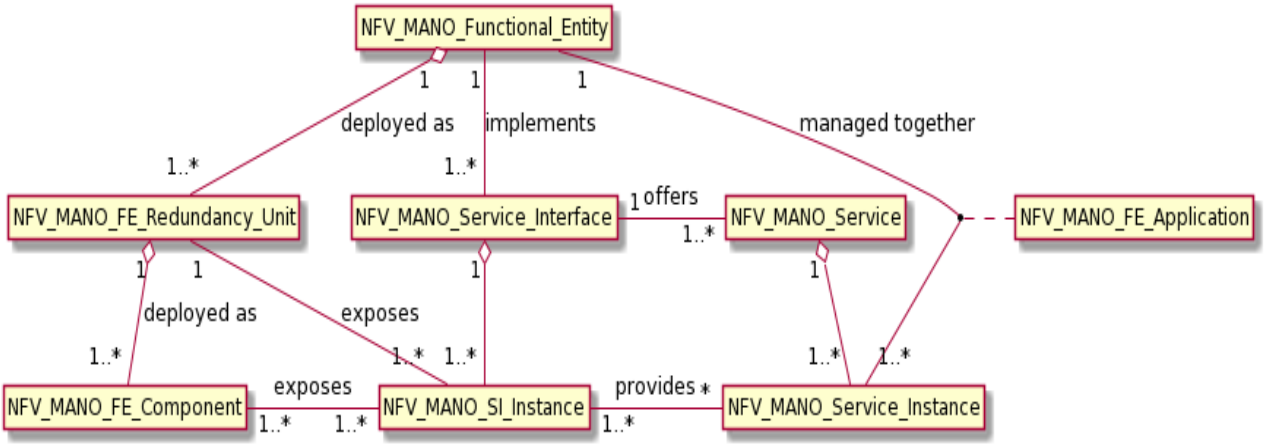

Figure 4.2.1-1: The refined concepts of the NFV-MANO functional entities

To address functional and non-functional requirements such as scalability, resiliency, versioning, an NFV-MANO functional entity can be deployed as one or more NFV-MANO functional entity redundancy units (NFV_MANO_FE_Redundancy_Unit). An NFV-MANO functional entity redundancy unit can be decomposed into NFV-MANO functional entity components (NFV_MANO_FE_Component). The redundancy units of an NFV-MANO functional entity are visible for a deployer, who can decide about the number of instances to deploy. The NFV-MANO functional entity components, on the other hand, could be hidden and known only to the NFV-MANO functional entity supplier. This is, for instance, the case when the NFV-MANO functional entity provides internal support for seamless software modification. There can be multiple instances of an NFV-MANO service or an NFV-MANO service interface (NFV_MANO_Service_Instance and NFV_MANO_SI_Instance, respectively).

At the type level, an NFV-MANO service is accessed via a single NFV-MANO interface type. However, to cater for the possibility of providing more than one API endpoints or to expose different versions of an NFV-MANO service interface, there can be multiple instances of an NFV-MANO service type and/or an NFV-MANO service interface type.

For the purpose of the management of the NFV-MANO services, the set of NFV-MANO service instances offered by an NFV-MANO functional entity can be grouped into an NFV-MANO functional entity application.

Each NFV-MANO functional entity redundancy unit and NFV-MANO functional entity component can support a set of NFV-MANO service instances. The same NFV-MANO service instance can be supported by more than one NFV-MANO functional entity redundancy units, for example, for resiliency purposes, so when one of the NFV-MANO functional entity redundancy units supporting a given NFV-MANO service instance fails or is deactivated, other redundancy units supporting the same service instance can take over the support. Thus, the lifecycle of redundancy units of an NFV-MANO functional entity can be managed independently from one another and from the NFV-MANO service instance they support.

To show these architectural concepts and their relations, the following example is based on the example in ETSI GS NFV-IFA 031 [i.2]:

EXAMPLE: Figure 4.2.1-2 is an example of services produced by a VNFM deployed as a single NFV-MANO functional entity redundancy unit of three NFV-MANO functional entity components: NFV-MANO functional entity component #1, #2 and #3. That is, no redundancy is protecting the services:

- "NFV-MANO service type A" is for VNF performance management, which is provided and accessible via interface type #1 - the "VNF performance management interface", there is one instance of such an interface;
- "NFV-MANO service type B" is for VNF fault management, which is provided and accessible via interface type #2 - the "VNF fault management interface", and there is one instance;
- "NFV-MANO service type C" is for VNF Indicator(s), which is provided and accessible via interface type #3 - the "VNF Indicator interface", and there is also one instance; and

- "NFV-MANO service type D" is for VNF lifecycle management, which is provided and accessible via interface type #4 - the "VNF Lifecycle Management interface", and there are two instances of such an interface providing different API endpoints. These API endpoints can provide different paths indicating the support of different versions of a same type of NFV-MANO service interface.

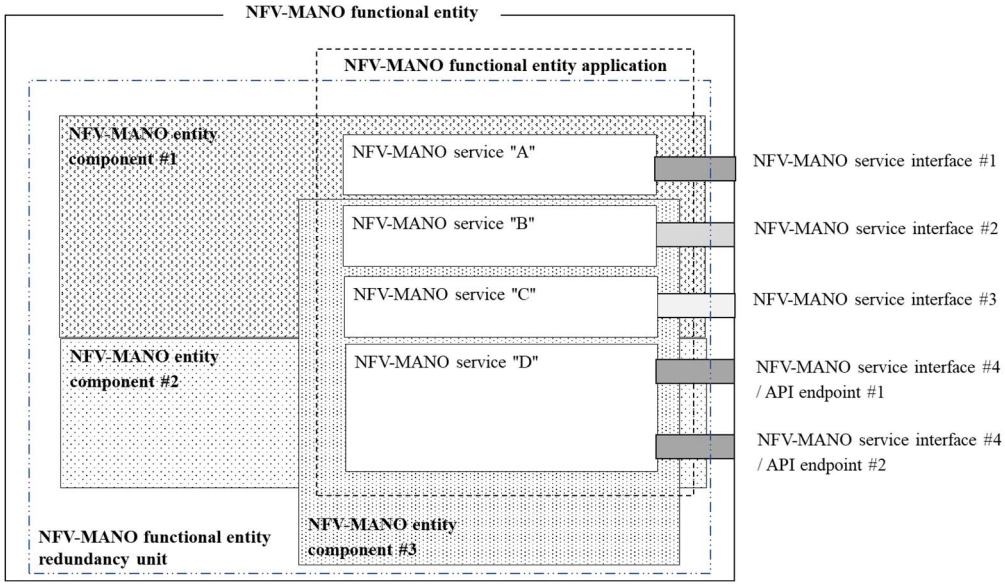

Figure 4.2.1-2: Example of relationship between NFV-MANO functional entity, NFV-MANO functional entity application, NFV-MANO service and NFV-MANO service interface [i.2]

The goal of the NFV-MANO software modification process is to modify the software deployed as redundancy units and their components of NFV-MANO functional entities, while maintaining the availability of the NFV-MANO service instances. This is only possible with the decoupling of the lifecycle of the redundancy units and their components of NFV-MANO functional entities and the NFV-MANO service instances they support.

Considering the example of figure 4.2.1-2 this means that when new software is delivered for, e.g. NFV-MANO functional entity component #2, it needs to be managed for the software modification in such a way that NFV-MANO service "D" remains available throughout the process. More specifically an instance of NFV-MANO service "D" provided through the API endpoint #1 instance of the NFV-MANO service interface type #4 continues to be provided without (or with minimal) interruption for the consumer(s) of this service instance.

In this particular case, the NFV-MANO functional entity redundancy unit encapsulates all three components, which determines the applicable software modification options. Since for the deployer the NFV-MANO functional entity redundancy unit is the smallest unit visible, it would need to manage the software modification at this level. For example, it can deploy a second NFV-MANO functional entity redundancy unit with the new version and redirect the services to it. However, a supplier being aware of the internal organization of the NFV-MANO functional entity redundancy unit can provide a software modification procedure carried out by an internal software management support, which only involves NFV-MANO functional entity component #2, e.g. by creating a redundant instance only for this component.

4.2.2 Architectural options

Considering the architectural framework of the NFV-MANO, the ETSI GS NFV-IFA 009 [i.3] report identified the following architectural options:

1) Architectural options related to VNFM, either using a generic VNFM or multiple specific VNFMs.

2) Architectural options related to VNF related resource management, depending whether VNF-related resource management is done in direct or indirect mode.

3) Architectural options related to the split of NFVO, where the NFV Orchestrator functionality could be further decomposed into several components (e.g. Resource Orchestration (RO) functions and Network Services Orchestration (NSO) functions).

4) Architectural options related to EM functions deployment in the scope of NFV.

5) Architectural options related to VNF lifecycle operation granting.

These options need to be considered with respect to the software modification of the NFV-MANO functional blocks, especially when the purpose or the consequence of the software modification is moving from one option to the other. Several of these cases imply API changes. Carrying out APIs changes without impacting the services provided by itself requires further study (see clause 5.4). Changing architectural options in addition to changing APIs adds additional complexity.

Figure 4.2.2-1 shows an example related to case (1) where multiple VNFMs dedicated to managing given VNFs are consolidated into a generic VNFM and software modification could be part of this process. In this picture, the VNFMs on the left-hand side (i.e. before the modification) are specific to manage certain VNFs.

In the case of specific VNFM, the Or-Vnfm and the Vi-Vnfm reference points become the main integration points not only for the VNFMs, but indirectly also for the VNFs and the EMs. Hence, replacing such a "specific" VNFM (e.g. VNFM managing VNF A of provider 1) with a generic VNFM of the right hand side of the figure can have impact on the EM and/or the VNF as well since the VNFM API towards them may need to be changed. The software modification process, which is involved in such changes, needs to be studied further (see clause 5.5.2).

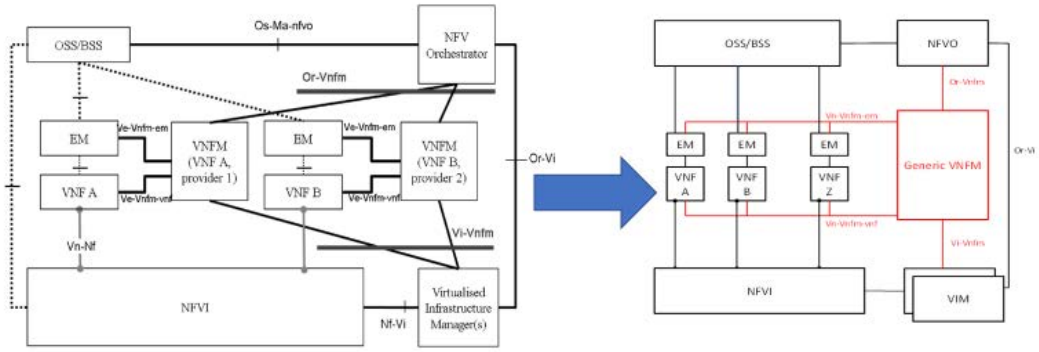

Figure 4.2.2-1: Consolidation of multiple specific VNFMs into a generic VNFM via software modification

5 Software modification use cases

5.1 Introduction

Clause 5 of the present document develops different use cases related to the software modification of NFV-MANO. A first distinction of NFV-MANO functional entities is their ability, or not, to rely on an internal support for their software modification. If this is the case, situations with a seamless support including, e.g. failures handling or compatibility issues, are treated in clause 5.2.2. Clause 5.2.3 describes the situations of a support which is not completely seamless, e.g. the software modification creates a new NFV-MANO service interface instance and the NFV-MANO service instance served before software modification is not transferred automatically over to the new one.

If there is no support for software modification embedded in the NFV-MANO functional entity, there is a need to create a redundant target NFV-MANO functional entity, in addition to the one currently running the source version of the software modification. Clause 5.3.2 analyses the situations in which NFV-MANO functional entities expose redundancy units, while clause 5.3.3 investigates software modification of NFV-MANO functional entities which do not expose their structure.

If the software of multiple NFV-MANO functional entities needs to be modified, special considerations have to be taken into account: clause 5.4 develops such situations for two NFV-MANO functional entities. A change of the NFV-MANO deployed architecture, e.g. by replacing a S-VNFM with a G-VNFM is considered in clause 5.5.

The mapping of these use cases to containerised architectures is developed in clause 5.6. Finally, use cases related to the recovery procedures (i.e. retry, rollback, fallback) which may be needed during NFV-MANO software modification are detailed in clauses 5.7, 5.8 and 5.9.

5.2 NFV-MANO functional entities with internal support for their software modification

5.2.1 Introduction

NFV-MANO functional entities may provide support for their software modification, for example, based on the internal redundancy of the components deploying an instance of the NFV-MANO functional entity. In this case, this redundancy is known and managed by the NFV-MANO functional entity itself. Therefore, for the software modification, it can be best utilized by the internal management aware of the different roles played by the different components of the NFV-MANO functional entity.

Depending on the extent to what an NFV-MANO functional entity hides its internal structure from the users of its services, there can be different cases with respect to the impact of the software modification on these users. In other words, the NFV-MANO service instances exposed through the different NFV-MANO service interface instances might or might not be impacted when, for example, the components and their redundancy unit supporting the NFV-MANO service instances fail. It can be expected that the software modification will have similar impacts. That is, if the NFV-MANO functional entity handles component/redundancy unit failures seamlessly, it is expected that it will handle the software modification in a similar manner, i.e. seamlessly.

Furthermore, since the software modification is managed by the NFV-MANO functional entity itself, it is expected that it also includes failure handling during the software modification. This means that if a failure occurs during the execution of the software modification, the NFV-MANO functional entity will try to handle it and, if the conclusion is that the software modification cannot be completed successfully, at least the changes already applied are undone and the procedure completes when the NFV-MANO functional entity returns to the configuration it was before the initiation of the software modification.

Obviously, not all failures can be handled. Therefore, it is possible that the NFV-MANO functional entity will return an error that the procedure could not be completed even by undoing the started software modification and the external manager or the administrator needs to handle the situation. This is also the case when the NFV-MANO functional entity crashes or exhibits other signs of complete failure.

In addition, if the NFV-MANO functional entity cannot complete the software modification and it assumes that this is due to an external cause, it can suspend the modification and give back the control to the SMM. Afterwards, the SMM asks the NFV-MANO functional entity to resume or to undo the already applied changes. This flow is described in clause 5.7.1.6.

5.2.2 NFV-MANO functional entities with seamless support for their software modification

5.2.2.1 Introduction

In this clause, the assumption is that the NFV-MANO functional entity is capable of providing a seamless support for software modifications which includes handling of failures and compatibility issues. Four scenarios are discussed in the following clauses for such NFV-MANO functional entities. The first two are the basic use cases, the case when the software modification is executed successfully (see flow in clause 5.2.2.5) and then, the case when a software rollback (see ETSI GR NFV 003 [i.4]) is launched either due to an error or after the successful software modification, e.g. based on the decision of an administrator (see flow in clause 5.2.2.6). These are followed by two more use cases, in which the first considers the software modification process during ongoing LCM/VRM operations, and the second considers LCM/VRM operations requested in parallel with the software modification process.

5.2.2.2 Actors and roles

Table 5.2.2.2-1 describes the actors and roles of the use case.

Table 5.2.2.2-1: Use case actors and roles

#	Role	Description
1	Software Modification Manager (SMM)	Entity deciding to launch the software modification process. Such an entity could be the OSS or an administrator.
2	Target NFV-MANO functional entity	The NFV-MANO functional entity, whose software is to be modified, may be deployed as a set of components. However, the role/redundancy of these components within this NFV-MANO functional entity is not known to other entities regardless whether they are peers or users.
3	Impacted NFV-MANO functional entity	Another NFV-MANO functional entity, which has a peering relation to the target NFV-MANO functional entity. Multiple NFV-MANO functional entities may exist with such peering relationship to the target NFV-MANO functional entity.
4	Impacted entity	Entity of a non-NFV-MANO functional block interacting with the target NFV-MANO functional entity, e.g. a VNF instance managed by a VNFM or a virtualised resource managed by a VIM. It is noteworthy that the target NFV-MANO functional entity may interact with multiple "impacted entities".
5	Tester	Entity capable of running a test, e.g. functional test, to determine that the target NFV-MANO functional entity is healthy and operating as expected. There could be multiple tests the target NFV-MANO functional entity needs to pass before reaching the final decision.

5.2.2.3 Pre-conditions

Table 5.2.2.3-1 describes the use case pre-conditions.

Table 5.2.2.3-1: Use case pre-conditions

#	Pre-condition	Additional description
1	A destination software version is available for the target NFV-MANO functional entity	The destination software version has been delivered for the target NFV-MANO functional entity together with the execution plan for the software modification process. The destination software version is backward compatible with the version currently run by the target NFV-MANO functional entity on all interfaces.
2	SMM, target and impacted NFV-MANO functional entities, as well as the impacted entities are normally operating	N/A
3	Resources are available	Any compute/storage/network resources needed for the execution of the target NFV-MANO functional entity software modification process and its testing are available. No assumption is made on the kind of resources needed and available, i.e. whether they are physical, cloud resources, or otherwise.
4	Test suites are available	Test suites for the source and the destination software versions of the target NFV-MANO functional entity are available.

5.2.2.4 Post-conditions

Table 5.2.2.4-1 describes the use case post-conditions.

Table 5.2.2.4-1: Use case post-conditions

#	Post-condition	Additional description
1	SMM, the impacted NFV-MANO functional entities, as well as the impacted entities are operating normally	N/A
2	The target NFV-MANO functional entity is operating normally	The software version of the target NFV-MANO functional entity corresponds to the destination software version if the software modification was successful; it corresponds to the source software version in the case of a successful software rollback.

5.2.2.5 Flow description for the software modification

Table 5.2.2.5-1 provides the flow description for the successful completion of an NFV-MANO software modification. Since the target NFV-MANO functional entity manages completely its software modification internally, it also handles the failure occurring during execution. Therefore, the successful completion of the software modification can also result in an undoing the started software modification and rolling back to the source software version. Thus, the successful completion does not imply that the destination software version was successfully deployed. It only means that the target NFV-MANO functional entity is in a stable operational state.

Table 5.2.2.5-1: Use case flow description for software modification

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the target NFV-MANO functional entity, including identifying a restoration point and starting special logs enabling software rollback (see notes 1 and 4).
2	SMM -> Target NFV-MANO functional entity	SMM requests the target NFV-MANO functional entity to proceed with the software modification with indication, if needed, of the appropriate resources it can use for the execution.
3	Target NFV-MANO functional entity	The target NFV-MANO functional entity executes the software modification according to its internal procedures. It also releases resources that are not needed any more.
4	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM that the software modification has completed (see notes 2 and 3).
5	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution. The tester runs all the test suites applicable for the verification of the health and the performance of the target NFV-MANO functional entity and the system as a whole.
6	Tester -> SMM	The tester determines that the software modification of the target NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 3).
Ends when	SMM	SMM commits the software modification process and releases resources that were used to protect the software modification process and enable software rollback (see note 4).

NOTE 1: The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the target NFV-MANO functional entity and its context in such a way that this target NFV-MANO functional entity can be restored from this restoration point with enough context to resume operation even after a crash.

NOTE 2: If the target NFV-MANO functional entity has deployed the destination software version successfully, it reports that the software modification was completed successfully. In case the target NFV-MANO functional entity encountered a failure, but was able to roll back to the source version successfully, the target NFV-MANO functional entity reports that it has completed the software modification with automatic rollback to the source software version. In this case, SMM can decide to retry the operation.

NOTE 3: If SMM detects a failure of the target NFV-MANO functional entity, e.g. it crashed, it returned with an error other than in note 2, or the testing in step 6 failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it executes a software rollback described in clause 5.2.2.6, or the fallback procedure described in clause 5.9.

NOTE 4: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, the software rollback becomes not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.

5.2.2.6 Flow description for the software rollback

Table 5.2.2.6-1 provides the flow description for a software rollback executed by the SMM. As mentioned in clause 5.2.2.5, the target NFV-MANO functional entity may automatically roll back to the source version of the software as part of the software modification process. This case is covered in clause 5.2.2.5, while the case covered in this clause is typically applicable, e.g. when the target NFV-MANO functional entity does not pass the tests performed after the completion of a successful software modification (i.e. the target NFV-MANO functional entity has deployed the destination software).

Table 5.2.2.6-1: Use case flow description for software rollback

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to start a software rollback (e.g. because the functional testing in step 6 of clause 5.2.2.5 ended unsuccessfully).
1	SMM	SMM checks if adequate resources are available for rolling back the software modification as indicated in the execution plan of the software modification process.
2	SMM -> Target NFV-MANO functional entity	SMM requests the target NFV-MANO functional entity to do a software modification to the source software version and, therefore, roll back to the software modification. SMM also indicates, if needed, the resources that can be used for the execution of the software rollback(i.e. software modification to the source software version).
3	Target NFV-MANO functional entity	The target NFV-MANO functional entity executes the modifications and rolls back to the source software version according to its internal procedures. It also releases resources that are not needed any more.
4	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM that the software modification to the source software version has completed.
5	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution. The tester runs all the test suites applicable for the verification of the health and the performance of the target NFV-MANO functional entity and the system as a whole.
6	Tester -> SMM	The tester determines that the software rollback of the target NFV-MANO functional entity has been completed successfully. Not needed resources are released.
Ends when	SMM	SMM commits the software modification process and releases resources that were used to protect the software modification process and enable software rollback.
NOTE: If, during execution, SMM detects a failure of the target NFV-MANO functional entity, e.g. it crashed, it returned with an error, or the testing of step 6 failed, SMM determines if a retry as described in clause 5.8 is appropriate, otherwise it can initiate the fallback procedure described in clause 5.9.		

5.2.2.7 Flow description for the software modification with ongoing LCM/VRM operation

Table 5.2.2.7-1 provides the flow description for the successful completion of an NFV-MANO software modification while an VNF LCM operation, NS LCM operation, and/or VRM operation is ongoing when the software modification process is triggered. This use case is supported by the requirements in clause 5.6.2 of ETSI GS NFV-IFA 010 [i.7].

Table 5.2.2.7-1: Use case flow description for software modification with ongoing LCM operation

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if a VNF LCM operation, NS LCM operation, and/or VRM operation is currently ongoing. If yes, depending on the type of the NS LCM operation, VNF LCM operation and/or VRM operation, the operator's configuration, and the capabilities of the software modification for the NFV-MANO functional entity to be modified, an appropriate action is taken (see note 5). Appropriate actions include: • the software modification is immediately continued with step 2 and is executed in parallel to the ongoing operation; or • the software modification is suspended until the NS LCM operation, VNF LCM operation and/or VRM operation are/is completed.
2	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the target NFV-MANO functional entity, including identifying a restoration point and starting special logs enabling software rollback (see notes 1 and 4).
3	SMM -> Target NFV-MANO functional entity	SMM requests the target NFV-MANO functional entity to proceed with the software modification with indication, if needed, of the appropriate resources it can use for the execution.
4	Target NFV-MANO functional entity	The target NFV-MANO functional entity executes the software modification according to its internal procedures. It also releases resources that are not needed any more.
5	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM that the software modification has completed (see notes 2 and 3).
6	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution. The tester runs all the test suites applicable for the verification of the health and the performance of the target NFV-MANO functional entity and the system as a whole.
7	Tester -> SMM	The tester determines that the software modification of the target NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 3).
Ends when	SMM	SMM commits the software modification process and releases resources that were used to protect the software modification process and enable software rollback (see note 4).

NOTE 1: The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the target NFV-MANO functional entity and its context in such a way that this target NFV-MANO functional entity can be restored from this restoration point with enough context to resume operation even after a crash.

NOTE 2: If the target NFV-MANO functional entity has deployed the destination software version successfully, it reports that the software modification was completed successfully. In case the target NFV-MANO functional entity encountered a failure, but was able to roll back to the source version successfully, the target NFV-MANO functional entity reports that it has completed the software modification with automatic rollback to the source software version. In this case, SMM can decide to retry the operation.

NOTE 3: If SMM detects a failure of the target NFV-MANO functional entity, e.g. it crashed, it returned with an error other than in note 2, or the testing in step 7 failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it executes a software rollback described in clause 5.2.2.6, or the fallback procedure described in clause 5.9.

NOTE 4: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, the software rollback becomes not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.

NOTE 5: SMM is aware of the related capabilities of the NFV-MANO functional entity as well as the operator's configuration via some sort of descriptors or policies provided by the functional entity and the operator, respectively.

5.2.2.8 Flow description for the software modification with parallel LCM/VRM operation

Table 5.2.2.8-1 provides the flow description for the successful completion of an NFV-MANO software modification when a VNF LCM operation, NS LCM operation and/or VRM operation is triggered during the software modification process. This use case is supported by the requirements in clause 5.6.2 of ETSI GS NFV-IFA 010 [i.7].

Table 5.2.2.8-1: Use case flow description for software modification with parallel LCM/VRM operation

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the target NFV-MANO functional entity, including identifying a restoration point and starting special logs enabling software rollback (see notes 1 and 4).
2	SMM	At any time during the software modification (step 3 to step 8), SMM learns about a new NS LCM operation, VNF LCM operation and/or VRM operation that was triggered. Depending on the type of the NS LCM operation, VNF LCM operation and/or VRM operation, the operator's configuration, and the capabilities of the software modification for the NFV-MANO functional entity to be modified, an appropriate action is taken (see note 5). Appropriate actions include: • execute the operation in parallel with the software modification of the NFV-MANO functional entity, • delay the execution of the operation until the NFV-MANO software modification has completed, • notify the pending VNF LCM operation to consumer such that the consumer can trigger the VNF LCM operation again after the completion of the software modification, or • suspend the ongoing software modification at the next possibility (see use case in clause 5.7) until the triggered operation has completed.
3	SMM -> Target NFV-MANO functional entity	SMM requests the target NFV-MANO functional entity to proceed with the software modification with indication, if needed, of the appropriate resources it can use for the execution.
4	Target NFV-MANO functional entity	The target NFV-MANO functional entity executes the software modification according to its internal procedures. It also releases resources that are not needed any more.
5	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM that the software modification has completed (see notes 2 and 3).
6	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution. The tester runs all the test suites applicable for the verification of the health and the performance of the target NFV-MANO functional entity and the system as a whole.
7	Tester -> SMM	The tester determines that the software modification of the target NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 3).
8	SMM	SMM commits the software modification process and releases resources that were used to protect the software modification process and enable software rollback (see note 4).
Ends when	SMM	Depending on the action taken in step 2, SMM will trigger the execution of the pending NS LCM, VNF LCM and/or VRM operation, or SMM will notify the consumer such that the consumer can again trigger the operation that had not yet been executed.

#	Actor/Role-condition	Action/Description
NOTE 1:	The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the target NFV-MANO functional entity and its context in such a way that this target NFV-MANO functional entity can be restored from this restoration point with enough context to resume operation even after a crash.	
NOTE 2:	If the target NFV-MANO functional entity has deployed the destination software version successfully, it reports that the software modification was completed successfully. In case the target NFV-MANO functional entity encountered a failure, but was able to roll back to the source version successfully, the target NFV-MANO functional entity reports that it has completed the software modification with automatic rollback to the source software version. In this case, SMM can decide to retry the operation.	
NOTE 3:	If SMM detects a failure of the target NFV-MANO functional entity, e.g. it crashed, it returned with an error other than in note 2, or the testing in step 7 failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it executes a software rollback described in clause 5.2.2.6, or the fallback procedure described in clause 5.9.	
NOTE 4:	Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, the software rollback becomes not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.	
NOTE 5:	SMM is aware of the related capabilities of the NFV-MANO functional entity as well as the operator's configuration via some sort of descriptors or policies provided by the functional entity and the operator, respectively.	

5.2.3 NFV-MANO functional entities with non-seamless support for their software modification

5.2.3.1 Introduction

As opposed to clause 5.2.2, clause 5.2.3 presents the use case for an NFV-MANO functional entity which provides internal support for software modifications; however, this support is not completely seamless. Namely, the software modification results in the creation of one or more new NFV-MANO service interface instances and the NFV-MANO service instance(s) served before the software modification are not transferred automatically over to these new NFV-MANO service interface instance(s). This could be, for example, because the new software version supports new NFV-MANO service interface versions. Accordingly, the use case distinguishes old and new NFV-MANO service interfaces and their instances, where old means NFV-MANO service interface instances open before the initiation of the software modification process; new means NFV-MANO service interface instances created as a result of the software modification process.

After the software modification, the NFV-MANO functional entity may expose the old and new NFV-MANO service interfaces simultaneously throughout its lifecycle, or the old NFV-MANO service interfaces may be shut down after some transitional period. The peculiarity of the use case is that rolling back removes the new NFV-MANO service interface instances. This means that if they were already started, they require graceful stopping to minimize service impact.

The use case assumes that in all other aspects of the software modification - such as ongoing/parallel LCM and failure handling - the NFV-MANO functional entity provides an internal support similar to the seamless case of clause 5.2.2. The following clauses present the flows for the successful software modification and for the software rollback scenarios.

5.2.3.2 Actors and roles

Table 5.2.3.2-1 describes the actors and roles of the use case.

Table 5.2.3.2-1: Use case actors and roles

#	Role	Description
1	Software Modification Manager (SMM)	Entity deciding to launch the software modification process. Such an entity could be the OSS or an administrator.
2	Target NFV-MANO functional entity	The NFV-MANO functional entity, whose software is to be modified, may be deployed as a set of components. However, the role/redundancy of these components within this NFV-MANO functional entity is not known to other entities regardless whether they are peers or users.
3	Impacted NFV-MANO functional entity	Another NFV-MANO functional entity, which has a peering relation to the target NFV-MANO functional entity. Multiple NFV-MANO functional entities may exist with such peering relationship to the target NFV-MANO functional entity.
4	Impacted entity	Entity of a non-NFV-MANO functional block interacting with the target NFV-MANO functional entity, e.g. a VNF instance managed by a VNFM or a virtualised resource managed by a VIM. It is noteworthy that the target NFV-MANO functional entity may interact with multiple "impacted entities".
5	Tester	Entity capable of running a test, e.g. functional test, to determine that the target NFV-MANO functional entity is healthy and operating as expected. There could be multiple tests the target NFV-MANO functional entity needs to pass before reaching the final decision.

5.2.3.3 Pre-conditions

Table 5.2.3.3-1 describes the use case pre-conditions.

Table 5.2.3.3-1: Use case pre-conditions

#	Pre-condition	Additional description
1	A destination software version is available for the target NFV-MANO functional entity	The destination software version has been delivered for the target NFV-MANO functional entity together with the execution plan for the software modification process. The new software version adds support of new NFV-MANO service interface versions, or it adds new NFV-MANO service(s) and corresponding NFV-MANO service interface(s) support.
2	SMM, target and impacted NFV-MANO functional entities, as well as the impacted entities are operating normally	N/A
3	Resources are available	Any compute/storage/network resources needed for the execution of the target NFV-MANO functional entity software modification process and its testing are available. No assumption is made on the kind of resources needed and available, i.e. whether they are physical, cloud resources, or otherwise.
4	Test suites are available	Test suites for the source and the destination software versions of the target NFV-MANO functional entity are available.

5.2.3.4 Post-conditions

Table 5.2.3.4-1 describes the use case post-conditions.

Table 5.2.3.4-1: Use case post-conditions

#	Post-condition	Additional description
1	SMM, the impacted NFV-MANO functional entities, as well as the impacted entities are operating normally	N/A
2	The target NFV-MANO functional entity is operating normally	The software version of the target NFV-MANO functional entity corresponds to the destination software version if the software modification was successful; it corresponds to the source software version in the case of a successful software rollback.

5.2.3.5 Flow description for the software modification

Table 5.2.3.5-1 provides the flow description for the successful completion of an NFV-MANO software modification. Since the target NFV-MANO functional entity manages its software modification internally, it also handles any failure occurring during execution. However, in contrast to clause 5.2.2 - the seamless case -, if this internal software modification results in an undoing of the started software modification and rolling back to the source software version, any new NFV-MANO service interface is not exposed and therefore the overall process fails and the SMM needs to handle this failure either by a retry, a rollback or a fallback.

Table 5.2.3.5-1: Use case flow description for software modification

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the target NFV-MANO functional entity, including identifying a restoration point and starting special logs enabling software rollback (see notes 1 and 4).
2	SMM	SMM collects the information needed to perform the software modification, especially the information about the old and new NFV-MANO service interfaces and their relevance to impacted entities and impacted NFV-MANO functional entities. This can be part of the execution plan, requested from an administrator or from the target NFV-MANO functional entity.
3	SMM -> Target NFV-MANO functional entity	SMM requests the target NFV-MANO functional entity to proceed with the software modification with indication, if needed, of the appropriate resources it can use for the execution.
4	Target NFV-MANO functional entity	The target NFV-MANO functional entity executes the software modification according to its internal procedures. It also releases resources that are not needed any more.
5	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM that the software modification has completed (see notes 2 and 3).
6	SMM -> Target NFV-MANO functional entity	According to the collected information, SMM verifies the LOCKED state of the new NFV-MANO service interfaces (see note 3).
7	SMM, target NFV-MANO functional entity, impacted NFV-MANO functional entity	SMM starts the new NFV-MANO service interfaces towards impacted NFV-MANO functional entities. Authentication of the target NFV-MANO functional entity to the impacted NFV-MANO functional entities (see note 3).
8	SMM, target NFV-MANO functional entity, impacted entity	SMM starts the NFV-MANO service interfaces towards impacted entities. Authentication of the target NFV-MANO functional entity to the impacted entity (see note 3).
9	SMM, target NFV-MANO functional entity	If applicable according to the execution plan, SMM gracefully stops the old NFV-MANO service interfaces towards impacted entities (see note 3).
10	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution. The tester runs all the test suites applicable for the verification of the health and the performance of the target NFV-MANO functional entity and the system as a whole.
11	Tester -> SMM	The tester determines that the software modification of the target NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 3).
12	SMM, target NFV-MANO functional entity	If any old NFV-MANO service interface was stopped in step 9, SMM waits until they reach the SHUTDOWN_UNLOCKED state. If applicable, SMM stops corresponding old NFV-MANO service interfaces towards impacted NFV-MANO functional entities. The target NFV-MANO functional entity may release resources that are not needed any more (see note 3).
Ends when	SMM	SMM commits the software modification process and releases resources that were used to protect the software modification process and enable software rollback (see note 4).

#	Actor/Role-condition	Action/Description
NOTE 1:	The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the target NFV-MANO functional entity and its context in such a way that this target NFV-MANO functional entity can be restored from this restoration point with enough context to resume operation even after a crash.	
NOTE 2:	If the target NFV-MANO functional entity has deployed the destination software version successfully, it reports that the software modification was completed successfully. In case the target NFV-MANO functional entity encountered a failure, but was able to roll back to the source version successfully, the target NFV-MANO functional entity reports that it has completed the software modification with automatic rollback to the source software version. In this case, SMM can decide to retry the operation.	
NOTE 3:	If SMM detects a failure of the target NFV-MANO functional entity, e.g. it crashed, it returned with an error other than in note 2, or the testing in step 11 failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it executes a software rollback described in clause 5.2.3.6 or the fallback procedure described in clause 5.9.	
NOTE 4:	Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, these resources with this information are released making software rollback not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.	

5.2.3.6 Flow description for the software rollback

Table 5.2.3.6-1 provides the flow description for a software rollback executed by the SMM on the NFV-MANO functional entity with software modification support. The first step executed in this flow depends on the step of the software modification flow at which the SMM decides to roll back the software modification, for example, because of a failure. The "At #" column indicates this step of the software modification flow.

Table 5.2.3.6-1: Use case flow description for software rollback

#	Actor/Role-condition	Action/Description	At #
Begins when	SMM	SMM decides to start a software rollback.	
1	SMM	SMM checks if adequate resources are available for rolling back the software modification as indicated in the execution plan of the software modification process (see note 1).	
2	SMM	SMM collects the information needed to perform the software rollback. This can be part of the execution plan, requested from an administrator or from the target NFV-MANO functional entity. Alternatively, the information collected during the software modification flow can be stored for the case the software modification needs to be rolled back (see note 2).	
3	SMM, target NFV-MANO functional entity, impacted NFV-MANO functional entity	If applicable according to the execution plan, SMM starts the old NFV-MANO service interfaces towards impacted NFV-MANO functional entities. Authentication of the target NFV-MANO functional entity to the impacted NFV-MANO functional (see note 3).	9 to 12
4	SMM, target NFV-MANO functional entity, impacted entity	If applicable according to the execution plan, SMM starts the old NFV-MANO service interfaces towards impacted entities. Authentication of the target NFV-MANO functional entity to the impacted entity (see note 3).	
5	SMM, target NFV-MANO functional entity,	SMM gracefully stops the new NFV-MANO service interfaces towards impacted entities and waits until the SHUTDOWN_UNLOCKED state is reached (see note 3).	8
6	SMM, target NFV-MANO functional entity	SMM gracefully stops the NFV-MANO service interfaces towards impacted NFV-MANO functional entities (see note 3).	6 to 7
7	SMM -> Target NFV-MANO functional entity	SMM requests the target NFV-MANO functional entity to do a software modification to the source software version and, therefore, roll back to the software modification. SMM also indicates, if needed, the resources that can be used for the execution of the software rollback (i.e. software modification to the source software version (see note 3).	3 to 5
8	Target NFV-MANO functional entity	The target NFV-MANO functional entity executes the modifications and rolls back to the source software version according to its internal procedures. It also releases resources that are not needed any more.	

#	Actor/Role-condition	Action/Description	At #
9	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM that the software modification to the source software version has completed (see note 3).	
10	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution. The tester runs all the test suites applicable for the verification of the health and the performance of the target NFV-MANO functional entity and the system as a whole.	
11	Tester -> SMM	The tester determines that the software modification of the target NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 3).	
Ends when	SMM	SMM commits the software modification process and releases resources that were used to protect the software modification process and enable software rollback (see note 4).	

NOTE 1: The software rollback can only proceed successfully if adequate resources are available. Therefore, this check can always be part of the decision of initiating the software rollback regardless at which step the software rollback flow is entered.
NOTE 2: None, some or the entire step of information collection can be performed before entering the software rollback flow depending on whether SMM maintained the information collected during the software modification to be used at the software rollback.
NOTE 3: If SMM detects a failure of the target NFV-MANO functional entity, e.g. it crashed, returned with an error, or the testing in step 11 has failed, SMM decides whether a retry is appropriate as described in clause 5.8, or initiates the fallback procedure described in clause 5.9.
NOTE 4: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, these resources with this information are released making software rollback not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.

5.2.4 Special considerations for NFVO

The NFVO software modification use case is obtained by using the generic use case defined in clause 5.2.2 after replacing the generic roles described in clause 5.2.2.2 by the NFVO-related roles indicated in table 5.2.4-1 below.

Table 5.2.4-1: NFVO use case actors and roles

#	Role described in clause 5.2.2.2	Role related to the NFVO use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Target NFV-MANO functional entity	NFVO instance
3	Impacted NFV-MANO functional entity	Other NFVO instances (if applicable), VNFM instance(s), VIM instance(s)
4	Impacted entity	OSS
5	Tester	Tester

5.2.5 Special considerations for VNFM

The VNFM software modification use case is obtained by using the generic use case defined in clause 5.2.2 after replacing the generic roles described in clause 5.2.2.2 by the VNFM-related roles indicated in table 5.2.5-1 below.

Table 5.2.5-1: VNFM use case actors and roles

#	Role described in clause 5.2.2.2	Role related to the VNFM use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Target NFV-MANO functional entity	VNFM instance
3	Impacted NFV-MANO functional entity	NFVO instance, VIM instance(s)
4	Impacted entity	VNF instance(s), EM(s)
5	Tester	Tester

5.2.6 Special considerations for VIM

The VIM software modification use case is obtained by using the generic use case defined in clause 5.2.2 after replacing the generic roles described in clause 5.2.2.2 by the VIM-related roles indicated in table 5.2.6-1 below.

Table 5.2.6-1: VIM use case actors and roles

#	Role described in clause 5.2.2.2	Role related to the VIM use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Target NFV-MANO functional entity	VIM instance
3	Impacted NFV-MANO functional entity	NFVO instance, VNFM instance(s)
4	Impacted entity	NFVI resources
5	Tester	Tester

5.3 NFV-MANO functional entities without internal support for their software modification

5.3.1 Introduction

An NFV-MANO functional entity may provide no support for a planned software modification. This is the case, for example, when the source version and the destination version of the planned software modification are not compatible and cannot interact to carry out the software modification process. Thus, in this case, there is no support embedded in the NFV-MANO functional entity or if there is one, it is not sufficient to rely on for the purpose of the software modification. Therefore, an external manager needs to manage the software modification process, including the software modification of the NFV-MANO functional entity. The external manager is referred to as the Software Modification Manager or SMM. Essentially the SMM needs to "forklift" the NFV-MANO functional entity from the source to the destination version.

For this purpose, the SMM creates a redundant target NFV-MANO functional entity, in addition to the one currently running the source version of the software modification. This new redundant target NFV-MANO functional entity will be running the destination version of the software modification. The SMM then manages the handover of service instances between the redundant target NFV-MANO functional entities. The SMM also takes care of any other aspects of the software modification process such as the steps necessary for failure handling should the endeavour fail, that is, enabling the software retry, rollback and fallback options.

5.3.2 NFV-MANO functional entities with redundancy units

5.3.2.1 Introduction

According to the detailed information modelling of NFV-MANO functional entities presented in clause 4.2.1, an NFV-MANO functional entity can be composed of redundant components deployed as redundancy units to ensure the availability and continuity of the NFV-MANO services of the NFV-MANO functional entity. In fact, an NFV-MANO functional entity providing internal support for the software modification process can be relying on such a redundant internal composition. However, since the redundancy of the NFV-MANO functional entity and the internal support for software modification are independent properties of the NFV-MANO functional entity, one needs to consider them separately.

Clause 5.2 considers the use cases for NFV-MANO functional entities with internal support for the software modification. In clause 5.3.2, the considered software modification use case is when no such internal support is provided by the NFV-MANO functional entities themselves which are composed of redundant components exposed as redundancy units that can be managed by an external entity such as the SMM. That is, for the SMM to be able to take advantage of the redundancy of NFV-MANO functional entity components, these components need to be visible to, and manageable by, the SMM as redundancy units. If so, the SMM can manage their software modification in a rolling manner.

There is no assumption made whether the SMM is aware of the role the different redundancy units of the NFV-MANO functional entity play in their redundancy model. It simply instantiates a new NFV-MANO functional entity redundancy unit with the destination software version, and once its NFV-MANO service interfaces are open, the SMM gracefully stops a selected NFV-MANO functional entity redundancy unit running the source software version. Information about this selected source NFV-MANO functional entity redundancy unit is provided to the destination NFV-MANO functional entity redundancy unit at its instantiation.

The destination NFV-MANO functional entity redundancy unit may use this information as necessary for state synchronization with the selected source NFV-MANO functional entity redundancy unit without the SMM being aware of it. The SMM is only aware of the selected source NFV-MANO functional entity redundancy unit entering the UNLOCKED_SHUTDOWN state (see ETSI GS NFV-IFA 031 [i.2]), when the SMM can terminate it without impacting the NFV-MANO services the NFV-MANO functional entity provides. If there are other source NFV-MANO functional entity redundancy units, the SMM repeats the process for each of them.

5.3.2.2 Actors and roles

Table 5.3.2.2-1 describes the actors and roles of the use case.

Table 5.3.2.2-1: Use case actors and roles

#	Role	Description
1	Software Modification Manager (SMM)	Entity deciding to launch and managing the software modification process. Such an entity could be the OSS or an administrator.
2	Source NFV-MANO functional entity redundancy unit	The NFV-MANO functional entity redundancy unit running the source version of the planned software modification. There can be many source NFV-MANO functional entity redundancy units (see note).
3	Destination NFV-MANO functional entity redundancy unit	The NFV-MANO functional entity redundancy unit running the destination version of the planned software modification. There can be many destination NFV-MANO functional entity redundancy units (see note).
4	Impacted NFV-MANO functional entity	Another NFV-MANO functional entity, which has a peering relation to the target NFV-MANO functional entity. Multiple NFV-MANO functional entities may exist with such peering relationship to the target NFV-MANO functional entities (see note).
5	Impacted entity	Entity of a non-NFV-MANO functional block interacting with the target NFV-MANO functional entity, e.g. a VNF instance managed by a VNFM or a virtualised resource managed by a VIM. It is noteworthy that the target NFV-MANO functional entity may interact with multiple "impacted entities".
6	Tester	Entity capable of running a test, e.g. functional test, to determine that the target NFV-MANO functional entity is healthy and operating as expected. There could be multiple tests the target NFV-MANO functional entity needs to pass.
NOTE: The source and destination NFV-MANO functional entity redundancy units are commonly referred to as target NFV-MANO functional entity redundancy unit (units) when the emphasis is on the fact that they are targeted by the software modification process. The number of the different target NFV-MANO functional entity redundancy units and the software version they are running depend on the stage of the execution. The target NFV-MANO functional entity redundancy units form redundancy for the purpose of protecting the NFV-MANO service instance(s) they provide.		

5.3.2.3 Pre-conditions

Table 5.3.2.3-1 describes the use case pre-conditions.

Table 5.3.2.3-1: Use case pre-conditions

#	Pre-condition	Additional description
1	A destination software version is available for the target NFV-MANO functional entity redundancy units	The destination software version has been delivered for the target NFV-MANO functional entity redundancy unit.
2	SMM is operating normally, it has access to the target NFV-MANO functional entity redundancy units, and has the software modification execution plan	SMM is aware of the target NFV-MANO functional entity redundancy units. The execution plan for the software modification process is available for SMM. SMM knows the steps and how to interact with the source and the destination versions of the target NFV-MANO functional entity redundancy units for the purpose of the software modification process.
3	Target and impacted NFV-MANO functional entities as well as the impacted entities are operating normally	The target NFV-MANO functional entity redundancy units operate in a redundant manner to protect the NFV-MANO service instances they support.
4	Resources are available	Any compute/storage/network resources needed for the execution of the software modification of the target NFV-MANO functional entity redundancy units and its testing are available. No assumption is made on the kind of resources needed and available, i.e. whether they are physical, cloud resources, or otherwise.
5	Test suites are available	Test suites for the source and the destination software versions of the target NFV-MANO functional entity are available.

5.3.2.4 Post-conditions

Table 5.3.2.4-1 describes the use case post-conditions.

Table 5.3.2.4-1: Use case post-conditions

#	Post-condition	Additional description
1	SMM, the impacted NFV-MANO functional entities, as well as the impacted entities are operating normally	SMM, the impacted entities as well as the impacted NFV-MANO functional entities can interact with the target NFV-MANO functional entity, i.e. the target NFV-MANO functional entity fulfils its duties towards them.
2	The target NFV-MANO functional entity redundancy units are operating normally.	The software version of the target NFV-MANO functional entity redundancy unit corresponds to the destination software version if the software modification was successful; it corresponds to the source software version in the case of a successful software rollback.

5.3.2.5 Flow description for the software modification

Table 5.3.2.5-1 provides the flow description for a successful software modification of the target NFV-MANO functional entity components.

Table 5.3.2.5-1: Use case flow description for software modification

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if adequate resources are available for the software modification process as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the target NFV-MANO functional entity redundancy units, including identifying a restoration point and starting special logs enabling rollback (see notes 1 and 3).
2	SMM	SMM collects the information needed to perform the software modification (e.g. network configuration parameters). This can be part of the execution plan, requested from an administrator or from the source NFV-MANO functional entity redundancy units.

#	Actor/Role-condition	Action/Description
3	SMM	According to the execution plan and using some of the collected information, SMM instantiates a destination NFV-MANO functional entity redundancy unit with its NFV-MANO service interfaces in the LOCKED state. The capacity, NFV-MANO service interfaces, peering information are set so that when the NFV-MANO service interfaces are unlocked, it will be able to replace a selected source NFV-MANO functional entity redundancy unit (see note 2).
4	SMM -> Tester	SMM invokes the tester to verify the health of the new destination NFV-MANO functional entity redundancy unit. SMM also indicates the appropriate resources the tester can use for execution (see note 2).
5	Tester	The tester executes the test suites appropriate to determine the health of the new destination NFV-MANO functional entity redundancy unit.
6	Tester -> SMM	The tester reports back that the destination NFV-MANO functional entity redundancy unit is healthy. If appropriate, not needed resources are released (see note 2).
7	SMM -> Destination NFV-MANO functional entity redundancy unit	If necessary, SMM transfers the operational state data (e.g. authentication key) from the selected source NFV-MANO functional entity redundancy unit to the destination NFV-MANO functional entity redundancy unit (see note 2).
8	SMM, destination NFV-MANO functional entity redundancy unit, impacted NFV-MANO functional entity	SMM starts the NFV-MANO service interfaces of the destination NFV-MANO functional entity redundancy unit towards impacted NFV-MANO functional entities. Authentication of the destination NFV-MANO functional entity redundancy unit to the impacted NFV-MANO functional entity if necessary (see note 2).
9	SMM, destination NFV-MANO functional entity redundancy unit, impacted entity	SMM starts the NFV-MANO service interfaces towards impacted entities. Authentication of the destination NFV-MANO functional entity redundancy unit to the impacted entity if necessary (see note 2).
10	SMM, source NFV-MANO functional entity redundancy unit	SMM gracefully stops the NFV-MANO service interfaces of the selected NFV-MANO functional entity redundancy unit towards impacted entities (see note 2).
11	SMM -> source NFV-MANO functional entity redundancy unit	When the selected source NFV-MANO functional entity redundancy unit reaches the SHUTDOWN_UNLOCKED state, SMM terminates the source NFV-MANO functional entity redundancy unit (see note 2).
12	SMM	If there is any source NFV-MANO functional entity redundancy unit remaining, SMM selects one and continues with step 3 to apply the software modification to it. If no source NFV-MANO functional entity redundancy unit is remaining, SMM proceeds to step 13.
13	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution (see note 2).
14	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the destination NFV-MANO functional entity redundancy units and the system as a whole (see note 2).
15	Tester -> SMM	The tester reports that the software modification of the destination NFV-MANO functional entity redundancy units have been completed successfully. Not needed resources are released (see note 2).
Ends when	SMM	SMM commits the software modification process and releases the resources that were used to protect the software modification process and that enabled the software rollback (see note 3).

NOTE 1: The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the target NFV-MANO functional entity or its redundancy units and its context in such a way that this target NFV-MANO functional entity can be restored from this restoration point with enough context to resume operation even after a crash.

NOTE 2: If SMM detects a failure of the destination NFV-MANO functional entity redundancy unit, e.g. it crashed, it returned with an error, or the testing in step 6 or 15 failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it executes a software rollback described in clause 5.2.2.6 or the fallback procedure described in clause 5.9.

#	Actor/Role-condition	Action/Description
NOTE 3: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification process without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, software rollback becomes not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.		

5.3.3 NFV-MANO functional entities not exposing their structure

5.3.3.1 Introduction

Two generic scenarios are discussed in the following clauses for the software modification of such NFV-MANO functional entities: first the case for the software modification (see flow in clause 5.3.3.5), then the case when a software rollback is launched, for example, due to some failure during the software modification. This software rollback can also be launched after the successful execution of the software modification, e.g. based on the decision of an administrator before committing the executed software modification process (see flow in clause 5.3.3.6).

5.3.3.2 Actors and roles

Table 5.3.3.2-1 describes the actors and roles of the use case.

Table 5.3.3.2-1: Use case actors and roles

#	Role	Description
1	Software Modification Manager (SMM)	Entity deciding to launch and managing the software modification process. Such an entity could be the OSS or an administrator.
2	Source NFV-MANO functional entity	The NFV-MANO functional entity running the source version of the planned software modification (see note).
3	Destination NFV-MANO functional entity	The NFV-MANO functional entity running the destination version of the planned software modification (see note).
4	Impacted NFV-MANO functional entity	Another NFV-MANO functional entity, which has a peering relation to the target NFV-MANO functional entity (see note). Multiple NFV-MANO functional entities may exist with such peering relationship to the target NFV-MANO functional entities.
5	Impacted entity	Entity of a non-NFV-MANO functional block interacting with the target NFV-MANO functional entity (see note), e.g. a VNF instance managed by a VNFM or a virtualised resource managed by a VIM. It is noteworthy that the target NFV-MANO functional entity may interact with multiple "impacted entities".
6	Tester	Entity capable of running a test, e.g. functional test, to determine that the target NFV-MANO functional entity (see note) is healthy and operating as expected. There could be multiple tests the target NFV-MANO functional entity needs to pass.
NOTE: The source and destination NFV-MANO functional entities are commonly referred to as target NFV-MANO functional entity (entities) when the emphasis is on the fact that they are targeted by the software modification process. The number of target NFV-MANO functional entities and the software version they are running depend on the stage of the execution.		

5.3.3.3 Pre-conditions

Table 5.3.3.3-1 describes the use case pre-conditions.

Table 5.3.3.3-1: Use case pre-conditions

#	Pre-condition	Additional description
1	A destination software version is available for the target NFV-MANO functional entity	The destination software version has been delivered for the target NFV-MANO functional entity.
2	SMM is operating normally and has the software modification execution plan	The execution plan for the software modification process is available for SMM. SMM knows the steps and how to interact with the source and the destination versions of the target NFV-MANO functional entity for the purpose of the software modification process.
3	Source and impacted NFV-MANO functional entities as well as the impacted entities are normally operating	N/A
4	Resources are available	Any compute/storage/network resources needed for the execution of the software modification of the target NFV-MANO functional entity and its testing are available. No assumption is made on the kind of resources needed and available, i.e. whether they are physical, cloud resources, or otherwise.
5	Test suites are available	Test suites for the source and the destination software versions of the target NFV-MANO functional entity are available.

5.3.3.4 Post-conditions

Table 5.3.3.4-1 describes the use case post-conditions.

Table 5.3.3.4-1: Use case post-conditions

#	Post-condition	Additional description
1	SMM, the impacted NFV-MANO functional entities, as well as the impacted entities are operating normally	SMM, the impacted entities as well as the impacted NFV-MANO functional entities can interact with the target NFV-MANO functional entity, i.e. the target NFV-MANO functional entity fulfils its duties towards them.
2	The target NFV-MANO functional entity is operating normally.	The software version of the target NFV-MANO functional entity corresponds to the destination software version if the software modification was successful; it corresponds to the source software version in the case of a successful software rollback.

5.3.3.5 Flow description for the software modification

Table 5.3.3.5-1 provides the flow description for the successful completion of an NFV-MANO software modification without internal support.

Table 5.3.3.5-1: Use case flow description for software modification

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if adequate resources are available for the software modification process as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the source NFV-MANO functional entity, including identifying a restoration point and starting special logs enabling rollback (see notes 1 and 3).
2	SMM	SMM collects the information needed to perform the software modification (e.g. network configuration parameters). This can be part of the execution plan, requested from an administrator or from the source NFV-MANO functional entity.

#	Actor/Role-condition	Action/Description
3	SMM	According to the execution plan and using some of the collected information, SMM instantiates the destination NFV-MANO functional entity with its functional entity application in the STARTED_LOCKED state. The capacity, NFV-MANO service interface, peering information are set so that when it is unlocked, it will be able to replace the source NFV-MANO functional entity (see note 2).
4	SMM -> Tester	SMM invokes the tester to verify the health of the destination NFV-MANO functional entity. SMM also indicates the appropriate resources the tester can use for execution (see note 2).
5	Tester	The tester executes the test suites appropriate to determine the health of the destination NFV-MANO functional entity.
6	Tester -> SMM	The tester reports back that the destination NFV-MANO functional entity is healthy. If appropriate, not needed resources are released (see note 2).
7	SMM -> Destination NFV-MANO functional entity	SMM transfers the operational state data (e.g. authentication key) from the source NFV-MANO functional entity to the destination NFV-MANO functional entity (see note 2).
8	SMM, destination NFV-MANO functional entity, impacted NFV-MANO functional entity	SMM unlocks the functional entity application of destination NFV-MANO functional entity and starts the NFV-MANO service interfaces towards impacted NFV-MANO functional entities. Authentication of the destination NFV-MANO functional entity to the impacted NFV-MANO functional entity (see note 2).
9	SMM, destination NFV-MANO functional entity, impacted entity	SMM starts the NFV-MANO service interfaces towards impacted entities. Authentication of the destination NFV-MANO functional entity to the impacted entity (see note 2).
10	SMM, source NFV-MANO functional entity	SMM gracefully stops the NFV-MANO service interfaces towards impacted entities (see note 2).
11	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution (see note 2).
12	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the destination NFV-MANO functional entity and the system as a whole (see note 2).
13	Tester -> SMM	The tester reports that the software modification of the destination NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 2).
14	SMM -> Source NFV-MANO functional entity	When the SHUTDOWN_UNLOCKED state is reached, SMM terminates the source NFV-MANO functional entity (see note 2).
Ends when	SMM	SMM commits the software modification process by releasing the resources that were used to protect the software modification process and that enabled the software rollback (see note 3).

NOTE 1: The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the source NFV-MANO functional entity and its context in such a way that this source NFV-MANO functional entity can be restored from this restoration point with enough context to resume operation even after a crash.

NOTE 2: If SMM detects a failure of the destination NFV-MANO functional entity, e.g. it crashed, returned with an error, or the testing in step 6 or 13 has failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it executes a software rollback described in clause 5.3.6 or the fallback procedure described in clause 5.9.

NOTE 3: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, these resources are released with this information making software rollback not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.

5.3.3.6 Flow description for the software rollback

Table 5.3.3.6-1 provides the flow description for a software rollback executed by the SMM. The first step executed in this flow depends on the step of the software modification flow at which the SMM decides to roll back the software modification, for example, because of a failure. The "At #" column indicates this step of the software modification flow of table 5.3.3.5-1.

Table 5.3.3.6-1: Use case flow description for software rollback

#	Actor/Role-condition	Action/Description	At #
Begins when	SMM	SMM decides to start a software rollback.	
1	SMM	SMM checks if adequate resources are available for rolling back the software modification process as indicated in the execution plan of the software modification (see note 1).	
2	SMM	SMM collects the information needed to perform the software rollback (e.g. network configuration parameters). This can be part of the execution plan, requested from an administrator or from the destination NFV-MANO functional entity, or the information collected at step 2 of the software modification flow can be stored for the case the software modifications need to be rolled back (see note 2).	
3	SMM	According to the execution plan and using some of the collected information, SMM instantiates the source NFV-MANO functional entity with its functional entity application in the STARTED_LOCKED state. The capacity, NFV-MANO service interface, peering information are set so that when it is unlocked, it will be able to replace the source NFV-MANO functional entity (see note 3).	14
4	SMM -> Tester	SMM invokes the tester to verify the health of the source NFV-MANO functional entity.	
5	Tester	The tester executes the test suites appropriate to determine the health of the source NFV-MANO functional entity (see note 3).	
6	Tester -> SMM	The tester reports back that the source NFV-MANO functional entity is healthy.	
7	SMM -> Source NFV-MANO functional entity	SMM transfers the operational state data (e.g. authentication key) from the destination NFV-MANO functional entity to the source NFV-MANO functional entity (see note 3).	7 to 13
8	SMM, source NFV-MANO functional entity, impacted NFV-MANO functional entity	SMM unlocks the functional entity application of source NFV-MANO functional entity and starts the NFV-MANO service interfaces towards impacted NFV-MANO functional entities. Authentication of the source NFV-MANO functional entity to the impacted NFV-MANO functional entity (see note 3).	
9	SMM, source NFV-MANO functional entity, impacted entity	SMM starts the NFV-MANO service interfaces towards impacted entities. Authentication of the source NFV-MANO functional entity to the impacted entity (see note 3).	
10	SMM, destination NFV-MANO functional entity	SMM gracefully stops certain NFV-MANO service interfaces towards impacted entities.	
11	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution.	
12	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the source NFV-MANO functional entity and the system as a whole.	
13	Tester -> SMM	The tester reports that the software rollback of the source NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 3).	
14	SMM -> Destination NFV-MANO functional entity	When the SHUTDOWN_UNLOCKED state is reached, SMM terminates the destination NFV-MANO functional entity.	3 to 5
Ends when	SMM	SMM commits the software rollback by releasing the resources that were used to protect the software modification process and enabled the software rollback (see note 4).	

#	Actor/Role-condition	Action/Description	At #
NOTE 1: The software rollback can only proceed successfully if adequate resources are available, therefore this check can always be part of the decision of initiating the software rollback regardless at which step the software rollback flow is entered.			
NOTE 2: None, some or the entire step of information collection can be performed before entering the software rollback flow depending on whether the entities requiring the information have already been terminated and whether SMM kept the information collected during the software modification for the software rollback.			
NOTE 3: If SMM detects a failure of the source NFV-MANO functional entity, e.g. it crashed, returned with an error, or the testing in step 6 or 13 has failed, SMM determines if a retry as described in clause 5.8 is appropriate, otherwise it initiates the fallback procedure described in clause 5.9.			
NOTE 4: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, these resources are released with this information making software rollback not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.			

5.3.3.7 Flow description for the software modification with ongoing LCM/VRM operation

Table 5.3.3.7-1 provides the flow description for the successful completion of an NFV-MANO software modification without internal support while an VNF LCM operation, NS LCM operation, and/or VRM operation is ongoing when the software modification process is triggered. This use case is supported by the requirements in clause 5.6.2 of ETSI GS NFV-IFA 010 [i.7].

Table 5.3.3.7-1: Use case flow description for software modification with ongoing LCM operation

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if an VNF LCM operation, NS LCM operation, and/or VRM operation is currently ongoing. If yes, depending on the type of the NS LCM operation, VNF LCM operation and/or VRM operation, the operator's configuration, and the capabilities of the NFV-MANO functional entity to be modified, an appropriate action is taken (see note 4). Appropriate actions include: • the software modification is immediately continued with step 2 and is executed in parallel to the ongoing operation; or • the software modification is suspended until the NS LCM operation, VNF LCM operation and/or VRM operation is completed.
2	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the source NFV-MANO functional entity, including identifying a restoration point (see note 1).
3	SMM	SMM collects the information needed to perform the software modification (e.g. network configuration parameters). This can be part of the execution plan, requested from an administrator or from the source NFV-MANO functional entity.
4	SMM	According to the execution plan and using some of the collected information, SMM instantiates the destination NFV-MANO functional entity with its functional entity application in the STARTED_LOCKED state. The capacity, NFV-MANO service interface, peering information are set so that when it is unlocked, it will be able to replace the source NFV-MANO functional entity (see note 2).
5	SMM -> Tester	SMM invokes the tester to verify the health of the destination NFV-MANO functional entity. SMM also indicates the appropriate resources the tester can use for execution (see note 2).
6	Tester	The tester executes the test suites appropriate to determine the health of the destination NFV-MANO functional entity.

#	Actor/Role-condition	Action/Description
7	Tester -> SMM	The tester reports back that the destination NFV-MANO functional entity is healthy. If appropriate, not needed resources are released (see note 2).
8	SMM -> Destination NFV-MANO functional entity	SMM transfers the operational state data (e.g. authentication key) from the source NFV-MANO functional entity to the destination NFV-MANO functional entity (see note 2).
9	SMM, destination NFV-MANO functional entity, impacted NFV-MANO functional entity	SMM unlocks the functional entity application of destination NFV-MANO functional entity and starts the NFV-MANO service interfaces towards impacted NFV-MANO functional entities. Authentication of the destination NFV-MANO functional entity to the impacted NFV-MANO functional entity (see note 2).
10	SMM, destination NFV-MANO functional entity, impacted entity	SMM starts the NFV-MANO service interfaces towards impacted entities. Authentication of the destination NFV-MANO functional entity to the impacted entity (see note 2).
11	SMM, source NFV-MANO functional entity	SMM gracefully stops the NFV-MANO service interfaces towards impacted entities (see note 2).
12	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution (see note 2).
13	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the destination NFV-MANO functional entity and the system as a whole (see note 2).
14	Tester -> SMM	The tester reports that the software modification of the destination NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 2).
15	SMM -> Source NFV-MANO functional entity	When the SHUTDOWN_UNLOCKED state is reached, SMM terminates the source NFV-MANO functional entity (see note 2).
Ends when	SMM	SMM commits the software modification process by releasing the resources that were used to protect the software modification process and that enabled the software rollback (see note 3).

NOTE 1: The restoration point can be an existing restoration point, in which case no further actions are taken; or it can be a new one created as a backup of the source NFV-MANO functional entity and its context in such a way that this source NFV-MANO functional entity can be restored from this restoration point with enough context to resume operation even after a crash.

NOTE 2: If SMM detects a failure of the destination NFV-MANO functional entity, e.g. it crashed, returned with an error; or the testing in step 7 or 14 has failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8; or it executes a software rollback described in clause 5.3.3.6 or the fallback procedure described in clause 5.9.

NOTE 3: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, these resources are released with this information making software rollback not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.

NOTE 4: SMM is aware of the related capabilities of the NFV-MANO functional entity as well as the operator's configuration via some sort of descriptors or policies provided by the functional entity and the operator, respectively.

5.3.3.8 Flow description for the software modification with parallel LCM/VRM operation

Table 5.3.3.8-1 provides the flow description for the successful completion of an NFV-MANO software modification without internal support when a VNF LCM operation, NS LCM operation and/or VRM operation is triggered during the software modification process. In comparison to the flow in clause 5.3.3.7, the flow in this clause also supports to delay or reject an LCM/VRM operation that is triggered while the software modification is already ongoing. This use case is supported by the requirements in clause 5.6.2 of ETSI GS NFV-IFA 010 [i.7].

Table 5.3.3.8-1: Use case flow description for software modification with parallel LCM operation

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the source NFV-MANO functional entity, including identifying a restoration point (see note 1).
2	SMM	At any time during the software modification (step 2 to step 14), SMM learns about a new NS LCM operation, VNF LCM operation and/or VRM operation that was triggered. Depending on the type of the NS LCM operation, VNF LCM operation and/or VRM operation, the operator's configuration, and the capabilities of the NFV-MANO functional entity to be modified, an appropriate action is taken (see note 5). Appropriate actions include: • execute the operation in parallel with the software modification process of the NFV-MANO functional entity, • delay the execution of the operation until the NFV-MANO software modification process has completed, • notify the pending VNF LCM operation to the consumer such that the consumer can trigger the VNF LCM operation again after the completion of the software modification process; or • suspend the ongoing software modification process at an appropriate step (see use case in clause 5.7) until the triggered operation has completed.
3	SMM	SMM collects the information needed to perform the software modification (e.g. network configuration parameters). This can be part of the execution plan, requested from an administrator or from the source NFV-MANO functional entity.
4	SMM	According to the execution plan and using some of the collected information, SMM instantiates the destination NFV-MANO functional entity with its functional entity application in the STARTED_LOCKED state. The capacity, NFV-MANO service interface, peering information are set so that when it is unlocked, it will be able to replace the source NFV-MANO functional entity (see note 2).
5	SMM -> Tester	SMM invokes the tester to verify the health of the destination NFV-MANO functional entity. SMM also indicates the appropriate resources the tester can use for execution (see note 2).
6	Tester	The tester executes the test suites appropriate to determine the health of the destination NFV-MANO functional entity.
7	Tester -> SMM	The tester reports back that the destination NFV-MANO functional entity is healthy. If appropriate, not needed resources are released (see note 2).
8	SMM -> Destination NFV-MANO functional entity	SMM transfers the operational state data (e.g. authentication key) from the source NFV-MANO functional entity to the destination NFV-MANO functional entity (see note 2).
9	SMM, destination NFV-MANO functional entity, impacted NFV-MANO functional entity	SMM unlocks the functional entity application of destination NFV-MANO functional entity and starts the NFV-MANO service interfaces towards impacted NFV-MANO functional entities. Authentication of the destination NFV-MANO functional entity to the impacted NFV-MANO functional entity (see note 2).
10	SMM, destination NFV-MANO functional entity, impacted entity	SMM starts the NFV-MANO service interfaces towards impacted entities. Authentication of the destination NFV-MANO functional entity to the impacted entity (see note 2).
11	SMM, source NFV-MANO functional entity	SMM gracefully stops the NFV-MANO service interfaces towards impacted entities (see note 2).
12	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution (see note 2).

#	Actor/Role-condition	Action/Description
13	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the destination NFV-MANO functional entity and the system as a whole (see note 2).
14	Tester -> SMM	The tester reports that the software modification of the destination NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 2).
15	SMM -> Source NFV-MANO functional entity	When the SHUTDOWN_UNLOCKED state is reached, SMM terminates the source NFV-MANO functional entity (see note 2).
16	SMM	SMM commits the software modification process by releasing the resources that were used to protect the software modification process and that enabled the software rollback (see note 3).
Ends when	SMM	Depending on the action taken in step 2, SMM will trigger the execution of the pending NS LCM, VNF LCM and/or VRM operation; or SMM will notify the consumer such that the consumer can again trigger the operation that had not yet been executed.
NOTE 1: The restoration point can be an existing restoration point, in which case no further actions are taken; or it can be a new one created as a backup of the source NFV-MANO functional entity and its context in such a way that this source NFV-MANO functional entity can be restored from this restoration point with enough context to resume operation even after a crash. NOTE 2: If SMM detects a failure of the destination NFV-MANO functional entity, e.g. it crashed, returned with an error; or the testing in step 7 or 14 has failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8; or it executes a software rollback described in clause 5.3.3.6 or the fallback procedure described in clause 5.9. NOTE 3: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, these resources are released with this information making software rollback not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process. NOTE 4: SMM is aware of the related capabilities of the NFV-MANO functional entity as well as the operator's configuration via some sort of descriptors or policies provided by the functional entity and the operator, respectively.		

5.3.4 Special considerations for NFVO

The NFVO software modification use case is obtained by using the generic use case defined in clause 5.3 after replacing the generic roles described in clause 5.3.3.2 by the NFVO-related roles indicated in table 5.3.4-1 below.

Table 5.3.4-1: NFVO use case actors and roles

#	Role described in clause 5.3.2.2	Role related to the NFVO use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Source NFV-MANO functional entity	Source NFVO instance
3	Destination NFV-MANO functional entity	Destination NFVO instance
4	Impacted NFV-MANO functional entity	Other NFVO instances (if applicable), VNFM instance(s), VIM instance(s)
5	Impacted entity	OSS
6	Tester	Tester

5.3.5 Special considerations for VNFM

The VNFM software modification use case is obtained by using the generic use case defined in clause 5.3 after replacing the generic roles described in clause 5.3.3.2 by the VNFM-related roles indicated in table 5.3.5-1 below.

Table 5.3.5-1: VNFM use case actors and roles

#	Role described in clause 5.3.2.2	Role related to the VNFM use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Source NFV-MANO functional entity	Source VNFM instance
3	Destination NFV-MANO functional entity	Destination VNFM instance
4	Impacted NFV-MANO functional entity	NFVO instance, VIM instance(s)
5	Impacted entity	VNF instance(s), EM(s)
6	Tester	Tester

5.3.6 Special considerations for VIM

5.3.6.1 Actors and role mapping

The VIM software modification use case is obtained by using the generic use case defined in clause 5.3 after replacing the generic roles described in clause 5.3.3.2 by the VIM-related roles indicated in table 5.3.6.1-1 below.

Table 5.3.6.1-1: VIM use case actors and roles

#	Role described in clause 5.3.2.2	Role related to the VIM use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Source NFV-MANO functional entity	Source VIM instance
3	Destination NFV-MANO functional entity	Destination VIM instance
4	Impacted NFV-MANO functional entity	NFVO instance, VNFM instance(s)
5	Impacted entity	NFVI resources
6	Tester	Tester

5.3.6.2 Special considerations for VIM deployed using OpenStack®

OpenStack® (see note) consists of independent components for the management of the different types of resources (compute, networking, storage, etc.) as well as for the operation and the management of the OpenStack® platform itself (database, messaging, user management, automation, etc.). The software of all of these components can be modified separately and independently. In general, there is no need to change resource assignments to VNFs during the modification process. This means that running VNFs are not necessarily affected.

NOTE: The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. ETSI is not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

Due to limitations in hardware availability, it is not guaranteed that the destination NFV-MANO functional entity (i.e. VIM) in step 3 of the flow description in clause 5.3.3.5 can be instantiated in parallel to the source NFV-MANO functional entity (i.e. VIM).

Since OpenStack® ("Train" release) does not support the state model described in the annex A (clause A.2.1.4) of ETSI GS NFV-IFA 031 [i.2], the SMM cannot execute the operations as described in clause 5.3.3.5 directly on OpenStack® or its components.

It is noteworthy that the transfer of operational state data as described in step 7 of the general procedure in clause 5.3.3.5 is specific to the versions of the involved OpenStack® components.

Suspension, retry, rollback and fallback capabilities of OpenStack® are expected to be taken into account for the NFV-MANO software modification.

5.4 Software modification of multiple NFV-MANO functional entities

5.4.1 Introduction

The two previous clauses 5.2 and 5.3 have described standalone software modification for NFV-MANO functional entities in two types of situations, i.e. with or without internal support. In some cases, e.g. changes related to (or including) interfaces, the situation may become complex. Figure 5.4.1-1 shows the example of three NFV-MANO functional entities (i.e. NFVO, VNFM and VIM) with their reference points.

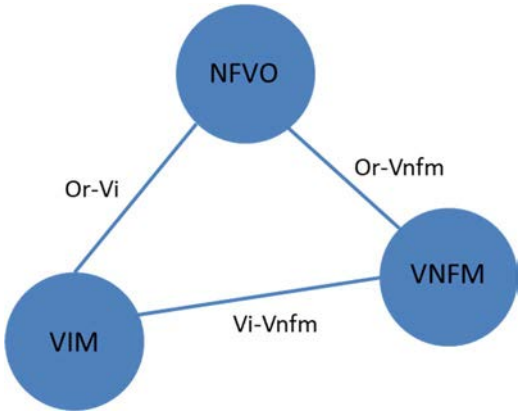

Figure 5.4.1-1: Example of NFV-MANO functional entities with their reference points

The different use cases of software modification for these functional entities are as follows:

1) Functional entities modification without impact on their interfaces, e.g. upgrade of NFVO with no changes to the interfaces supported on the reference points Or-Vi and Or-Vnfm - no new use case is needed, i.e. clauses 5.2 and 5.3 are applicable.

2) Modification of the interface between two functional entities, e.g. upgrade of the interface between NFVO and VNFM (i.e. interfaces supported on the reference point Or-Vnfm). With respect to the interface(s) between the two functional entities:

   - The new interface is backward compatible for both functional entities, i.e. NFVO and VNFM - no new use case is needed, i.e. clauses 5.2 and 5.3 are applicable.

   - The new interface is not backward compatible for at least one functional entity, i.e. NFVO or VNFM, a new use case is needed and introduced in clause 5.4.2.

It is noteworthy that the present document does not deal with situations in which more than two NFV-MANO functional entities are involved in a single software modification procedure.

5.4.2 Simultaneous software modification of two NFV-MANO functional entities

5.4.2.1 Introduction

Clause 5.4.2 presents the procedure of simultaneous software modification (and software rollback) of two NFV-MANO functional entities. This procedure can be used to avoid compatibility issues that can occur during the software modification (software rollback) between two NFV-MANO functional entities, between whom the interface has changed in a not fully backward compatible way.

To handle this case, the software modification (software rollback) of the NFV-MANO functional entities starts out as it would for an individual NFV-MANO functional entity. However, to ensure compatibility between the two NFV-MANO functional entities, a synchronization step is inserted in the flow in which the SMM plays a key role. This synchronization step can only be performed when both NFV-MANO functional entities are up and running with a compatible version, but with locked interfaces.

When this is achieved, the SMM can interconnect the two compatible NFV-MANO functional entities by starting the interfaces between them. Thus, it is ensured that the NFV-MANO functional entities only interact with an instance of the compatible version. Once the correct versions of the NFV-MANO functional entities are put in contact, the software modification (software rollback) can complete as for individual NFV-MANO functional entities by gracefully stopping the instances of the old versions.

5.4.2.2 Actors and roles

Table 5.4.2.2-1 describes the actors and roles of the use case.

Table 5.4.2.2-1: Use case actors and roles

#	Role	Description
1	Software Modification Manager (SMM)	Entity deciding to launch and managing the software modification process. Such an entity could be the OSS or an administrator.
2	Source NFV-MANO functional entity	Two NFV-MANO functional entities running their respective source version of the planned software modification (see note 1). The two NFV-MANO functional entities have a peering relation, for which compatibility is a concern during the software modification process (see note 2).
3	Destination NFV-MANO functional entity	Two NFV-MANO functional entities running their respective destination version of the planned software modification (see note 1). The two NFV-MANO functional entities have a peering relation, for which compatibility is a concern during the software modification process (see note 2).
4	Impacted NFV-MANO functional entity	Another NFV-MANO functional entity, which has a peering relation to the target NFV-MANO functional entities (see note 1). Multiple NFV-MANO functional entities may exist with such peering relationship to the target NFV-MANO functional entities. The compatibility of these peering relations is not jeopardized by the software modification process.
5	Impacted entity	Entity of a non-NFV-MANO functional block interacting with the target NFV-MANO functional entities (see note 1), e.g. a VNF instance managed by a VNFM or a virtualised resource managed by a VIM. It is noteworthy that the target NFV-MANO functional entities may interact with multiple "impacted entities".
6	Tester	Entity capable of running a test, e.g. functional test, to determine that the target NFV-MANO functional entities (see note 1) are healthy and operating as expected. There could be multiple tests the target NFV-MANO functional entities need to pass.

NOTE 1: The source and destination NFV-MANO functional entities are commonly referred to as target NFV-MANO functional entities when the emphasis is on the fact that they are targeted by the software modification process. The number of target NFV-MANO functional entities and the software version they are running depend on the stage of the execution.

NOTE 2: The source version of either NFV-MANO functional entities is not compatible with the destination version of the other NFV-MANO functional entity.

5.4.2.3 Pre-conditions

Table 5.4.2.3-1 describes the use case pre-conditions.

Table 5.4.2.3-1: Use case pre-conditions

#	Pre-condition	Additional description
1	A destination software version is available for each of the target NFV-MANO functional entities	The source version of either NFV-MANO functional entities is not compatible with the destination version of the other NFV-MANO functional entity.
2	SMM is operating normally and has the software modification execution plan	The execution plan for the software modification process is available for SMM. SMM knows the steps and how to interact with the source and the destination versions of the target NFV-MANO functional entities for the purpose of the software modification process.

#	Pre-condition	Additional description
3	Source and impacted NFV-MANO functional entities as well as the impacted entities are normally operating	In addition, the two target NFV-MANO functional entities are able to communicate with each other.
4	Resources are available	Any compute/storage/network resources needed for the execution of the software modification of the target NFV-MANO functional entities and their testing are available. No assumption is made on the kind of resources needed and available, i.e. whether they are physical, cloud resources, or otherwise.
5	Test suites are available	Test suites for the source and the destination software versions of the target NFV-MANO functional entities are available.

5.4.2.4 Post-conditions

Table 5.4.2.4-1 describes the use case post-conditions.

Table 5.4.2.4-1: Use case post-conditions

#	Post-condition	Additional description
1	SMM, the impacted NFV-MANO functional entities, as well as the impacted entities are operating normally	SMM, the impacted entities as well as the impacted NFV-MANO functional entities can interact with the target NFV-MANO functional entities, i.e. the target NFV-MANO functional entities fulfils their duties towards them.
2	The target NFV-MANO functional entities are operating normally and are able to communicate with each other	The software version of the target NFV-MANO functional entities corresponds to the destination software version if the software modification was successful; it corresponds to the source software version in the case of a successful software rollback.

5.4.2.5 Flow description for the software modification

Table 5.4.2.5-1 provides the flow description for a successful completion of the simultaneous software modification of two NFV-MANO functional entities.

Table 5.4.2.5-1: Use case flow description for software modification

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the two source NFV-MANO functional entities, including identifying their restoration point(s) (see note 1).
2	SMM	SMM collects the information needed to perform the software modification (e.g. network configuration parameters). This can be part of the execution plan, requested from an administrator or from the source NFV-MANO functional entities.
3	SMM	According to the execution plan and using some of the collected information, SMM instantiates the two destination NFV-MANO functional entities with their respective functional entity application in the STARTED_LOCKED state. The capacity, NFV-MANO service interface, peering information are set so that when they are unlocked, the destination NFV-MANO functional entities will be able to replace the source NFV-MANO functional entities (see note 2).
4	SMM -> Tester	SMM invokes the tester to verify the health of the destination NFV-MANO functional entities. SMM also indicates the appropriate resources the tester can use for execution (see note 2).
5	Tester	The tester executes the test suites appropriate to determine the health of the destination NFV-MANO functional entities.
6	Tester -> SMM	The tester reports back that the destination NFV-MANO functional entities are healthy. If appropriate, not needed resources are released (see note 2).

#	Actor/Role-condition	Action/Description
7	SMM -> Destination NFV-MANO functional entities	SMM transfers the respective operational state data (e.g. authentication key) from the source NFV-MANO functional entities to the respective destination NFV-MANO functional entities (see note 2).
8	SMM, destination NFV-MANO functional entities	SMM unlocks the functional entity application of the destination NFV-MANO functional entities and starts the NFV-MANO service interface between these two destination NFV-MANO functional entities. Respective authentication of the two destination NFV-MANO functional entities (see note 2).
9	SMM, destination NFV-MANO functional entities, impacted NFV-MANO functional entity	SMM starts the NFV-MANO service interfaces towards the impacted NFV-MANO functional entities. Authentication of the destination NFV-MANO functional entities to the impacted NFV-MANO functional entities (see note 2).
10	SMM, destination NFV-MANO functional entities, impacted entity	SMM starts the NFV-MANO service interfaces towards the impacted entities. Authentication of the destination NFV-MANO functional entities to the impacted entities (see note 2).
11	SMM, source NFV-MANO functional entities	SMM gracefully stops the NFV-MANO service interfaces towards the impacted entities (see note 2).
12	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution (see note 2).
13	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the destination NFV-MANO functional entities and the system as a whole (see note 2).
14	Tester -> SMM	The tester reports that the software modification of the destination NFV-MANO functional entities has been completed successfully. Not needed resources are released (see note 2).
15	SMM -> Source NFV-MANO functional entities	When the SHUTDOWN_UNLOCKED state is reached, SMM terminates the source NFV-MANO functional entities (see note 2).
Ends when	SMM	SMM commits the software modification process by releasing the resources that were used to protect the software modification process and that enabled the software rollback (see note 3).

NOTE 1: The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as backup of the source NFV-MANO functional entities and their context in such a way that the source NFV-MANO functional entities can be restored from this restoration point with enough context to resume operation even after a crash.

NOTE 2: If SMM detects a failure of a destination NFV-MANO functional entity, e.g. any of the two NFV-MANO functional entities crashed, returned with an error, or the testing in step 6 or 14 has failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it executes for both destination NFV-MANO functional entities a software rollback as described in clause 5.4.2.6, or the fallback procedure as described in clause 5.9.

NOTE 3: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, these resources are released with this information making software rollback not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.

5.4.2.6 Flow description for the software rollback

Table 5.4.2.6-1 provides the flow description for a software rollback executed by the SMM in the case of simultaneous software modification of two NFV-MANO functional entities.

Table 5.4.2.6-1: Use case flow description for software rollback

#	Actor/Role-condition	Action/Description	At #
Begins when	SMM	SMM decides to start a software rollback.	
1	SMM	SMM checks if adequate resources are available for rolling back the software modification as indicated in the execution plan of the software modification process (see note 1).	
2	SMM	SMM collects the information needed to perform the software rollback (e.g. network configuration parameters). This can be part of the execution plan, requested from an administrator or from the destination NFV-MANO functional entities, or the information collected at step 2 of the software modification flow can be stored for the case the software modifications need to be rolled back (see note 2).	
3	SMM	According to the execution plan and using some of the collected information, SMM instantiates the source NFV-MANO functional entities with their respective functional entity application in the STARTED_LOCKED state. The capacity, NFV-MANO service interface, peering information are set so that when it is unlocked, they will be able to replace the source NFV-MANO functional entities (see note 3).	14
4	SMM -> Tester	SMM invokes the tester to verify the health of the source NFV-MANO functional entities.	
5	Tester	The tester executes the test suites appropriate to determine the health of the source NFV-MANO functional entities (see note 3).	
6	Tester -> SMM	The tester reports back that the source NFV-MANO functional entities are healthy.	
7	SMM -> Source NFV-MANO functional entities	SMM transfers the respective operational state data (e.g. authentication key) from the destination NFV-MANO functional entities to the source NFV-MANO functional entities (see note 3).	7 to 13
8	SMM, source NFV-MANO functional entities	SMM unlocks the functional entity application of the source NFV-MANO functional entities and starts the NFV-MANO service interface between these two source NFV-MANO functional entities. Respective authentication of the two source NFV-MANO functional entities (see note 3).	
9	SMM, source NFV-MANO functional entities, impacted NFV-MANO functional entity	SMM starts the NFV-MANO service interfaces towards the impacted NFV-MANO functional entities. Authentication of the source NFV-MANO functional entities to the impacted NFV-MANO functional entities (see note 3).	
10	SMM, source NFV-MANO functional entities, impacted entity	SMM starts the NFV-MANO service interfaces towards the impacted entities. Authentication of the source NFV-MANO functional entities to the impacted entities (see note 3).	
11	SMM, destination NFV-MANO functional entities	SMM gracefully stops certain NFV-MANO service interfaces towards the impacted entities.	
12	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution.	
13	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the source NFV-MANO functional entities and the system as a whole.	
14	Tester -> SMM	The tester reports that the software rollback of the source NFV-MANO functional entities has been completed successfully. Not needed resources are released (see note 3).	
15	SMM -> Destination NFV-MANO functional entities	When the SHUTDOWN_UNLOCKED state is reached, SMM terminates the destination NFV-MANO functional entities.	3 to 5
Ends when	SMM	SMM commits the software rollback by releasing the resources that were used to protect the software modification process and enabled the software rollback (see note 4).	

#	Actor/Role-condition	Action/Description	At #
NOTE 1:	The software rollback can only proceed successfully if adequate resources are available, therefore this check can always be part of the decision of initiating the software rollback, regardless at which step the software rollback flow is entered.		
NOTE 2:	None, some or the entire step of information collection can be performed before entering the software rollback flow depending on whether the entities requiring the information have already been terminated and whether SMM kept the information collected during the software modification for the software rollback.		
NOTE 3:	If SMM detects a failure of a source NFV-MANO functional entity, e.g. any of the two NFV-MANO functional entities crashed, returned with an error, or the testing in step 6 or 14 has failed, SMM checks if a retry, as described in clause 5.8, is appropriate, otherwise it initiates the fallback procedure for both destination NFV-MANO functional entities, as described in clause 5.9.		
NOTE 4:	Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, these resources are released with this information making software rollback not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.		

5.5 Change of the NFV-MANO deployed architectural option

5.5.1 Introduction

As described in clause 4.2.2, several architectural options are possible and available for implementing NFV-MANO functional entities. The goal of some software modifications can be to change the currently deployed architectural option to its alternative one, for example, moving away from the use of specific VNFMs and replacing them with one or more generic VNFMs.

In addition to the architectural changes, such changes can imply non-backward compatible API changes as well. Non-backward compatible API changes can also happen without architectural changes as discussed in clause 5.4, e.g. due to changes in the specifications, or standardizing APIs that were not standardized yet.

A common feature in many of these software modification processes is that they impact not only the NFV-MANO functional entity, but also the entities it manages and/or interacts with. Therefore, the software modification depends on the availability of new software versions for both the NFV-MANO functional entity and the entity(ies) it manages or interacts with so that they can interact properly after the completion of the software modifications.

This also means that the software modification process also needs to ensure that the dependency between the NFV-MANO functional entity and its managed entities is satisfied regardless whether the software modification completes successfully, or it needs to be rolled back due to some failure or an administrative decision. This clause collects such use cases.

5.5.2 Replacing a S-VNFM with a G-VNFM

5.5.2.1 Introduction

In this use case, as shown on the left-hand side of figure 5.5.2.1-1, the assumption is that the VNFs (VNF1v1, VNF2v1) are managed by specific VNFMs (S-VNFM), S-VNFM1 and S-VNFM2 respectively. They might use proprietary interfaces indicated by the dashed line between the VNFM and its managed VNF. A new version of VNF2, namely VNF2v2, has become available, which will allow VNF2 to interact with a generic VNFM (G-VNFM) as reflected on the right-hand side by the solid lines between VNF2v2 and the G-VNFM. Therefore, the goal of the software modification is to transition the subsystem from the S-VNFM managed deployment of VNF2 to the G-VNFM managed one.

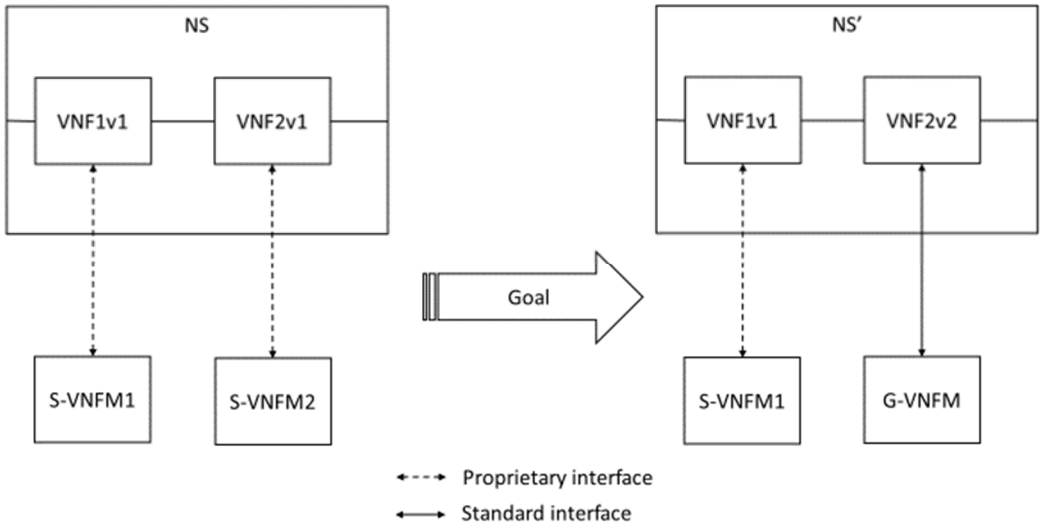

Figure 5.5.2.1-1: Use case of replacing a S-VNFM with a G-VNFM with standard interface

The process of transitioning can be accomplished (see figure 5.5.2.1-2) by instantiating first a G-VNFM, or finding one with enough capacity, to manage the destination version of VNF2, i.e. VNF2v2. This allows the instantiation of the destination VNF2v2 as it can only interact with, and be managed by, a G-VNFM.

Then, to avoid service impact, the destination VNF2v2 can be added to the NS (resulting in NS") in parallel to the source VNF2v1 path, and the traffic can be directed from VNF1v1 to both VNF2v1 and VNF2v2 instances according to an appropriate schedule. For example, new requests can be sent to the new destination VNF2v2, while old request are served up to completion by the source VNF2v1. Once all the traffic is served by the new destination VNF2v2, the source VNF2v1 can be removed together with its managing S-VNFM2.

Later, at a moment in time when a similar new version for the software of VNF1 becomes available, the same software modification procedure can be repeated for VNF1 transferring its management to the G-VNFM as well.

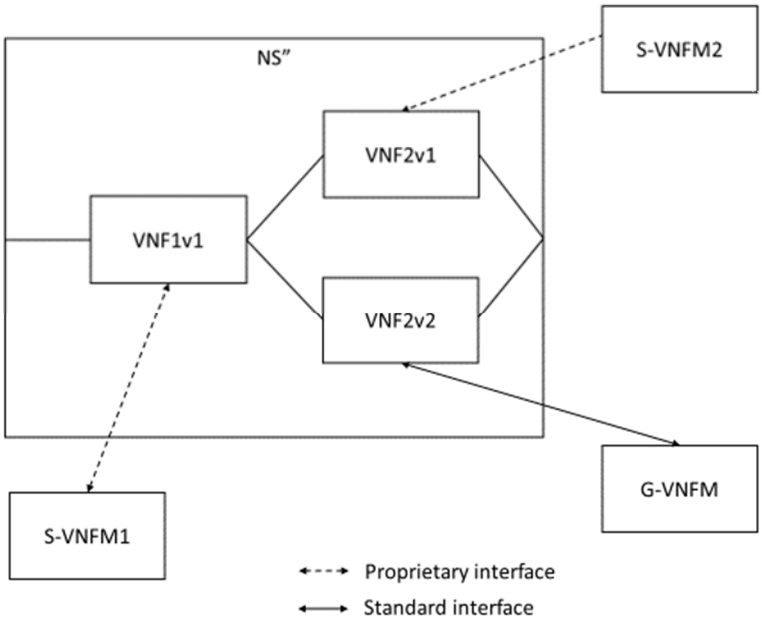

Figure 5.5.2.1-2: In the transitional phase the S-VNFM2 and the G-VNFM operate in parallel managing two different versions of VNF2 belonging to NS"

Note that the architectural change, that is the functional split between the VNF2v1 and S-VNFM2 is hidden throughout the modification process and it is irrelevant. The key point is that the new version of VNF2, that is VNF2v2, includes all the special functionality that might have been implemented before by S-VNFM2, and it is now capable of interacting with a G-VNFM. Thus, the architectural change is reduced to a non-backward compatible interface change, which is handled by the simultaneous software modification of VNF2 and its managing VNFM.

5.5.2.2 Actors and roles

Table 5.5.2.2-1 describes the actors and roles of the use case.

Table 5.5.2.2-1: Use case actors and roles

#	Role	Description
1	Software Modification Manager (SMM)	Entity deciding to launch and managing the software modification process. Such an entity could be the OSS or an administrator.
2	NFVO	NFVO managing the NS whose VNF is managed by the S-VNFM.
3	S-VNFM	Specific VNFM managing the source VNF via a proprietary interface.
4	G-VNFM	Generic VNFM managing the destination VNF via the standard interface
5	Source VNF	VNF managed by the S-VNFM via a proprietary interface.
6	Destination VNF	VNF managed by the G-VNFM via the standard interface.

5.5.2.3 Pre-conditions

Table 5.5.2.3-1 describes the use case pre-conditions.

Table 5.5.2.3-1: Use case pre-conditions

#	Pre-condition	Additional description
1	Destination VNF software version is available	The destination VNF software version has been delivered for the source VNF and it references the G-VNFM as manager. Other external interfaces remain the same as of the source VNF.
2	G-VNFM software version is available	There is a VNFM software version available implementing the standard interface towards the VNF
3	SMM is operating normally and has the software modification execution plan	The execution plan for the software modification process is available for SMM. SMM knows the steps and how to interact with NFVO, S-VNFM and G-VNFM for the purpose of the software modification.
4	S-VNFM and source VNF are operating normally	N/A
5	Resources are available	Any compute/storage/network resources needed for the execution of the software modification of the VNF and the VNFM are available. No assumption is made on the kind of resources needed and available, i.e. whether they are physical, cloud resources, or otherwise.

5.5.2.4 Post-conditions

Table 5.5.2.4-1 describes the use case post-conditions.

Table 5.5.2.4-1: Use case post-conditions

#	Post-condition	Additional description
1	VNF is operating normally	In case of successful completion of the software modification, the VNF corresponds to the destination software version. If the software modification was successfully rolled back, the VNF corresponds to the source software version.
2	VNFM is operating normally.	In case of successful completion of the software modification the VNFM is a Generic VNFM (G-VNFM). If the software modification was successfully rolled back the VNFM is a Specific VNFM (S-VNFM).

5.5.2.5 Flow description for the software modification

Table 5.5.2.5-1 provides the flow description for a successful completion of the software modification which consists of replacing the S-VNFM with the G-VNFM and their managed VNF.

Table 5.5.2.5-1: Use case flow description for software modification

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM takes appropriate steps to prepare the system for the planned software modification process of the source VNF and the S-VNFM including identifying a restoration point and starting special logs enabling the software rollback (see notes 1 and 4).
2	SMM	SMM checks if a G-VNFM is available to manage an additional VNF, otherwise SMM instantiates a new G-VNFM instance and verifies that it is operational (see notes 2 and 5).
3	SMM -> G-VNFM	If a new G-VNFM is instantiated, SMM sets the peering information in the G-VNFM and in NFVO. SMM unlocks the G-VNFM (see note 2).
4	G-VNFM <-> NFVO	If needed, G-VNFM peers with NFVO (see note 2).
5	SMM -> NFVO	SMM updates the NSD by adding a parallel path, which includes the destination VNF referencing the G-VNFM, and associates the NSD with the NS instance.
6	NFVO -> G-VNFM, VIM	NFVO (based on the reference) requests the G-VNFM to instantiate the destination VNF and the VIM to instantiate the virtual networks needed (see note 2).
7	G-VNFM, VIM -> NFVO	G-VNFM and VIM report back to NFVO the instantiation of the destination VNF and its virtual networks (see note 2).
8	NFVO -> SMM	NFVO reports to SMM the completion of the deployment of the modified NS.
9	SMM	SMM performs the steps necessary to redirect the traffic to the destination VNF according to the schedule indicated in the execution plan (see notes 2 and 3).
10	SMM -> NFVO	When no traffic is handled by the source VNF anymore, SMM updates the NSD by removing the path with the source VNF, and associates the NSD with the NS instance (see notes 2 and 6).
11	NFVO -> S-VNFM, VIM	NFVO requests the S-VNFM to terminate the source VNF and the VIM to remove the virtual networks not needed anymore.
12	S-VNFM, VIM -> NFVO	S-VNFM and VIM report back to NFVO the removal of the source VNF and its virtual networks.
13	NFVO->SMM	NFVO reports to SMM the completion of the deployment of the modified NS.
14	SMM	SMM locks and terminates the S-VNFM if it does not manage any other VNF instance.
15	SMM	SMM performs the steps necessary to verify that the VNF and the G-VNFM are operating normally (see notes 2 and 5).
Ends when	SMM	SMM commits the software modification process by releasing the resources that were used to protect the software modification and that enabled the software rollback (see notes 2 and 4).

#	Actor/Role-condition	Action/Description
NOTE 1:	The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the source VNF, its S-VNFM and their context in such a way that this source VNF and its S-VNFM can be restored from this restoration point with enough context to resume operation even after a crash.	
NOTE 2:	If SMM detects a failure of the destination VNF or the G-VNFM, e.g. any of them crashes, returns with an error, or the testing fails (see note 5), SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it can execute a software rollback described in clause 5.5.2.6, or needs to initiate the fallback procedure described in clause 5.9.	
NOTE 3:	The steps and the schedule of redirecting the traffic from the source to the destination VNF can be different depending on the NS and its functionality. SMM might need to interact directly or indirectly with different entities such as an EM, an SDN controller, etc. to carry it out.	
NOTE 4:	Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, these resources are released with this information making software rollback not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.	
NOTE 5:	As part of the verification, SMM can request a tester to execute some test cases to evaluate whether the instantiated entity (entities) and/or the system as a whole operate(s) as expected.	
NOTE 6:	SMM uses the criteria specified in the execution plan to determine when the source VNF instance stops handling traffic. This can include monitoring or other means.	

5.5.2.6 Flow description for the software rollback

Table 5.5.2.6-1 provides the flow description for a software rollback executed by the SMM. The first step depends on when (at which step) in the flow of table 5.5.2.5-1 the SMM decides about the software rollback. This step is indicated in the last column of the table.

Table 5.5.2.6-1: Use case flow description for software rollback

#	Actor/Role-condition	Action/Description	At #
Begins when	SMM	SMM decides to launch the software rollback.	
1	SMM	SMM takes appropriate steps to prepare the system for the software rollback of the destination VNF and the G-VNFM.	15
2	SMM	SMM instantiates a new S-VNFM instance and verifies that it is operational (see notes 1 and 3).	
3	SMM -> S-VNFM	SMM sets the peering information in the S-VNFM and in NFVO. SMM unlocks the S-VNFM (see note 1).	
4	S-VNFM <-> NFVO	S-VNFM peers with NFVO (see note 1).	
5	SMM -> NFVO	SMM updates the NSD by adding a parallel path, which includes the source VNF referencing the S-VNFM and associates the NSD with the NS instance.	10
6	NFVO -> S-VNFM, VIM	NFVO (based on the reference) requests the S-VNFM to instantiate the source VNF and the VIM to instantiate the virtual networks needed (see note 1).	
7	S-VNFM, VIM -> NFVO	S-VNFM and VIM report back to NFVO the instantiation of the source VNF and its virtual networks (see note 1).	
8	NFVO -> SMM	NFVO reports to SMM the completion of the deployment of the modified NS (see note 1).	
9	SMM	SMM performs the steps necessary to redirect the traffic to the source VNF according to the schedule indicated in the execution plan (see notes 1 and 2).	9
10	SMM -> NFVO	When no traffic is handled by the destination VNF instance anymore, SMM updates the NSD by removing the path with the destination VNF instance and associates the NSD with the NS instance (see notes 1 and 4).	
11	NFVO -> G-VNFM, VIM	NFVO requests the G-VNFM to terminate the destination VNF and the VIM to remove the virtual networks not needed anymore.	6, 7
12	G-VNFM, VIM -> NFVO	G-VNFM and VIM report back to NFVO the removal of the destination VNF and its virtual networks.	
13	NFVO -> SMM	NFVO reports to SMM the completion of the deployment of the modified NS.	

#	Actor/Role-condition	Action/Description	At #
14	SMM	SMM locks and terminates the G-VNFM if it is not used for any other VNF.	2, 3, 4
15	SMM	SMM performs the steps necessary to verify that the source VNF and the S-VNFM are operating normally (see notes 1 and 3).	
Ends when	SMM	SMM commits the software modification process by releasing the resources that were used to protect the software modification and that enabled the software rollback (see note 1).	

NOTE 1: If SMM detects a failure of the source VNF or the S-VNFM, e.g. any of them crashes, returns with an error, or the testing fails (see note 3), SMM determines if a retry as described in clause 5.8 is appropriate, otherwise it executes the fallback procedure described in clause 5.9.
NOTE 2: The steps and the schedule of redirecting the traffic from the destination to the source VNF can be different depending on the NS and its functionality. SMM might need to interact directly or indirectly with different entities such as an EM, an SDN controller, etc. to carry it out.
NOTE 3: As part of the verification, SMM can request a tester to execute some test cases to evaluate whether the instantiated entity (entities) and/or the system as a whole operate(s) as expected.
NOTE 4: SMM uses the criteria specified in the execution plan to determine when the source VNF stops handling traffic. This can include monitoring or other means.

5.6 Special considerations for containerised architectures

5.6.1 Introduction

Clause 5.3 of ETSI GR NFV-IFA 029 [i.5] provides five examples of the utilization of containers in NFV. In these examples (see clause A.2), the containers created by the Container Infrastructure Services (CISs) are managed by a Container Infrastructure Service Management (CISM) considered as a functionality.

The CISM functionality is viewed as composed of two management functions. The first manages Managed Container Infrastructure Objects (MCIO) such as Kubernetes®'s Pods, while the second manages virtualised resources exposed by CIS, the container runtime environment. In clause 7.2.4 of ETSI GR NFV-IFA 029 [i.5], different options are envisioned to map these two CISM functions to the NFV architectural framework (see clause A.3). These options will be considered in the following clauses for NFV-MANO software modification.

5.6.2 Case of software modification with internal support

5.6.2.1 Introduction

As shown in clause A.3, different ways to integrate the CISM functionality into the NFV picture are possible:

- in the current NFV-MANO architecture (in the VIM or across VNFM and VIM);
- as a distinct functional entity (standalone functional entity or replacing VIM and VNFM);
- embedded into a VNF with support for shared container service.

All these example scenarios are considered in the subsequent clauses for the NFV-MANO functional entities with internal support for software modification. As seen in clause 5.2, this internal support aspect encompasses both seamless support (clause 5.2.2) and non-seamless support (clause 5.2.3). This distinction has no importance in the discussion of clause 5.6.2.

Therefore, the generic use case defined in clause 5.2.2 is used as the 'template' whose generic roles defined for each functional entity (clause 5.2.2.2) will be mapped in the following clauses to the appropriate roles. This mapping for each example scenario is realized on the basis of the related special considerations for NFVO, VNFM and VIM (clauses 5.2.4, 5.2.5 and 5.2.6).

5.6.2.2 CISM functionality embedded in the VIM example scenario

- NFVO: unchanged (see table 5.2.4-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- VNFM: unchanged (see table 5.2.5-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- VIM: unchanged (see table 5.2.6-1) with the addition to this table of another "Impacted entity" (CIS instance(s)).

5.6.2.3 CISM functionality distributed across VNFM and VIM example scenario

The two CISM management functions, i.e. MCIO management and container-specific virtual resource management, can be independent from each other or not. In the first case, the software modification of VNFM (respectively VIM) which hosts the MCIO management (respectively container-specific virtual resource management) can be processed independently from its counterpart, i.e. VIM (respectively VNFM). The mapping is then as follows:

- NFVO: unchanged (see table 5.2.4-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- VNFM: unchanged (see table 5.2.5-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (Container-specific virtual resource management of the CISM instance(s)) and with the addition to this table of another "Impacted entity" (MCIO(s)).
- VIM: unchanged (see table 5.2.6-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (MCIO management of the CISM instance(s)) and with the addition to this table of another "Impacted entity" (CIS instance(s)).

In the second case, modifying one of the two NFV-MANO functional entities (i.e. VNFM or VIM) cannot be done independently. This use case is analysed in clause 5.4 "Software modification of multiple NFV-MANO functional entities".

5.6.2.4 CISM functionality as a standalone functional entity example scenario

- NFVO: unchanged (see table 5.2.4-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- VNFM: unchanged (see table 5.2.5-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- VIM: unchanged (see table 5.2.6-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- CISM: this new functional entity uses the 'template' provided in clause 5.2.2 with the following actors and roles (table 5.6.2.4-1).

Table 5.6.2.4-1: CISM use case actors and roles

#	Role described in clause 5.2.2.2	Role related to the use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Target NFV-MANO functional entity	CISM instance(s)
3	Impacted NFV-MANO functional entity	NFVO instance, VNFM instance(s), VIM instance(s)
4	Impacted entity	NFVI resources, MCIO instance(s), CIS instance(s)
5	Tester	Tester

5.6.2.5 CISM functionality-only replacing VIM and VNFM example scenario

This new functional entity in an architecture without VIM and VNFM uses the 'template' provided in clause 5.2.2 with the following actors and roles (table 5.6.2.5-1).

Table 5.6.2.5-1: CISM use case actors and roles

#	Role described in clause 5.2.2.2	Role related to the use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Target NFV-MANO functional entity	CISM instance(s)
3	Impacted NFV-MANO functional entity	NFVO instance
4	Impacted entity	VNF instance(s), MCIO instance(s), CIS instance(s), EM(s), NFVI resources
5	Tester	Tester

5.6.2.6 CISM functionality embedded into VNF with support for shared container service example scenario

From the software modification perspective, the CISM functionality is embedded in the VNF; therefore, an appropriate software modification procedure of the VNF applies. However, since the CISM functionality is also exposed to the NFV-MANO, from the NFV-MANO viewpoint, this is similar to the use case of the CISM functionality deployed as a standalone functional entity discussed in clause 5.6.2.4. Accordingly:

- NFVO: unchanged (see table 5.2.4-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s) embedded in a VNF) and another "Impacted entity" (VNF embedding the CISM) reflecting the two aspects of the embedded CISM.
- VNFM: unchanged (see table 5.2.5-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s) embedded in a VNF) and another "Impacted entity" (VNF embedding the CISM).
- VIM: unchanged (see table 5.2.6-1).
- VNF embedding the CISM instance(s): this new functional entity uses the 'template' provided in clause 5.2.2 with the following actors and roles (table 5.6.2.6-1). Triggering on the target NFV-MANO functional entity, the software modification means triggering the appropriate software modification procedure on the VNF embedding the CISM instance(s), for example, initiating the Change Current VNF Package operation for the VNF.

Table 5.6.2.6-1: CISM use case actors and roles

#	Role described in clause 5.2.2.2	Role related to the use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Target NFV-MANO functional entity	VNF embedding the CISM instance(s)
3	Impacted NFV-MANO functional entity	VNFM instance(s), VIM instance(s)
4	Impacted entity	NFVI resources, MCIO instance(s), CIS instance(s)
5	Tester	Tester

5.6.2.7 CISM functionality embedded into VNF without support for shared container service example scenario

From the software modification perspective, the CISM functionality is not visible and has no impact on the NFV-MANO modification use cases.

5.6.3 Case of software modification without internal support

5.6.3.1 Introduction

Just as for the case of software modification with seamless support, the different example scenarios for integrating the CISM functionality into the NFV picture which are shown in clause A.3 will be considered in the subsequent clauses for the NFV-MANO functional entities without internal support for software modification.

In what follows, the generic use case defined in clause 5.3 is used as the 'template' whose generic roles defined for each functional entity (clause 5.3.2.2) will be mapped in the subsequent clauses to the appropriate roles. This mapping for each example scenario is realized on the basis of the related special considerations for NFVO, VNFM, and VIM (clauses 5.3.4, 5.3.5, and 5.3.6).

5.6.3.2 CISM functionality embedded in the VIM example scenario

- NFVO: unchanged (see table 5.3.4-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- VNFM: unchanged (see table 5.3.5-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- VIM: unchanged (see table 5.3.6.1-1) with the addition to this table of another "Impacted entity" (CIS instance(s)).

5.6.3.3 CISM functionality distributed across VNFM and VIM example scenario

The two CISM management functions, i.e. MCIO management and container-specific virtual resource management, can be independent from each other or not. In the first case, the software modification of VNFM (respectively VIM) which hosts the MCIO management (respectively container-specific virtual resource management) can be processed independently from its counterpart, i.e. VIM (respectively VNFM). The mapping is then as follows:

- NFVO: unchanged (see table 5.3.4-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- VNFM: unchanged (see table 5.3.5-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (Container-specific virtual resource management of the CISM instance(s)) and with the addition to this table of another "Impacted entity" (MCIO(s)).
- VIM: unchanged (see table 5.3.6.1-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (MCIO management of the CISM instance(s)) and with the addition to this table of another "Impacted entity" (CIS instance(s)).

In the second case, modifying one of the two NFV-MANO functional entities (i.e. VNFM or VIM) cannot be done independently. This use case is analysed in clause 5.4.

5.6.3.4 CISM functionality as a standalone functional entity example scenario

- NFVO: unchanged (see table 5.3.4-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- VNFM: unchanged (see table 5.3.5-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- VIM: unchanged (see table 5.3.6.1-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s)).
- CISM: this new functional entity uses the 'template' provided in clause 5.3.2.2 with the following actors and roles (table 5.6.3.4-1).

Table 5.6.3.4-1: CISM use case actors and roles

#	Role described in clause 5.3.2.2	Role related to the use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Source NFV-MANO functional entity	Source CISM instance(s)
3	Destination NFV-MANO functional entity	Destination CISM instance(s)
4	Impacted NFV-MANO functional entity	NFVO instance, VNFM instance(s), VIM instance(s)
5	Impacted entity	NFVI resources, MCIO instance(s), CIS instance(s)
6	Tester	Tester

5.6.3.5 CISM functionality-only replacing VIM and VNFM example scenario

This new functional entity in an architecture without VIM and VNFM uses the 'template' provided in clause 5.3.2.2 with the following actors and roles (table 5.6.3.5-1).

Table 5.6.3.5-1: CISM use case actors and roles

#	Role described in clause 5.3.2.2	Role related to the use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Source NFV-MANO functional entity	Source CISM instance(s)
3	Destination NFV-MANO functional entity	Destination CISM instance(s)
4	Impacted NFV-MANO functional entity	NFVO instance
5	Impacted entity	VNF instance(s), MCIO instance(s), CIS instance(s), EM(s), NFVI resources
6	Tester	Tester

5.6.3.6 CISM functionality embedded into VNF with support for shared container service example scenario

From the software modification perspective, the CISM functionality is embedded in the VNF; therefore, an appropriate software modification procedure of VNFs applies. However, since the CISM functionality is exposed to the NFV-MANO, from the NFV-MANO viewpoint, this is similar to the use case of the CISM functionality deployed as a standalone functional entity (see clause 5.6.3.4). Accordingly:

- NFVO: unchanged (see table 5.3.4-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s) embedded in a VNF) and another "Impacted entity" (VNF embedding the CISM) reflecting the two aspects of the embedded CISM.
- VNFM: unchanged (see table 5.3.5-1) with the addition to this table of another "Impacted NFV-MANO functional entity" (CISM instance(s) embedded in a VNF) and another "Impacted entity" (VNF embedding the CISM).
- VIM: unchanged (see table 5.3.6.1-1).
- VNF embedding the CISM instance(s): this new functional entity uses the 'template' provided in clause 5.3.2.2 with the following actors and roles (table 5.6.3.6-1). Accordingly, the lifecycle management operations performed on the source/destination NFV-MANO functional entities are performed on the source/destination VNFs embedding the CISM functionality.

Table 5.6.3.6-1: CISM use case actors and roles

#	Role described in clause 5.3.2.2	Role related to the use case
1	Software Modification Manager (SMM)	Software Modification Manager (SMM)
2	Source NFV-MANO functional entity	Source VNF embedding the CISM instance(s)
3	Destination NFV-MANO functional entity	Destination VNF embedding the CISM instance(s)
4	Impacted NFV-MANO functional entity	VNFM instance(s), VIM instance(s)
5	Impacted entity	NFVI resources, MCIO instance(s), CIS instance(s)
6	Tester	Tester

5.6.3.7 CISM functionality embedded into VNF without support for shared container service example scenario

From the software modification perspective, the CISM functionality is not visible and has no impact on the NFV-MANO modification use cases.

5.7 Status monitoring and suspend operation

5.7.1 NFV-MANO functional entities with internal support for their software modification

5.7.1.1 Introduction

Completing a software modification process can take a significant amount of time during which the operational conditions can change (e.g. resources needed for the software modification process become unavailable due to some failure, or operations that were blocked by the software modification process become urgent). Such a change may require a decision on whether the software modification process can complete in due course, or it needs to be suspended at this time and to be completed later. In certain cases, it might not be desirable to complete the started software modification. Instead, the executed changes could be rolled back.

This means that one needs to be able to monitor the status of an ongoing software modification process and, if it is deemed necessary, request that the activities associated with the software modification process are halted temporarily in a state which does not jeopardize the operation of the system. Later, when the conditions allow for it, the software modification process can resume so that it can be completed. Alternatively, it could also be rolled back.

Providing the capabilities of monitoring the status of a software modification process, suspending then resuming it, are especially important for a long running software modification of NFV-MANO functional entities with internal support for this operation. There could be different approaches to monitoring the status of an ongoing software modification:

1) an NFV-MANO functional entity can provide a query interface, which is pulled each time the status information is required; or

2) an NFV-MANO functional entity can report the status of an ongoing of software modification on a regular basis or at specific steps of the software modification, or in special circumstances.

It is desirable that the supplier of the NFV-MANO functional entity defines steps or stages for the software modification procedure based on which the SMM can decide whether suspending the software modification is possible, and when it is appropriate. That is, if suspending at a given step or stage imposes constraints, these constraints need to be part of the description. For example, there can be steps at which suspending is not possible and the software modification needs to complete the step before the suspend request can take effect.

For example, if an NFV-MANO functional entity is composed of redundant MANO-functional entity components/redundancy units and the software modification is a rolling upgrade procedure, the suspension is appropriate at the completion of the upgrade of one MANO-functional entity component/redundancy unit and before starting the upgrade of the next MANO-functional entity component/redundancy unit.

In addition, there could be situations in which the NFV-MANO functional entity with internal software modification support requires external assistance to resolve some issues preventing the continuation of the execution of the requested software modification.

For certain software modifications, suspension may not be possible at all, yet the NFV-MANO functional entity can still report the progress of the software modification according to the defined steps.

According to these considerations, the use case presented in clause 5.7.1 for the status monitoring and suspend operation uses as a basis the software modification flow of the use case for NFV-MANO functional entities with seamless internal support described in clause 5.2.2.5. Two scenarios are shown. In the first one, it is assumed that the NFV-MANO functional entities provide a query interface, which is pulled each time the status information is required. In the second scenario, the target NFV-MANO functional entity reports its status when it encounters an external issue, which it cannot resolve by itself. Hence, it suspends the software modification to pass the control to the SMM to resolve the issue. When the SMM passes the control back to the NFV-MANO functional entity, it can request it to continue the execution or to undo the completed part of the software modification. The undo scenario is shown in the flow.

5.7.1.2 Actors and roles

Table 5.7.1.2-1 describes the actors and roles of the use case.

Table 5.7.1.2-1: Use case actors and roles

#	Role	Description
1	Software Modification Manager (SMM)	Entity managing the software modification process including launching, querying and suspending/resuming the process. Such an entity could be the OSS or an administrator.
2	Target NFV-MANO functional entity	Running NFV-MANO functional entity undergoing the software modification process.
3	Impacted NFV-MANO functional entity	Another NFV-MANO functional entity, which has a peering relation to the target NFV-MANO functional entity. Multiple NFV-MANO functional entities may exist with such peering relationship to the target NFV-MANO functional entities.
4	Impacted entity	Entity of a non-NFV-MANO functional block interacting with the target NFV-MANO functional entity, e.g. a VNF instance managed by a VNFM or a virtualised resource managed by a VIM. It is noteworthy that the target NFV-MANO functional entity may interact with multiple "impacted entities".
5	Tester	Entity capable of running a test, e.g. functional test, to determine that the target NFV-MANO functional entity is healthy and operating as expected. There could be multiple tests the target NFV-MANO functional entity needs to pass.

5.7.1.3 Pre-conditions

Table 5.7.1.3-1 describes the use case pre-conditions.

Table 5.7.1.3-1: Use case pre-conditions

#	Pre-condition	Additional description
1	A destination software version is available for the target NFV-MANO functional entity	The destination software version has been delivered for the target NFV-MANO functional entity.
2	SMM is operating normally and has the software modification execution plan	The execution plan for the software modification process is available for SMM. SMM knows the steps and how to interact with the source and the destination versions of the target NFV-MANO functional entity for the purpose of the software modification process.
3	Targeted and impacted NFV-MANO functional entities, as well as the impacted entities, are operating normally	N/A
4	Resources are available	Any compute/storage/network resources needed for the execution of the software modification process of the target NFV-MANO functional entity and its testing are available. No assumption is made on the kind of resources needed and available, i.e. whether they are physical, cloud resources, or otherwise.
5	Test suites are available	Test suites for the source and the destination software versions of the target NFV-MANO functional entity are available.

5.7.1.4 Post-conditions

Table 5.7.1.4-1 describes the use case post-conditions.

Table 5.7.1.4-1: Use case post-conditions

#	Post-condition	Additional description
1	SMM, the impacted NFV-MANO functional entities, as well as the impacted entities are operating normally	SMM, the impacted entities as well as the impacted NFV-MANO functional entities can interact with the target NFV-MANO functional entity, i.e. the target NFV-MANO functional entity fulfils its duties towards them.
2	The target NFV-MANO functional entity is operating normally.	The software version of the target NFV-MANO functional entity corresponds to the destination software version if the software modification was successful; it corresponds to the source software version in the case of a successful software rollback.

5.7.1.5 Flow description for the query and suspend operations

Table 5.7.1.5-1 provides the flow description for the status query followed by the suspend and resume operations of an NFV-MANO software modification with internal support.

Table 5.7.1.5-1: Use case flow description for query and suspend

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the target NFV-MANO functional entity, including identifying a restoration point and starting special logs enabling software rollback (see notes 1 and 4).
2	SMM -> Target NFV-MANO functional entity	SMM requests the target NFV-MANO functional entity to proceed with the software modification with indication, if needed, of the appropriate resources it can use for the execution (see note 3).
3	Target NFV-MANO functional entity	The target NFV-MANO functional entity starts executing the software modification according to its internal procedures.
4	SMM -> Target NFV-MANO functional entity	Due to changes in some conditions, SMM needs to query the target NFV-MANO functional entity about the status of the ongoing software modification (see notes 3 and 5).
5	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM the status of the ongoing software modification (see notes 3 and 5).
6	SMM -> Target NFV-MANO functional entity	Based on the received status and other information, SMM requests the target NFV-MANO functional entity to suspend the ongoing software modification (see note 3).
7	Target NFV-MANO functional entity	The target NFV-MANO functional entity continues the software modification until a state is reached in which suspending the operation is safe according to its internal procedures.
8	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM that the ongoing software modification has been suspended (see note 3).
9	SMM -> Target NFV-MANO functional entity	Once the conditions change and allow to continue the software modification, SMM requests the target NFV-MANO functional entity to resume the software modification with indication, if needed, of the appropriate resources it can use for the execution (see note 3).
10	Target NFV-MANO functional entity	The target NFV-MANO functional entity resumes and completes the execution of the software modification according to its internal procedures. It also releases resources that are not needed any more.
11	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM that the software modification has been completed (see notes 2 and 3).
12	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution.
13	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the target NFV-MANO functional entity and the system as a whole.
14	Tester -> SMM	The tester determines that the software modification of the target NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 3).
Ends when	SMM	SMM commits the software modification process and releases resources that were used to protect the software modification and enable software rollback (see note 4).

NOTE 1: The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the target NFV-MANO functional entity and its context in such a way that this target NFV-MANO functional entity can be restored to this restoration point with enough context to resume operation even after a crash.

NOTE 2: If the target NFV-MANO functional entity has deployed the destination software version successfully, it reports that the software modification was completed successfully. In case the target NFV-MANO functional entity encountered a failure but was able to roll back to the source version successfully, the target NFV-MANO functional entity reports that it has completed the software modification with automatic rollback to the source software version. In this case, SMM can decide to retry the operation.

#	Actor/Role-condition	Action/Description
NOTE 3:	If SMM detects a failure of the target NFV-MANO functional entity, e.g. it crashed, it returned with an error other than in note 2, or the testing in step 14 failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it executes a software rollback described in clause 5.2.2.6 or the fallback procedure described in clause 5.9.	
NOTE 4:	Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, software rollback becomes not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.	
NOTE 5:	The target NFV-MANO functional entity can provide status reports on the ongoing software modification on a regular basis or at specific steps of the software modification. SMM can make its decision about the suspension based on these status reports instead of querying the target NFV-MANO functional entity.	

5.7.1.6 Flow description for the status reporting with suspend operation

Table 5.7.1.6-1 provides the flow description for the status reporting with suspension operation due to an issue external to the NFV-MANO functional entity. The control is given back to the SMM which can ask NFV-MANO functional entity to resume operation once the problem is resolved, or request the NFV-MANO functional entity to undo completed part of the software modification. The following flow describes the undo of already completed parts.

Table 5.7.1.6-1: Use case flow description for status reporting with suspend

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the target NFV-MANO functional entity, including identifying a restoration point and starting special logs enabling software rollback (see notes 1 and 4).
2	SMM -> Target NFV-MANO functional entity	SMM requests the target NFV-MANO functional entity to proceed with the software modification with indication, if needed, of the appropriate resources it can use for the execution (see note 3).
3	Target NFV-MANO functional entity	The target NFV-MANO functional entity starts executing the software modification according to its internal procedures.
4	Target NFV-MANO functional entity -> SMM	Due to changes in some conditions, the target NFV-MANO functional entity encounters an external issue for which it requires assistance. Therefore, it suspends the software modification at a state in which suspending the operation is safe and reports this status to SMM requesting its assistance (see notes 3 and 5).
5	SMM	Based on the received status and other information, SMM performs the actions it deems necessary to assist the NFV-MANO functional entity.
6	SMM -> Target NFV-MANO functional entity	Since the conditions have changed and they do not allow to continue the software modification, SMM requests the target NFV-MANO functional entity to undo the executed part of the software modification with indication, if needed, of the appropriate resources it can use for the execution.
7	Target NFV-MANO functional entity	The target NFV-MANO functional entity resumes the software modification to undo the executed changes according to its internal procedures. It also releases resources that are not needed any more.
8	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM that the software modification has been undone (see notes 2 and 3).
9	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution.
10	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the target NFV-MANO functional entity and the system as a whole.

#	Actor/Role-condition	Action/Description
11	Tester -> SMM	The tester determines that the software modification of the target NFV-MANO functional entity has been completed and the source version software has been redeployed successfully. Not needed resources are released (see note 3).
Ends when	SMM	SMM commits the software modification process and releases resources that were used to protect the software modification and enable software rollback (see note 4).

NOTE 1: The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the target NFV-MANO functional entity and its context in such a way that this target NFV-MANO functional entity can be restarted from this restoration point with enough context to resume operation even after a crash.

NOTE 2: If the target NFV-MANO functional entity has deployed the destination software version successfully, it reports that the software modification was completed successfully. In case the target NFV-MANO functional entity encountered a failure but was able to roll back to the source version successfully, the target NFV-MANO functional entity reports that it has completed the software modification with automatic rollback to the source software version. In this case, SMM can decide to retry the operation.

NOTE 3: If SMM detects a failure of the target NFV-MANO functional entity, e.g. it crashed, it returned with an error other than in note 2, or the testing in step 14 failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it executes a software rollback described in clause 5.2.2.6 or the fallback procedure described in clause 5.9.

NOTE 4: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, software rollback becomes not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.

NOTE 5: The target NFV-MANO functional entity can provide status reports on the ongoing software modification on a regular basis, at specific steps of the software modification or under specific circumstances. If the specific circumstance requires external assistance the NFV-MANO functional entity can suspend the ongoing software modification to wait for the assistance.

5.7.2 NFV-MANO functional entities without internal support for their software modification

5.7.2.1 Introduction

In case the NFV-MANO functional entity has no internal support for the software modification, the process is fully managed by the SMM. This means that the status query and suspension of the software modification process need to be fulfilled by the SMM. With respect to the status, the SMM can use the steps of the flows described in clause 5.3 as a reference together with the states of the NFV-MANO functional entities, their application and NFV-MANO service interfaces.

Similarly, the suspension can be performed at the boundaries of these steps. In addition, if a step is performed by another entity such as a 'Tester', it is possible that this entity also supports the suspend operation, in which case the SMM can use it as well to fulfil the suspension request faster. The use case described in the subsequent clauses describes such a scenario of suspension of the software modification in the testing step.

Note that it is possible that, after suspension, it is determined that it is not desirable any more to complete the started software modification, e.g. due to some changes in the system. In this case, the SMM can enter the rollback flow corresponding to the executed software modification at the step appropriate for the suspension.

5.7.2.2 Actors and roles

Table 5.7.2.2-1 describes the actors and roles of the use case.

Table 5.7.2.2-1: Use case actors and roles

#	Role	Description
1	Software Modification Manager (SMM)	Entity managing the software modification process.
2	Source NFV-MANO functional entity	NFV-MANO functional entity running the source version of the planned software modification (see note).
3	Destination NFV-MANO functional entity	NFV-MANO functional entity running the destination version of the planned software modification (see note).
4	Impacted NFV-MANO functional entity	Another NFV-MANO functional entity, which has a peering relation to the target NFV-MANO functional entity. Multiple NFV-MANO functional entities may exist with such peering relationship to the target NFV-MANO functional entities.
5	Impacted entity	Entity of a non-NFV-MANO functional block interacting with the target NFV-MANO functional entity, e.g. a VNF instance managed by a VNFM or a virtualised resource managed by a VIM. It is noteworthy that the target NFV-MANO functional entity may interact with multiple "impacted entities".
6	Tester	Entity capable of running a test, e.g. functional test, to determine that the target NFV-MANO functional entity is healthy and operating as expected. There could be multiple tests the target NFV-MANO functional entity needs to pass.
7	External Manager	Entity launching, querying and suspending/resuming the software modification process. Such an entity could be the OSS or an administrator.
NOTE: The source and destination NFV-MANO functional entities are commonly referred to as target NFV-MANO functional entity (entities) when the emphasis is on the fact that they are targeted by the software modification process. The number of target NFV-MANO functional entities and the software version they are running depend on the stage of the execution.		

5.7.2.3 Pre-conditions

Table 5.7.2.3-1 describes the use case pre-conditions.

Table 5.7.2.3-1: Use case pre-conditions

#	Pre-condition	Additional description
1	A destination software version is available for the target NFV-MANO functional entity	The destination software version has been delivered for the target NFV-MANO functional entity.
2	SMM is operating normally and has the software modification execution plan	The execution plan for the software modification process is available for SMM. SMM knows the steps and how to interact with the source and the destination versions of the target NFV-MANO functional entity for the purpose of the software modification process.
3	Targeted and impacted NFV-MANO functional entities as well as the impacted entities are operating normally	N/A
4	Resources are available	Any compute/storage/network resources needed for the execution of the software modification process of the target NFV-MANO functional entity and its testing are available. No assumption is made on the kind of resources needed and available, i.e. whether they are physical, cloud resources, or otherwise.
5	Test suites are available	Test suites for the source and the destination software versions of the target NFV-MANO functional entity are available.

5.7.2.4 Post-conditions

Table 5.7.2.4-1 describes the use case post-conditions.

Table 5.7.2.4-1: Use case post-conditions

#	Post-condition	Additional description
1	SMM, the impacted NFV-MANO functional entities, as well as the impacted entities are operating normally	SMM, the impacted entities as well as the impacted NFV-MANO functional entities can interact with the target NFV-MANO functional entity, i.e. the target NFV-MANO functional entity fulfils its duties towards them.
2	The target NFV-MANO functional entity is operating normally.	The software version of the target NFV-MANO functional entity corresponds to the destination software version if the software modification was successful; it corresponds to the source software version in the case of a successful software rollback.

5.7.2.5 Flow description for the query and suspend operations

Table 5.7.2.5-1 provides the flow description for the status query followed by the suspend and resume operations of an NFV-MANO software modification without internal support. If it is not desirable to complete the software modification, the SMM can be requested (by the external manager) to roll it back by entering the corresponding rollback flow at the point of the suspension.

Table 5.7.2.5-1: Use case flow description for query and suspend

#	Actor/Role-condition	Action/Description
Begins when	External Manager	The external manager decides to launch the software modification process.
1	External Manager -> SMM	The external manager requests SMM to start the software modification process.
2	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the target NFV-MANO functional entity, including identifying a restoration point and starting special logs enabling software rollback (see notes 1 and 3).
3	SMM	SMM collects the information needed to perform the software modification (e.g. network configuration parameters). This can be part of the execution plan, requested from an administrator or from the source NFV-MANO functional entity.
4	SMM	According to the execution plan and using some of the collected information, SMM instantiates the destination NFV-MANO functional entity with its functional entity application in the STARTED_LOCKED state. The capacity, NFV-MANO service interface, peering information are set so that when it is unlocked, it will be able to replace the source NFV-MANO functional entity (see note 2).
5	SMM -> Tester	SMM invokes the tester to verify the health of the destination NFV-MANO functional entity. SMM also indicates the appropriate resources the tester can use for execution (see note 2).
6	Tester	The tester executes the test suites appropriate to determine the health of the destination NFV-MANO functional entity.
7	External Manager -> SMM , Tester	Due to changes in some conditions, SMM is queried about the status of the ongoing software modification process. SMM can query the tester for the percentage of completed/remaining testing activities and the remaining time (see note 2).
8	SMM -> External Manager	SMM reports back that the testing of the instantiated destination NFV-MANO functional entity with its functional entity application in the STARTED_LOCKED state is in progress and any further details are available (see note 2).
9	External Manager -> SMM	SMM is requested to suspend the software modification process.
10	SMM -> Tester	SMM requests the tester to suspend the testing activities (see note 4).

#	Actor/Role-condition	Action/Description
11	Tester -> SMM	The tester suspends the testing activities and reports that the testing was incomplete. If appropriate, not needed resources are released (see note 4).
12	SMM -> External Manager	SMM reports back that the software modification process was suspended at testing the instantiated destination NFV-MANO functional entity with its functional entity application in the STARTED_LOCKED state and any further details available (see note 4).
13	External Manager -> SMM	Once the conditions have changed and allow to continue the software modification, SMM is requested to resume the software modification (see note 2).
14	SMM -> Tester	SMM invokes the tester to continue the verification of the health of the destination NFV-MANO functional entity. SMM also indicates the appropriate resources the tester can use for execution (see note 2).
15	Tester	The tester executes the (remaining part of the) test suites appropriate to determine the health of the destination NFV-MANO functional entity.
16	Tester -> SMM	The tester reports back that the destination NFV-MANO functional entity is healthy. If appropriate, not needed resources are released (see note 2).
17	SMM -> Destination NFV-MANO functional entity	SMM transfers the operational state data (e.g. authentication key) from the source NFV-MANO functional entity to the destination NFV-MANO functional entity (see note 2).
18	SMM, destination NFV-MANO functional entity, impacted NFV-MANO functional entity	SMM unlocks the functional entity application of destination NFV-MANO functional entity and starts the NFV-MANO service interfaces towards impacted NFV-MANO functional entities. Authentication of the destination NFV-MANO functional entity to the impacted NFV-MANO functional entity (see note 2).
19	SMM, destination NFV-MANO functional entity, impacted entity	SMM starts the NFV-MANO service interfaces towards impacted entities. Authentication of the destination NFV-MANO functional entity to the impacted entity (see note 2).
20	SMM, source NFV-MANO functional entity	SMM gracefully stops certain NFV-MANO service interfaces towards impacted entities (see note 2).
21	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution (see note 2).
22	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the destination NFV-MANO functional entity and the system as a whole (see note 2).
23	Tester -> SMM	The tester reports that the software modification of the destination NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 2).
24	SMM -> Source NFV-MANO functional entity	When the SHUTDOWN_UNLOCKED state is reached, SMM terminates the source NFV-MANO functional entity (see note 2).
25	SMM -> External Manager	SMM reports the completion of the software modification process.
Ends when	External Manager -> SMM	External Manager commits the software modification process to SMM, which releases the resources that were used to protect the software modification and that enabled the software rollback (see note 3).

#	Actor/Role-condition	Action/Description
NOTE 1:	The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the target NFV-MANO functional entity and its context in such a way that this target NFV-MANO functional entity can be restored from this restoration point with enough context to resume operation even after a crash.	
NOTE 2:	If SMM detects a failure of the destination NFV-MANO functional entity, e.g. it crashed, it returned with an error, or the testing in step 11, 16 or 23 failed, SMM needs to decide whether a retry is appropriate as described in clause 5.8, or it executes a software rollback described in clause 5.3.3.6 or the fallback procedure described in clause 5.9.	
NOTE 3:	Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, software rollback becomes not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.	
NOTE 4:	If SMM detects a failure of the destination NFV-MANO functional entity while the software modification process is suspended, e.g. it crashes, SMM decides if fault isolation is necessary and from what step to continue with the software modification once it is resumed, i.e. whether a retry is appropriate as described in clause 5.8, or it executes a software rollback described in clause 5.3.3.6 or the fallback procedure described in clause 5.9.	
NOTE 5:	The external manager can decide that the software modification cannot be completed and needs to be rolled back. In this case, to execute the rollback, SMM enters the rollback flow of the software modification flow being executed at the step it was suspended.	

5.8 Retry

5.8.1 Introduction

To accomplish a software modification process, the SMM needs to execute a number of steps, which can consist of different operations. Some of these operations might not succeed on the first attempt raising the question whether the software modification process can continue. Problems preventing the completion of an operation typically can be classified into two main categories. They can be either permanent or transient. When the problem is transient, there is a good chance that by retrying the execution of the same operation, it will complete successfully. Thus, the software modification process can proceed without any further delay.

This is the purpose of the retry procedure described in clause 5.8. It is explained using an example use case, which is based on clause 5.2.2.5, the flow of the software modification use case of NFV-MANO functional entity with internal support. As opposed to clause 5.2.2.5, in this example use case, the tester fails to get hold of the resources that are required for the testing in spite of the SMM indicating them as available. This is a temporary problem as the SMM can provide more or other resources to tester and initiate a retry of the operation.

Such a retry procedure is not limited to this use case or to the software modification flow. It can be applied in the same way as described to other flows, including the software rollback. Moreover, a retry can be reattempted a number of times as long as the operation has no side effects, i.e. it is idempotent. However, to ensure completion of the software modification process, the number of retry attempts is usually limited to a small number.

5.8.2 Actors and roles

Table 5.8.2-1 describes the actors and roles of the use case.

Table 5.8.2-1: Use case actors and roles

#	Role	Description
1	Software Modification Manager (SMM)	Entity deciding to launch and managing the software modification process. Such an entity could be the OSS or an administrator.
2	Source NFV-MANO functional entity	NFV-MANO functional entity running the source version of the planned software modification (see note).
3	Destination NFV-MANO functional entity	NFV-MANO functional entity running the destination version of the planned software modification (see note).
4	Impacted NFV-MANO functional entity	Another NFV-MANO functional entity, which has a peering relation to the target NFV-MANO functional entity (see note). Multiple NFV-MANO functional entities may exist with such peering relationship to the target NFV-MANO functional entities.
5	Impacted entity	Entity of a non-NFV-MANO functional block interacting with the target NFV-MANO functional entity (see note), e.g. a VNF instance managed by a VNFM or a virtualised resource managed by a VIM. It is noteworthy that the target NFV-MANO functional entity may interact with multiple "impacted entities".
6	Tester	Entity capable of running a test, e.g. functional test, to determine that the target NFV-MANO functional entity (see note) is healthy and operating as expected. There could be multiple tests the target NFV-MANO functional entity needs to pass.
NOTE: The source and destination NFV-MANO functional entities are commonly referred to as target NFV-MANO functional entity (entities) when the emphasis is on the fact that they are targeted by the software modification process. The number of target NFV-MANO functional entities and the software version they are running depend on the stage of the execution.		

5.8.3 Pre-conditions

Table 5.8.3-1 describes the use case pre-conditions.

Table 5.8.3-1: Use case pre-conditions

#	Pre-condition	Additional description
1	A destination software version is available for the target NFV-MANO functional entity	The destination software version has been delivered for the target NFV-MANO functional entity.
2	SMM is operating normally and has the software modification execution plan	The execution plan for the software modification process is available for SMM. SMM knows the steps and how to interact with the source and the destination versions of the target NFV-MANO functional entity for the purpose of the software modification process.
3	Source and impacted NFV-MANO functional entities, as well as the impacted entities, are normally operating	N/A
4	Resources are available with some delay causing a transient error	Any compute/storage/network resources needed for the execution of the software modification process of the target NFV-MANO functional entity and its testing are available eventually. No assumption is made on the kind of resources needed and available, i.e. whether they are physical, cloud resources, or otherwise.
5	Test suites are available	Test suites for the source and the destination software versions of the target NFV-MANO functional entity are available.

5.8.4 Post-conditions

Table 5.8.4-1 describes the use case post-conditions.

Table 5.8.4-1: Use case post-conditions

#	Post-condition	Additional description
1	SMM, the impacted NFV-MANO functional entities, as well as the impacted entities are operating normally	SMM, the impacted entities as well as the impacted NFV-MANO functional entities can interact with the target NFV-MANO functional entity, i.e. the target NFV-MANO functional entity fulfils its duties towards them.
2	The target NFV-MANO functional entity is operating normally.	The software version of the target NFV-MANO functional entity corresponds to the destination software version if the software modification was successful; it corresponds to the source software version in the case of a successful software rollback.

5.8.5 Flow description for the retry procedure

Table 5.8.5-1 provides the flow description for the successful completion of an NFV-MANO software modification with internal support after a retry.

Table 5.8.5-1: Use case flow description for retry

#	Actor/Role-condition	Action/Description
Begins when	SMM	SMM decides to launch the software modification process.
1	SMM	SMM checks if adequate resources are available for the software modification as indicated in the execution plan of the software modification process. SMM takes appropriate steps to prepare the system for the planned software modification of the target NFV-MANO functional entity, including identifying a restoration point and starting special logs enabling software rollback (see notes 1 and 4).
2	SMM -> Target NFV-MANO functional entity	SMM requests the target NFV-MANO functional entity to proceed with the software modification with indication, if needed, of the appropriate resources it can use for the execution.
3	Target NFV-MANO functional entity	The target NFV-MANO functional entity executes the software modification according to its internal procedures. It also releases resources that are not needed any more.
4	Target NFV-MANO functional entity -> SMM	The target NFV-MANO functional entity reports back to SMM that the software modification has completed (see notes 2 and 3).
5	SMM -> Tester	SMM invokes the tester with indication of appropriate resources it can use for execution.
6	Tester	The tester fails to get hold of the indicated resources.
7	Tester -> SMM	The tester reports to SMM the failure to obtain the indicated resources.
8	SMM	SMM verifies the indicated resources and allocates new resources as necessary.
9	SMM -> Tester	SMM retries the invocation of the tester and indicates the appropriate resources that have been verified it can use for execution.
10	Tester	The tester runs all the test suites applicable for the verification of the health and the performance of the target NFV-MANO functional entity and the system as a whole.
11	Tester -> SMM	The tester determines that the software modification of the target NFV-MANO functional entity has been completed successfully. Not needed resources are released (see note 3).
Ends when	SMM	SMM commits the software modification process and releases resources that were used to protect the software modification and enable software rollback (see note 4).

#	Actor/Role-condition	Action/Description
NOTE 1: The restoration point can be an existing restoration point, in which case no further actions are taken, or it can be a new one created as a backup of the target NFV-MANO functional entity and its context in such a way that this target NFV-MANO functional entity can be restored from this restoration point with enough context to resume operation even after a crash.		
NOTE 2: If the target NFV-MANO functional entity has deployed the destination software version successfully, it reports that the software modification was completed successfully. In case the target NFV-MANO functional entity encountered a failure, but was able to roll back to the source version successfully, the target NFV-MANO functional entity reports that it has completed the software modification with automatic rollback to the source software version.		
NOTE 3: If SMM detects a failure of the target NFV-MANO functional entity, e.g. it crashed, it returned with an error other than in note 2, or the testing in step 11 failed, SMM needs to decide whether it executes a software rollback described in clause 5.2.2.6 or the fallback procedure described in clause 5.9.		
NOTE 4: Resources enabling the software rollback are used by SMM to store information necessary to revert the changes of the software modification without service impact through the software rollback such as log of executed operations, state/configuration of removed entities, etc. Once the software modification process is committed, software rollback becomes not possible. Instead, if necessary, a software modification process can be initiated where the destination software version is the same as the source software version of the just committed software modification process.		

5.9 Fallback

5.9.1 Introduction

Before starting a software modification process, it has to be verified that a restoration point has been created that can be used for a fallback. The restoration point needs to contain all data that is necessary to revert the NFV-MANO to a working state it was before the initiation of the software modification process. There is no requirement on when the restoration point has to be created. However, it needs to be mentioned that when a fallback to this restoration point is executed, all changes that have been made to the sub-system which is included in the restoration point after this restoration point has been created are lost.

NOTE 1: The NFV-MANO state may change as a result of NFV-MANO operations. For instance, the instantiation of a new NS may have changed the state data of the NFV-MANO while the software modification process was running. Two solutions can be used to avoid state loss: avoid LCM operations during the software modification process, or reapply the operations, for instance, by logging and replaying the involved LCM operations.

Since in case of a fallback, there is a high probability of losing data, fallbacks should be avoided as much as possible. Instead, in case a failure occurs during a software modification process, attempting a retry or a software rollback is preferred. If a retry and a software rollback are not possible, then a fallback is the only option to revert to the state stored in the restoration point before the software modification process has started. It is a last resort.

The advantage of a fallback is that it can be executed at any time. This allows the NFV-MANO or its functional entity to resume operation even in unexpected, not understood, situations not related to software modifications. The fallback is managed by a Fallback Manager. As opposed to the SMM, the Fallback Manager does not need to know about the steps of a previous software modification process.

NOTE 2: In many cases, the NFV-MANO functional entities share data or their data is interdependent. So it is advisable to ensure that the restoration point contains a consistent set of data so that all dependent NFV-MANO functional entities can be restored at the same time.

5.9.2 Actors and roles

Table 5.9.2-1 describes the actors and roles of the use case.

Table 5.9.2-1: Use case actors and roles

#	Role	Description
1	Fallback Manager	Entity executing the decision to fall back to a sub-system state contained in a restoration point. The reason for this decision may be a software modification process that could not be completed.
2	Fallback target NFV-MANO functional entity	NFV-MANO functional entity that needs to be set back (restored) to a previous state. There may be more than one fallback target NFV-MANO functional entity.
3	Impacted managed entity	Non-NFV-MANO entity that the fallback target NFV-MANO functional entity is managing, e.g. a VNF instance managed by a VNFM or a virtualised resource managed by a VIM. A fallback target NFV-MANO functional entity may interact with multiple "impacted managed entities".
4	Tester	Entity capable of running a test, e.g. functional test, to determine that the fallback target NFV-MANO functional entity is healthy and operating as expected. There could be multiple tests a fallback target NFV-MANO functional entity needs to pass before reaching the final decision.

5.9.3 Pre-conditions

Table 5.9.3-1 describes the use case pre-conditions.

Table 5.9.3-1: Use case pre-conditions

#	Pre-condition	Additional description
1	A restoration point of the fallback target NFV-MANO functional entity is available (restoration point)	If the restoration point contains state from different NFV-MANO functional entities, then the fallback applies to all these NFV-MANO functional entities.
2	Test suites are available	Test suites to verify the complete set of NFV-MANO services, including services not provided by fallback target NFV-MANO functional entities.

5.9.4 Post-conditions

Table 5.9.4-1 describes the use case post-conditions.

Table 5.9.4-1: Use case post-conditions

#	Post-condition	Additional description
1	The fallback target NFV-MANO functional entities and the impacted managed entities are operating normally	The NFV-MANO is operating on a consistent set of data.

5.9.5 Flow description for the fallback procedure

Table 5.9.5-1 provides the flow description for a fallback.

Table 5.9.5-1: Use case flow description for fallback

#	Actor/Role-condition	Action/Description
Begins when	Fallback Manager	Receives a request that a fallback has to be executed.
1	Fallback Manager-> Fallback target NFV-MANO functional entity	The fallback manager forcibly stops all fallback target NFV-MANO functional entities.
2	Fallback Manager	The fallback manager restores the state stored in the restoration point.

#	Actor/Role-condition	Action/Description
3	Fallback Manager -> Fallback target NFV-MANO functional entity	Restores the fallback target NFV-MANO functional entities according to the restored state and puts their functional entity application in the STARTED_LOCKED state.
4	Target NFV-MANO functional entity -> Impacted managed entity	Updates the state information it has about the state of the managed entities by discovering their actual state.
5	Target NFV-MANO functional entity -> Fallback Manager	Notifies the manager about the completion of the state reconciliation.
6	Fallback Manager	Unlocks the fallback target NFV-MANO functional entities.
7	Fallback Manager -> Tester	The fallback manager invokes the tester with indication of appropriate resources it can use for execution. The tester runs all the test suites applicable for the verification of the health and the performance of the fallback target NFV-MANO functional entities and the system as a whole.
8	Tester -> Fallback Manager	The tester determines that all NFV-MANO functional entities are running as expected and reports it back to the fallback manager.
Ends when	Fallback Manager	The fallback manager reports back that the fallback target NFV-MANO functional entities have been set back (restored) to the state provided by the restoration point adjusted to the state of the running entities and that the NFV system is working correctly.
NOTE: In case the fallback procedure cannot be completed successfully from a given restoration point, there is the possibility to try another fallback using a different (potentially older/wider scope) restoration point. If such a restoration point is not available, then the NFV-MANO system needs to be reinstalled.		

6 Recommendations

6.1 Introduction

General requirements for the software modification of NFV-MANO functional entities were identified in clause 5.6.2 in ETSI GS NFV-IFA 010 [i.7]. These requirements include:

1) The guarantee that running NSs/VNFs are not impacted by the software modification which will also leave the NFV-MANO functional entities reconnected with each other and towards external entities in the same way as prior to the software modification.

2) The possibility to process ongoing and incoming NS/VNF LCM and virtualised resource operations during the software modification - this processing may take different forms such as execution, delay, notification (for a subsequent retry).

In the light of the use cases developed in the present document, clause 6 presents a list of recommendations targeting further work to provide detailed requirements for the software modification of NFV-MANO functional entities.

NOTE: Clause 5.6 has introduced special considerations to be taken into account if containerized architectures are present:

- For all architectures with the CISM exposed to the NFV-MANO (clauses 5.6.2.2 to 5.6.2.6 and 5.6.3.2 to 5.6.3.6), all the recommendations related to NFV-MANO functional entities established in the following clauses are also applicable to the CISM.
- For architectures with CISM embedded into the VNF (clauses 5.6.2.7 and 5.6.3.7), there is no impact on NFV-MANO

6.2 General recommendations for NFV-MANO functional entities

Table 6.2-1 provides general recommendations related to the NFV-MANO functional entities.

Table 6.2-1: General recommendations related to the NFV-MANO functional entities

Identifier	Recommendation description	Use case reference
Gen.001	It is recommended that a requirement be specified to provide a functionality that is able to manage a fallback (see note 1).	Clause 5.9
Gen.002	It is recommended that a requirement be specified to provide a capability to create a restoration point containing all data that is necessary to restore an NFV-MANO functional entity. A restoration point may contain the data to restore more than one NFV-MANO functional entity (see notes 2 and 3).	Clauses 5.2 to 5.8
Gen.003	It is recommended that a requirement be specified that an NFV-MANO functional entity supports the capability of being restored using a specific restoration point to continue normal operation.	Clause 5.9
Gen.004	It is recommended that a requirement be specified that a restoration point indicates which NFV-MANO functional entities are contained in its data.	Clause 5.9
Gen.005	It is recommended that a requirement be specified to create an operation that allows to forcibly stop an NFV-MANO functional entity.	Clause 5.9
NOTE 1: The fallback manager could be implemented as part of the SMM. NOTE 2: Specification of the granularity and detailed content of restoration points is part of the stage 2 work. NOTE 3: Creating a restoration point includes any techniques for capturing the software and the actual state of NFV-MANO functional entities. This can also be a snapshot.		

6.3 Recommendations of functional requirements for NFV-MANO functional entities

Table 6.3-1 provides recommendations related to the functional requirements for NFV-MANO functional entities.

Table 6.3-1: Recommendations related to the functional requirements for NFV-MANO functional entities

Identifier	Recommendation description	Use case reference
Fun.001	It is recommended that an NFV-MANO functional entity provides internal support for software modification. This support includes operations such as starting an upgrade/downgrade.	Clause 5.2
Fun.002	It is recommended that an NFV-MANO functional entity with internal support handles any failure occurring during the execution of its software modification process (see note 1).	Clause 5.2
Fun.003	It is recommended that an NFV-MANO functional entity allows for any incoming NS LCM operation, VNF LCM operation and/or VRM operation to be executed in parallel to the software modification of the NFV-MANO functional entity (see note 2).	Clauses 5.2.2.8 and 5.3.3.8
Fun.004	It is recommended that an NFV-MANO functional entity with internal support for software modification supports service switchover procedure with entities towards whom the NFV-MANO service interface(s) change(s) incompatibly during the execution of the software modification.	Clause 5.4
Fun.005	It is recommended that a requirement be specified that an NFV-MANO functional entity with internal support for software modification provides at least the same service availability guarantees during software modification as for the case of failures during normal operation.	Clause 5.2
Fun.006	It is recommended that a requirement be specified that an NFV-MANO functional entity with internal support for software modification, when detecting that the software modification cannot be completed as requested, either internally executes a software rollback of the software modification and reports this in the result, or reports a failure of the software modification, or suspends the modification and reports this result (see notes 3 and 10).	Clauses 5.2 and 5.7.1
Fun.007	It is recommended that a requirement be specified, in the case of an NFV-MANO functional entity with internal support for software modification, that as part of the software rollback process the affected NFV-MANO functional entity is capable of synchronizing the state with other impacted peer NFV-MANO functional entities in order to reach a consistent state, if synchronization was also part of the software modification.	Clause 5.2

Identifier	Recommendation description	Use case reference
Fun.008	It is recommended that a requirement be specified that an NFV-MANO functional entity with internal support for software modification is capable of the suspension of an ongoing software modification process (see note 4).	Clause 5.7.1
Fun.009	It is recommended that a requirement be specified that an NFV-MANO functional entity with internal support for software modification is capable of resuming the software modification after suspending such a modification (see note 10).	Clause 5.7.1
Fun.010	It is recommended that a requirement be specified that an NFV-MANO functional entity with internal support for software modification is capable of performing the software modification in steps.	Clause 5.7.1
Fun.011	It is recommended that a requirement be specified that an NFV-MANO functional entity with internal support for software modification is capable of undoing a software modification after suspending such a modification (see note 10).	Clause 5.7.1
Fun.012	It is recommended that a requirement be specified that an NFV-MANO functional entity with internal support for software modification is capable of internally committing the software modification process.	Clauses 5.2 to 5.8
Fun.013	It is recommended that a requirement be specified that an NFV-MANO functional entity with internal support for software modification is capable of reporting the status of an ongoing software modification process.	Clause 5.7.1
Fun.014	It is recommended that a requirement be specified that the execution of the software modification of an NFV-MANO functional entity does not interrupt any on-going NFV-MANO service (see note 5).	Clauses 5.2.2.7 and 5.3.3.7
Fun.015	It is recommended that a requirement be specified that an NFV-MANO functional entity without internal support for software modification supports the option of being deployed as a set of redundancy units so that such organization can be exploited during software modification to maintain service availability (see note 6).	Clause 5.3.2
Fun.016	It is recommended that a requirement be specified that an NFV-MANO functional entity without internal support for software modification supports lifecycle management capabilities for individual redundancy units and for the single NFV-MANO functional entity they compose as a set (see notes 6 and 7).	Clause 5.3.2
Fun.017	It is recommended that a requirement be specified that an NFV-MANO functional entity is instantiated with its application in the STARTED_LOCKED state.	Clauses 5.3 to 5.9
Fun.018	It is recommended that a requirement be specified that an NFV-MANO functional entity supports its testing while its NFV-MANO functional entity application is in the STARTED_LOCKED state. The execution of the test can be triggered upon request.	Clauses 5.3 to 5.9
Fun.019	It is recommended that a requirement be specified that an NFV-MANO functional entity without internal support for software modification supports state management capabilities for individual NFV-MANO service interfaces (see note 8).	Clauses 5.3 to 5.9
Fun.020	It is recommended that a requirement be specified that an NFV-MANO functional entity is able to discover which entities it is supposed to manage (see note 9).	Clause 5.9
Fun.021	It is recommended that a requirement be specified that an NFV-MANO functional entity is able to query managed entities about state information that the NFV-MANO functional entity needs in order to provide its service (see note 9).	Clause 5.9

NOTE 1: If the conclusion is that the software modification cannot be completed successfully, it is recommended that, at least, the changes already applied are undone and the process completes when the NFV-MANO functional entity returns to the configuration it was before the initiation of the software modification.

NOTE 2: Requirements in clause 5.6.2 of ETSI GS NFV-IFA 010 [i.7] mandate the ability to process and handle an incoming NS LCM operation, VNF LCM operation, or virtualised resource management operation during the software modification. However, according to these requirements, "handling" the request also allows to postpone or reject the operation to after the completion of the software modification, while recommendation 003 states that executing the operation in parallel to the software modification should be possible as it is the preferred action to handle the incoming operation request.

NOTE 3: The NFV-MANO functional entity with internal support for software modification can internally retry some steps of the software modification, and these retries are recommended to be reported externally, for example, to SMM. The externally distinguishable cases according to the returned result are at least: the software modification was executed successfully, the software modification was rolled back successfully, or the software modification has failed.

Identifier	Recommendation description	Use case reference
NOTE 4:	Clause 5.6.1 in ETSI GS NFV-IFA 010 [i.7], mandates that the entire software modification task including preparation work and closing actions be completed within a maximum maintenance period or, in case of a long-running maintenance, be completed as a set of separate phases, whereby each phase (including its preparation and closing work) can be completed within the specified maximum maintenance period. It is further clarified that the functionality to support splitting the entire software modification into multiple phases could be realized by allowing to "pause" (i.e. suspend and resume) the software modification process at certain provider defined steps of the modification process.	
NOTE 5:	See also clause 5.6.2 in ETSI GS NFV-IFA 010 [i.7].	
NOTE 6:	When multiple redundancy units of an NFV-MANO functional entity are deployed as a set, they compose a single instance of the NFV-MANO functional entity and, accordingly, act together to deliver a better availability of the services provided by the NFV-MANO functional entity.	
NOTE 7:	The lifecycle management capabilities include instantiation, graceful and forceful termination.	
NOTE 8:	The state management capabilities include starting, graceful and forceful stopping of individual NFV-MANO service interfaces.	
NOTE 9:	See also clause 9.3 in ETSI GS NFV-IFA 010 [i.7].	
NOTE 10:	After encountering an external issue or failure, the NFV-MANO functional entity with internal support for software modification can suspend the software modification at a state in which suspending the operation is safe.	

6.4 Recommendations for interfaces of NFV-MANO functional entities

Table 6.4-1 provides recommendations related to the interfaces of NFV-MANO functional entities.

Table 6.4-1: Recommendations related to the interfaces of NFV-MANO functional entities

Identifier	Recommendation description	Use case reference
If.001	It is recommended that an NFV-MANO functional entity with internal support for software modification provides synchronization points with those entities towards whom the NFV-MANO service interface(s) change(s) during the execution of the software modification to indicate when switching over to the new NFV-MANO service interface(s) can be started and to indicate when the switching has completed - the old NFV-MANO service interface(s) is/are not in use any more.	Clause 5.4
If.002	It is recommended that a requirement be specified to support the capability to notify the progress of the software modification, including start and end of the overall software modification, as well as the end of each major step of the software modification.	Clause 5.7
If.003	It is recommended that a requirement be specified to support the capability to query the status of the software modification, including the estimated completion percentage, the estimated time to complete and the current step/phase of the software modification.	Clause 5.7
If.004	It is recommended that a requirement be specified that the result of a software modification operation returned by an NFV-MANO functional entity with internal support for software modification provides enough information based on which an external management entity, such as SMM, can decide whether retrying the operation is possible and feasible (see note 4).	Clauses 5.2, 5.8 and 5.9
If.005	It is recommended that a requirement be specified that an NFV-MANO functional entity provides an option to be deployed as a set of redundancy units so that such organization can be exploited by the SMM to eliminate service impact (see note 3).	Clause 5.3.2
If.006	It is recommended that a requirement be specified that an NFV-MANO functional entity provides NFV-MANO management lifecycle management capabilities for the NFV-MANO functional entity as a whole and for its redundancy units individually (see notes 1 and 3).	Clauses 5.3 and 5.4
If.007	It is recommended that a requirement be specified that an NFV-MANO functional entity provides state management capabilities for the NFV-MANO functional entity as a whole and for its redundancy units individually (see notes 2 and 3).	Clauses 5.3, 5.4 and 5.9

Identifier	Recommendation description	Use case reference
If.008	It is recommended that a requirement be specified that an NFV-MANO functional entity exposes the state of individual NFV-MANO service interfaces for the NFV-MANO functional entity, its individual redundancy units, and the state of the NFV-MANO functional entity application (see note 3).	Clauses 5.3, 5.4 and 5.9
If.009	It is recommended that a requirement be specified that an NFV-MANO functional entity with internal support for software modification provides the capability to process requests for suspending, resuming and undoing an ongoing software modification process (see notes 5 and 7).	Clause 5.7
If.010	It is recommended that a requirement be specified to expose the capability to enable an NFV-MANO functional entity to discover which entities it is supposed to manage (see note 6).	Clause 5.9
If.011	It is recommended that a requirement be specified to create an interface to enable an NFV-MANO functional entity to query from a 'managed entity' the information needed for managing this 'managed entity' (see note 6).	Clause 5.9

NOTE 1: The NFV-MANO management lifecycle management capabilities include instantiation, graceful and forceful termination.
NOTE 2: The state management capabilities include starting, graceful and forceful stopping of individual NFV-MANO service interfaces.
NOTE 3: When multiple redundancy units of an NFV-MANO functional entity are deployed, they compose a single instance of the NFV-MANO functional entity and, accordingly, act together to deliver better availability of the services provided by the NFV-MANO functional entity.
NOTE 4: The returned result needs to distinguish at least whether the software modification was executed successfully, has failed, was rolled back successfully and a retry is possible, or was rolled back successfully but a retry is not feasible.
NOTE 5: Clause 5.6.1 of ETSI GS NFV-IFA 010 [i.7], mandates that the entire software modification task, including preparation work and closing actions, is completed within a maximum maintenance period or, in case of a long-running maintenance, be completed as a set of separate phases, whereby each phase (including its preparation and closing work) can be completed within the specified maximum maintenance period. It is further clarified that the functionality to support splitting the entire software modification into multiple phases could be realized by allowing to "pause" (i.e. suspend and resume) the software modification process at certain provider defined steps of the modification process.
NOTE 6: See also clause 9.3 in ETSI GS NFV-IFA 010 [i.7].
NOTE 7: Upon encountering an external issue or failure, by resuming, the NFV-MANO functional entity with internal support for software modification will continue the execution of the software modification; and by undoing, the NFV-MANO functional entity with internal support will undo the completed part of the software modification.

6.5 Recommendations for information associated with the software modification of NFV-MANO functional entities

Table 6.5-1 provides recommendations related to information associated with the software modification of NFV-MANO functional entities.

Table 6.5-1: Recommendations related to information associated with the software modification of NFV-MANO functional entities

Identifier	Recommendation description	Use case reference
Inf.001	It is recommended that information is provided with an NFV-MANO functional entity to support SMM's decision making about retries and software rollback (see notes 1 and 3).	Clauses 5.2 to 5.9
Inf.002	It is recommended that the software modification execution plan allows to specify operations for SMM in an abstract form which does not require SMM full awareness of the operations' semantics (e.g. SMM would need to understand the NFV-MANO functional entity's state, but not the interface operations towards managed entities).	Clauses 5.2 to 5.9
Inf.003	It is recommended that a requirement be specified that information is provided with an NFV-MANO functional entity with internal support for software modification on the scope and constraints of the software modifications it is capable of performing (see note 3).	Clause 5.2

Identifier	Recommendation description	Use case reference
Inf.004	It is recommended that a requirement be specified that information is provided with an NFV-MANO functional entity with internal support for software modification on the additional resources needed to execute the software modification. These resources can also be used to enable recovery procedures from a failure during the software modification such as software rollback (see note 3).	Clause 5.2
Inf.005	It is recommended that a requirement be specified that information is provided with an NFV-MANO functional entity on the compatibility information of all supported NFV-MANO service interfaces, e.g. supported versions.	Clauses 5.2 to 5.4
Inf.006	It is recommended that an artefact and description template is specified to be used for execution plans for SMM to orchestrate the software modification processes of NFV-MANO functional entities.	Clauses 5.2 to 5.9
Inf.007	It is recommended that a requirement be specified that information is provided with an NFV-MANO functional entity for SMM to know what information to be collected for a service handover on the different service interfaces with respect to the peering entity types and the information associated with the service interface (see note 2).	Clauses 5.3 to 5.4
Inf.008	It is recommended that a requirement be specified that information is provided with an NFV-MANO functional entity with internal support for software modification on whether it is capable of handling the software modification during an ongoing - or in parallel with an - NS LCM operation, VNF LCM operation and/or VRM operation.	Clauses 5.2.2.7, 5.2.2.8, 5.3.3.7 and 5.3.3.8
Inf.009	It is recommended that a requirement be specified that information is provided by the operator on the conditions under which it is permitted to perform the software modification process during an ongoing - or in parallel with an - NS LCM operation, VNF LCM operation and/or VRM operation.	Clauses 5.2.2.7, 5.2.2.8, 5.3.3.7 and 5.3.3.8

NOTE 1: NFV-MANO functional entities with internal support for software modification return the result of a software modification as described in note 4 of table 6.4-1, based on which SMM can decide whether it needs to retry the operation or trigger the software rollback of the overall procedure. When an NFV-MANO functional entity provides no internal support for software modification, the results of different operations are not specific for the context of software modification. Therefore, additional hints are desirable on the handling of the operations results in the context of a software modification. For example, if instantiation or starting a service interface fails during a software modification, under what circumstances retry is feasible.

NOTE 2: During a software modification process, SMM can provide the target NFV-MANO functional entity and its peering entities with information necessary for proper peering to transfer the setup from the source to the destination configuration. For this purpose, SMM needs to be able to identify the entities from which it needs to collect the information and the information that needs to be collected. That is, the SMM needs to know the entity types that need such information for a given interface. A given entity can connect to different interfaces of the target NFV-MANO functional entity in different roles as different types. In some cases, no information needs to be collected.

NOTE 3: A way to deliver such information could be a capabilities descriptor of the NFV-MANO functional entity, or some documentation/files within a package.

6.6 Recommendations for the SMM

All use cases studied in the present document make an assumption about the existence of a functionality managing the software modification process of NFV-MANO functional entities and refer to the entity fulfilling this assumption as the SMM - the Software Modification Manager.

The SMM is responsible to coordinate the software modification process at the system level, coordinating potentially multiple different entities in the process. Thus, the SMM is not part of any internal support of software modification of an NFV-MANO functional entity, whose scope is limited to the NFV-MANO functional entity itself. Rather the SMM, when receiving a software modification request, can make use of such internal support for software modification of individual NFV-MANO functional entities to fulfil the request.

There is no assumption made where the SMM resides and which existing or new functional entity provides this functionality (e.g. OSS, NFV-MANO, etc.), or whether this role is performed by an operator.

Accordingly, the recommendations related to the SMM provided in this clause capture the results of the study in the context of this assumption. Depending on the decision about the placement of this functionality, some of the recommendations may not be applicable, or may need to be adjusted.

Table 6.6-1 provides recommendations related to the SMM.

Table 6.6-1: Recommendations related to the SMM

Identifier	Recommendation description	Use case reference
Smm.001	It is recommended that the SMM is capable of using the information collected about and during a software modification of NFV-MANO functional entities for orchestrating a software rollback procedure if required or requested.	Clauses 5.2 to 5.8
Smm.002	It is recommended that the general requirements specified in clause 5.6.2 of ETSI GS NFV-IFA 010 [i.7] are extended to clarify that the requirements apply for any deployment, i.e. independent of whether an SMM is deployed or not.	Clauses 5.2 to 5.9
Smm.003	It is recommended that a requirement be specified that the SMM provides the capability to support the software modification of NFV-MANO functional entities.	Clauses 5.2 to 5.9
Smm.004	It is recommended that a requirement be specified that the SMM is capable of orchestrating the software modification processes of NFV-MANO functional entities, including any necessary retries and software rollback.	Clauses 5.2 to 5.9
Smm.005	It is recommended that a requirement be specified that the SMM is capable of using a software modification execution plan to orchestrate the software modification processes of NFV-MANO functional entities.	Clauses 5.2 to 5.9
Smm.006	It is recommended that a requirement be specified that the SMM is capable of interacting with a Tester entity to test the target NFV-MANO functional entities, and the system under test as a whole.	Clauses 5.2 to 5.9
Smm.007	It is recommended that a requirement be specified that the SMM is capable of making a decision on whether the software modification process can proceed with the next step, whether a failed step can be retried, or whether a software rollback or a fallback needs to be triggered (see note 1).	Clauses 5.2 to 5.9
Smm.008	It is recommended that a requirement be specified that the SMM is capable of making a decision about the steps needed to be performed for the software rollback depending on the steps of the software modification executed up to the point at which the software rollback is triggered (see note 1).	Clauses 5.2 to 5.9
Smm.009	It is recommended that a requirement be specified that the SMM is capable of making a decision about the outcome of a software modification process and, if it is completed or rolled back successfully, committing it (see note 1).	Clauses 5.2 to 5.8
Smm.010	It is recommended that a requirement be specified that the SMM is capable of evaluating the amount of resources needed to be allocated, as well as reserving, allocating and releasing resources necessary for a software modification process, including resources enabling software rollback and testing.	Clauses 5.2 to 5.9
Smm.011	It is recommended that a requirement be specified that the SMM is capable of reporting the status of an ongoing software modification process. The SMM can request from collaborating entities (e.g. the tester) the status of the operations they are performing as part of the software modification process.	Clauses 5.2 to 5.9
Smm.012	It is recommended that a requirement be specified that the SMM is capable of detecting the state of individual NFV-MANO service interfaces of an NFV-MANO functional entity, its redundancy units, and of the NFV-MANO functional entity application.	Clauses 5.2, 5.8 and 5.9
Smm.013	It is recommended that a requirement be specified that the SMM is capable of collecting information about the NFV-MANO functional entities at the beginning and during a software modification process necessary for orchestrating a software rollback.	Clauses 5.2 to 5.8
Smm.014	It is recommended that a requirement be specified that the SMM is capable of consuming the interfaces exposed by the NFV-MANO functional entities for software modification management purposes (see note 2).	Clauses 5.2 to 5.9

Identifier	Recommendation description	Use case reference
Smm.015	It is recommended that a requirement be specified that the SMM is capable of generating notifications about status changes in an ongoing software modification process (see note 3).	Clause 5.7
Smm.016	It is recommended that a requirement be specified that the SMM is capable of suspending and resuming an ongoing software modification process. This implies the suspension at least at step boundaries of the software modification process. It can also imply the SMM requesting collaborating entities (e.g. the tester) to suspend/resume an operation they are carrying out as part of the software modification process (see note 4).	Clause 5.7
Smm.017	It is recommended that a requirement be specified that the SMM is capable of evaluating, based on the related capabilities of the NFV-MANO functional entity as well as the operator's configuration, whether the software modification process is permitted during an ongoing - or in parallel with an - NS LCM operation, VNF LCM operation and/or VRM operation.	Clauses 5.2.2.7, 5.2.2.8, 5.3.3.7 and 5.3.3.8

NOTE 1: The SMM decision making process depends on the operator policies, the encountered situation such as a failure; it can include waiting for feedback from the consumer that has triggered the software modification.
NOTE 2: The NFV-MANO functional entities could be the target NFV-MANO functional entities and also the impacted NFV-MANO functional entities.
NOTE 3: Status changes include at least the start and the end of the software modification, as well as the step boundaries of the software modification process.
NOTE 4: Clause 5.6.1 of ETSI GS NFV-IFA 010 [i.7], mandates the entire software modification task including preparation work and closing actions to be completed within a maximum maintenance period or, in case of a long-running maintenance, be completed as a set of separate phases, whereby each phase (including its preparation and closing work) can be completed within the specified maximum maintenance period. It is further clarified that the functionality to support splitting the entire software modification into multiple phases could be realized by allowing to "pause" (i.e. suspend and resume) the software modification process at certain provider defined steps of the modification process.

7 Conclusion

The present document studies the scenarios of NFV-MANO software modification. The overall software modification process and related concepts are introduced in clause 4.1.2, while clause 4.1.3 elaborates further on the recovery procedures, namely, fallback, rollback and retry.

Different use cases of NFV-MANO software modification are then developed:

- NFV-MANO functional entities with seamless support for their software modification (clause 5.2.2);
- NFV-MANO functional entities with non-seamless support for their software modification (clause 5.2.3);
- NFV-MANO functional entities with redundant components (clause 5.3.2);
- NFV-MANO functional entities not exposing their structure (clause 5.3.3);
- simultaneous software modification of two NFV-MANO functional entities (clause 5.4.2); and
- replacement of a S-VNFM with a G-VNFM (clause 5.5.2), as an example of the NFV-MANO architectural options sketched in clause 4.2.2.

An analysis of the following use cases is also realized:

- considerations for containerized architectures (clause 5.6);
- status monitoring and suspend operation (clause 5.7); and
- retry (clause 5.8) and fallback (clause 5.9) procedures.

Two categories of recommendations are finally derived from this study: recommendations calling for requirements specification (i.e. "It is recommended that a requirement be specified"), and recommendations indicating preferences. These two categories of recommendations cover five areas related to NFV-MANO software modification:

- general requirements for NFV-MANO functional entities (clause 6.2);
- functional requirements for NFV-MANO functional entities (clause 6.3);
- interfaces requirements for NFV-MANO functional entities (clause 6.4);
- information associated with the software modification of NFV-MANO functional entities (clause 6.5); and
- Software Modification Manager (SMM) (clause 6.6).

Annex A:
Utilization of containers in NFV

A.1 Introduction

This annex provides a summary of clauses 5.3 and 7.2.4 of ETSI GR NFV-IFA 029 [i.5] related to containers. More details can be found in that document.

A.2 Examples of the utilization of containers in NFV

The first of the five example use cases is "Container-based NFV micro-services within the VNF" (clause 5.3.2 of ETSI GR NFV-IFA 029 [i.5]). In this example, each VNFC is implemented as OS containers provided within a VM (figure A.2-1 based on ETSI GR NFV-IFA 029 [i.5]). The Container Infrastructure Service Management (CISM) functionality, also hosted in a VM, manages the lifecycle of containers provided by the Container Infrastructure Service (CIS) functionality, and initiates container-specific virtual resource management based on the VMs allocated to the VNF. The VNFM manages the VNF/VNFCs and the VMs are allocated to the VNF as usual. As the functionalities of CIS and CISM are invisible to NFV-MANO functional entities, no additional requirement is needed from NFV-MANO in this example.

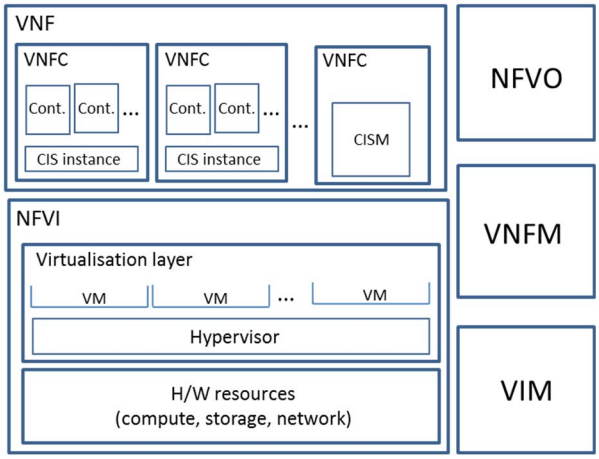

Figure A.2-1: Example of the utilization of containers in NFV

Based on ETSI GR NFV-IFA 029 [i.5], figure A.2-2 shows the four other example use cases of containers usage in NFV in which the CISM functionality is to be implemented differently, i.e. not within a VNF.

For the use case "VNFC in containers on bare metal" (figure A.2-2 a)), each VNFC of the VNF is implemented as a single OS container (i.e. in a 1:1 relation) and all containers run on bare metal (clause 5.3.3 of ETSI GR NFV-IFA 029 [i.5]). The NFVI provides the virtualisation containers, the NFV-MANO provides the management for the VNFCs (e.g. resource allocation), while the CISM function provides and provisions the containers in which the VNFCs are deployed.

For the use case "VNFC in containers in VM" (figure A.2-2 b)), each VNFC of the VNF is also implemented as a single OS container running as nested virtualisation in a VM (clause 5.3.4 of ETSI GR NFV-IFA 029 [i.5]). The NFVI provides VMs in which there are virtualisation containers that are OS containers. The NFV-MANO provides the management, e.g. LCM, for the VNFCs, with a CISM function providing and provisioning the containers.

For the use case "VNFC in a group of containers", each VNFC of the VNF is implemented as a group of OS containers that forms a managed container infrastructure object (clause 5.3.5 of ETSI GR NFV-IFA 029 [i.5]). The OS containers of such a group share the resources and run-time environment. The group of OS containers can run in nested virtualisation (figure A.2-2 c)) but can also run on bare metal (not shown).

Finally, a combined approach is possible with the use case "Containers on bare metal and VMs provided by NFVI" (figure A.2-2 d)) with the NFVI providing both OS containers on bare metal and VMs (clause 5.3.6 of ETSI GR NFV-IFA 029 [i.5]). The VNFs are deployed on CIS instances: some CIS instances run on VMs, and others on bare metal. A variety of options are possible for container deployment, e.g. a VNF can be hosted by one or more CIS instances.

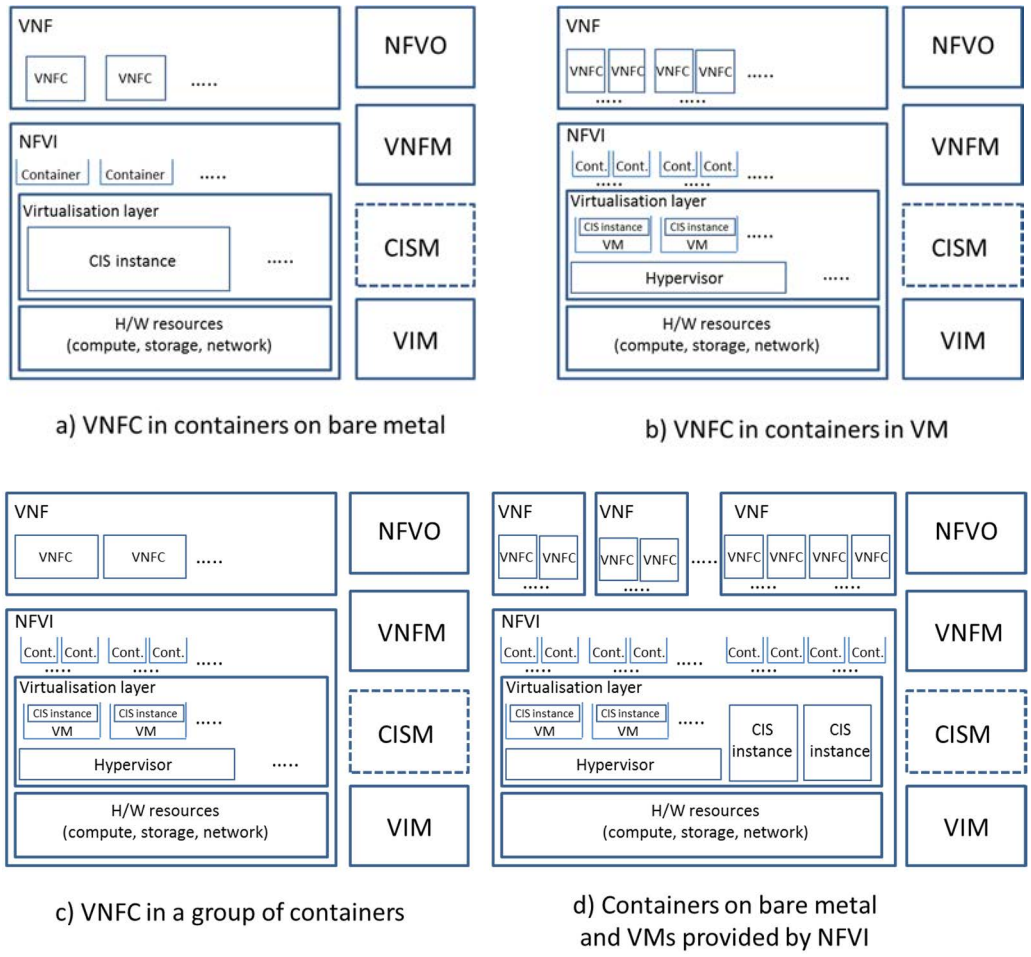

Figure A.2-2: Other examples of the utilization of containers in NFV

In the four previous examples, a VNF is composed of containerized VNFCs whose functions are encapsulated in container images and instantiated by the CISM function. From its southbound interfaces perspective, the CISM function manages the CIS function according to the request of the CISM function northbound consumers (e.g. VNFM). The CIS function provides a container runtime environment such as Docker, which executes the containerized workload (e.g. VNF, VNFC) encapsulated in the OS container and managed by the CISM function northbound consumer. It is noteworthy that the CISM function belongs to the VIM's northbound consumers. As for the CISM function northbound interfaces, the VNFM consumes the lifecycle management of containerized workload running in OS containers provided by the CISM function, while the NFVO consumes the management of service resources for the containerized workload provided by the CISM function in indirect mode.

The introduction of the container technology induces some changes from the current NFV-MANO. The NFVO is enhanced with the capability:

- To access and consume interfaces exposed by a container image registry.
- To initiate the management of service resources for the containerized workload towards the CISM function (indirect mode) or grant service resources for the containerized workload requested by the VNFM (direct mode).

The VNFM is enhanced with the capability to initiate:

- The operations towards the CISM function related to the managed container infrastructure object package management.
- The management of service resources for the containerized workload towards the NFVO in indirect mode (or its granting equivalence in direct mode).

There is no impact foreseen for the VIM which provides to its consumer the NFVI virtualised resources (compute, storage, network) within the operator's NFVI.

A.3 Integration of the CISM functionality in the NFV framework

Based on clauses 7.2.4.2 and 7.2.4.3 of ETSI GR NFV-IFA 029 [i.5], the figure A.3-1 shows how the CISM functionality can be integrated in the current NFV-MANO architecture by introducing it into the VIM (figure A.3-1 a)), or by placing the MCIO management function in the VNFM, and the container-specific management function in the VIM (figure A.3-1 b)).

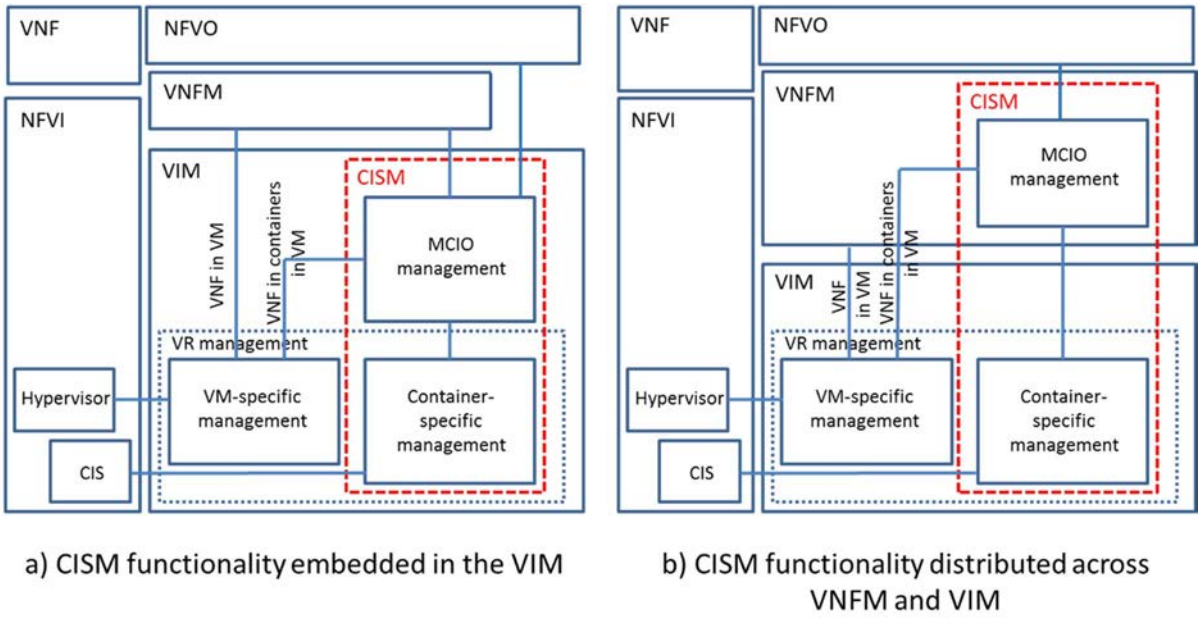

Figure A.3-1: CISM functionality integrated in the current NFV-MANO architecture

Based on clauses 7.2.4.4 and 7.2.4.5 of ETSI GR NFV-IFA 029 [i.5], the figure A.3-2 shows the CISM functionality implemented as a distinct functional entity, i.e. as a standalone functional entity (figure A.3-2 a)), or replacing both VIM and VNFM (figure A.3-2 b)).

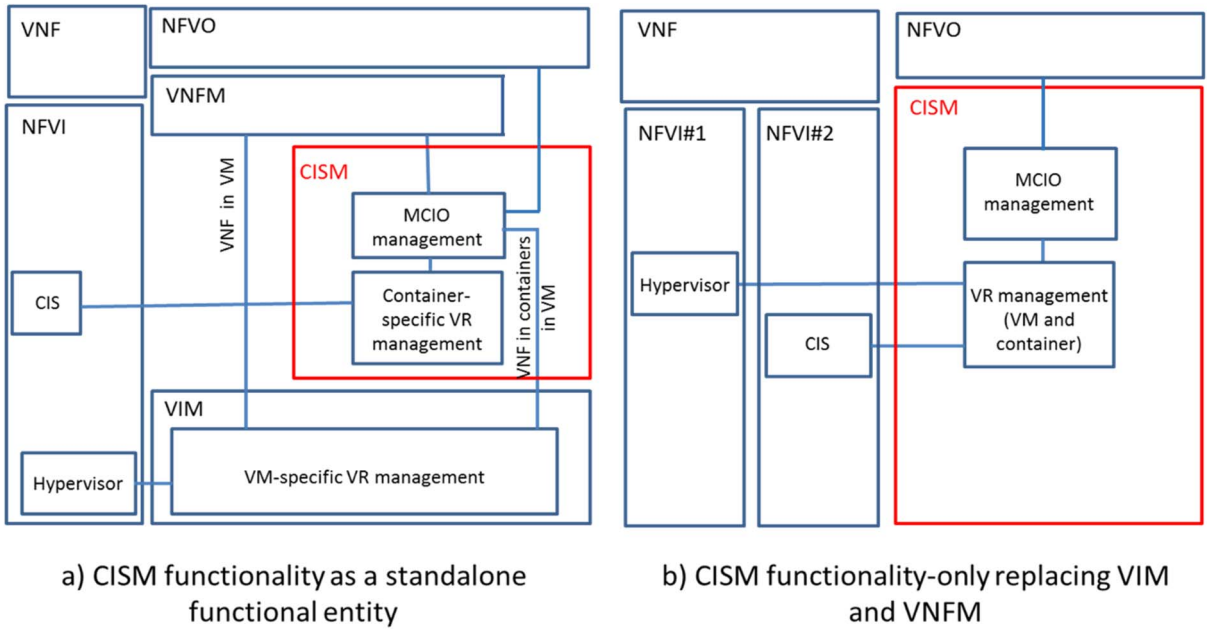

Figure A.3-2 : CISM functionality as a distinct functional entity

Based on clauses 7.2.4.6 and 7.2.4.7 of ETSI GR NFV-IFA 029 [i.5], the figure A.3-3 shows the CISM functionality implemented in a VNF with support for shared container service (figure A.3-3 a)), or without support for shared container service (figure A.3-3 b)). In the first case, all container-based VNFs benefit from the shared CISM functions, while in the second case, each container-based VNF needs to host its own CISM functions. Accordingly, the CISM functionality is exposed to the NFV-MANO in the first case, but it is transparent in the second case.

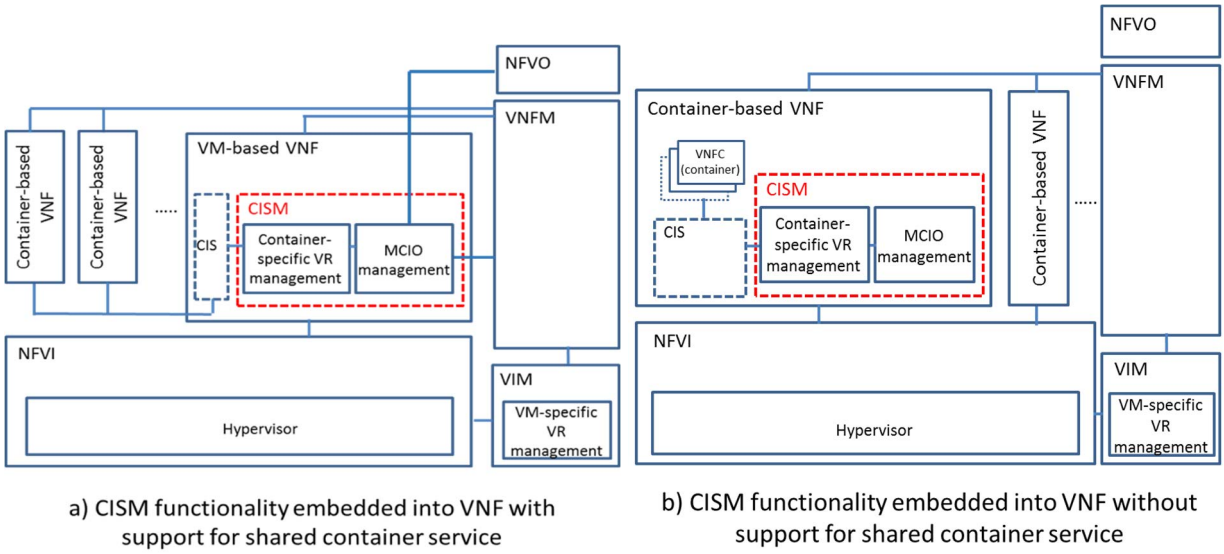

Figure A.3-3: CISM functionality in a VNF

History

Document history		
V1.1.1	November 2020	Publication
V4.1.1	November 2020	Publication