ETSI GS NFV-TST 010 V2.6.1 (2020-09)

Network Functions Virtualisation (NFV) Release 2; Testing; API Conformance Testing Specification

Disclaimer

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

Reference
RGS/NFV-TST010ed261

Keywords
API, conformance, NFV, testing

ETSI

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

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

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

Foreword

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

Modal verbs terminology

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

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

Introduction

The interoperability among the functional entities which supports the requirements and functionalities specified in ETSI NFV deliverables is one of the key and most important aspects to develop and deploy the NFV environment. In order to achieve such interoperability, ETSI GR NFV-TST 007 [i.1] specifies the methodologies and test scenario for the testing of interoperability mainly based on the ETSI NFV IFA specification series which specifies the reference point/interface and functional requirements. At the same time, the validation of the NFV specification compliance of each functional entities is also key aspects to be ensured for the interoperability, in particular protocol solution/API level. Therefore the present document specifies the API conformance testing.

1 Scope

Scope of API conformance is the functionality test in an automated way for ETSI NFV APIs.

The goal of the present document is to specify the methodologies of conformance test including Test Descriptions for NFV implementations with interfaces specified in the following NFV specifications: ETSI GS NFV-SOL 002 [2] for the *Ve-Vnfm* reference point, ETSI GS NFV-SOL 003 [1] for the *Or-Vnfm* reference point and ETSI GS NFV-SOL 005 [3] for the *Os-ma-nfvo* reference point.

Each ETSI NFV SOL deliverable specifies a set of interfaces built on the RESTful approach and meant to be used over the HTTP protocol. The aim of the present document is to define the methodologies and the procedures with Test Descriptions to test conformance of the exchanged HTTP payloads and the implementation of required actions for one or more of the available interfaces within a reference point.

Since the targets of the testing are functionality (semantic checks) and the HTTP payloads (syntax checks), methodologies, including test suite(s) and/or any technologies from any organizations (in particular Open Source Initiatives) that can improve or help the (automated) test execution are also considered as being in the scope of the present document.

2 References

2.1 Normative references

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

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

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

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

[1] ETSI GS NFV-SOL 003 (V2.6.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point".

[2] ETSI GS NFV-SOL 002 (V2.6.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Ve-Vnfm Reference Point".

[3] ETSI GS NFV-SOL 005 (V2.6.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point".

[4] ETSI GS NFV-TST 002 (V1.1.1): "Network Functions Virtualisation (NFV); Testing Methodology; Report on NFV Interoperability Testing Methodology".

[5] ETSI GS NFV-SOL 013 (V2.6.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; Specification of common aspects for RESTful NFV MANO APIs".

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 GR NFV-TST 007 (V2.5.1): "Network Functions Virtualisation (NFV) Release 2; Testing; Guidelines on Interoperability Testing for MANO".

[i.2] Robot Framework.

NOTE: Available at http://robotframework.org.

[i.3] Robot2doc tool.

NOTE: Available at https://forge.etsi.org/rep/forge-tools/robot2doc.

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

Test Description (TD): set of information required to define/run the API conformance test and to realize the verdict for the API conformance test

3.2 Symbols

Void.

3.3 Abbreviations

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

API	Application Programming Interface
AUT	API Under Test
EM	Element Manager
FUT	Function Under Test
IUT	Implementation Under Test
LCM	Life Cycle Management
LCOG	Lifecycle Operation Granting
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
NS	Network Service
NSD	Network Service Destriptor
OCC	OCCurrence
PM	Performance Management
PNF	Physical Network Function
PNFD	PNF Descriptor
SUT	System Under Test
TD	Test Description
TLS	Transport Layer Security

VE	Virtual Element
VIM	Virtualised Infrastructure Manager
VNF	Virtual Network Function
VNFD	VNF Descriptor
VNFM	VNF Manager
VRQAN	Virtualised Resources Quota Available Notification

4 Methodology

4.1 General

The purpose of general conformance testing is to determine to what extent a single implementation of a particular standard conforms to the individual requirements of that standard. Concepts from ETSI GS NFV-TST 002 [4] are used in the present document.

The important factors which characterize conformance testing are as follows:

- the System or Implementation Under Test (SUT or IUT) defines the boundaries (open interfaces) for testing;
- the conformance test system is a specialized tool (system) built for the purpose of testing and on which specific test scripts can be run;
- the SUT comes from a single supplier (or, at least, a single product line);
- the tests are executed by a dedicated test system that has full control of the SUT and the ability to observe all communications from the SUT;
- the tests are performed at open standardized interfaces that are not (usually) accessible to a normal user (i.e. they are specified at the protocol level);
- the tests are specified at the detailed protocol level and are not usually based on functionality as experienced by a user;
- the tests verify response or related request operation from SUT.

4.2 System Under Test (SUT)

The system under test is identified by an implementation of the function under test producing or consuming the API under test e.g. in the case of the Or-vnfm reference point the function under test may be either a NFVO implementation or a VNFM implementation.

The function shall be tested in isolation with respect to other functional blocks in a NFV platform, to guarantee that the outcomes of the conformance tests are not result of interoperability issues with other components.

4.3 Test configurations

4.3.1 General

In accordance with clause 1, the scope of the present document is to define a testing methodology and test suite for both the conformant protocol exchange (i.e. valid serialization and order of messages) and the initialization or execution of the functionalities mandated for each protocol operation, including the conformant management of internal state.

In order to enable the FUT to correctly execute the operations mandated the FUT shall be tested while being executed in a test environment (TSTENV) which provides all the functional elements needed for the correct outcome of the operation.

NOTE: For example, to correctly execute an instantiation a VNFM requires evaluation in a test environment which provides a VIM and NFVI plus the NFVO to grant the operation.

The test system shall provide the implementation of an API Consumer and a Notification Endpoint for the API Under Test (AUT). Moreover, the test configuration may contain observation interfaces between the Test System and the FUT or any other functional block which is part of the test environment. The specification of the mentioned observation interfaces is out of the scope of the present document.

Stimuli to the FUT shall be injected by the Test System via the AUT only.

Conformance checks on the status and outcome of the operations triggered by the protocol shall be verified by the Test System by means of:

- read operations issued via the AUT; or
- reception of notifications on the Notification Endpoint exposed by the test system; or
- other test interfaces to support triggers or verifications.

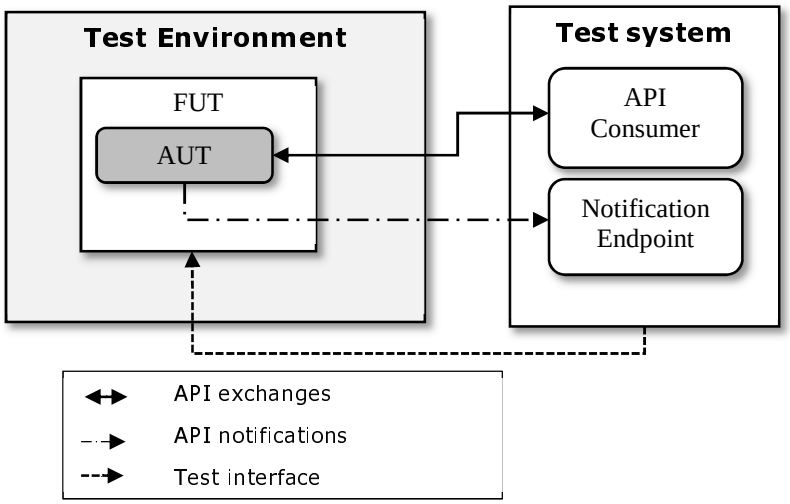

Figure 4.3.1-1: Generic SUT configuration

The test configurations specified in clause 4.3 fulfil the needs of Test Descriptions specified in annexes D, E and F contained in archive gs_nfv-tst010v020601p0.zip which accompanies the present document for the different FUTs and AUTs in scope of the present document.

4.3.2 Config_prod_VE

The configuration config_prod_VE shall be implemented to test APIs which are produced by FUTs in a VNF or EM. The test environment of the VNF/EM is the NFVI where the test is executed.

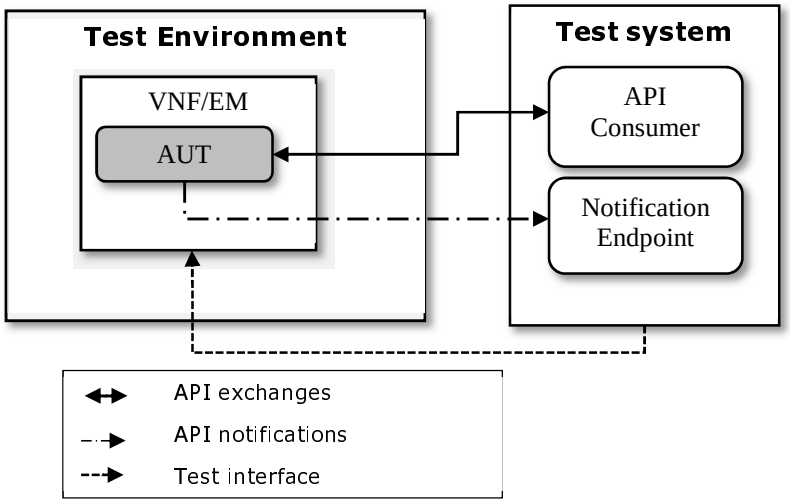

Figure 4.3.2-1: Configuration for tests of APIs with the FUTs as Producer run in a VNF/EM

4.3.3 Config_prod_VNFM

The configuration config_prod_VNFM shall be implemented to test APIs produced by FUTs which implement a VNFM. The test environment of the virtual element is the NFVI where the VE is executed.

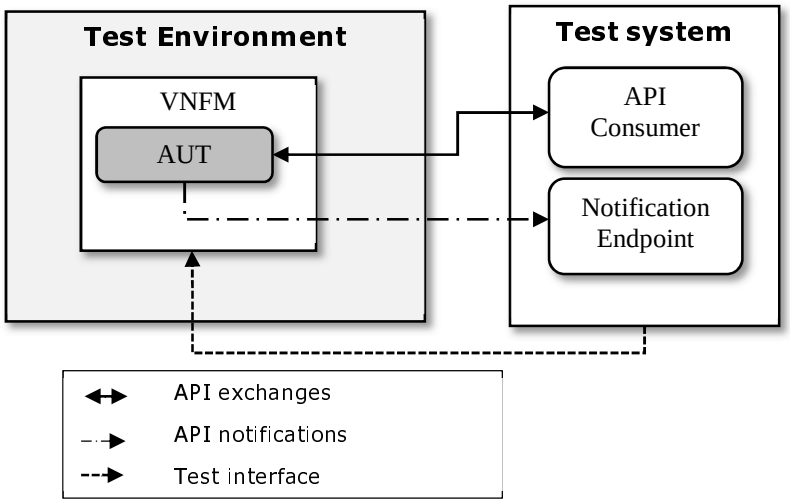

Figure 4.3.3-1: Configuration for tests of APIs with VNFM as Producer

4.3.4 Config_prod_NFVO

The configuration config_prod_NFVO shall be implemented to test APIs produced by FUTs which implement a NFVO. The test environment of the virtual element is an NFV platform providing VNFM, VIM and NFVI.

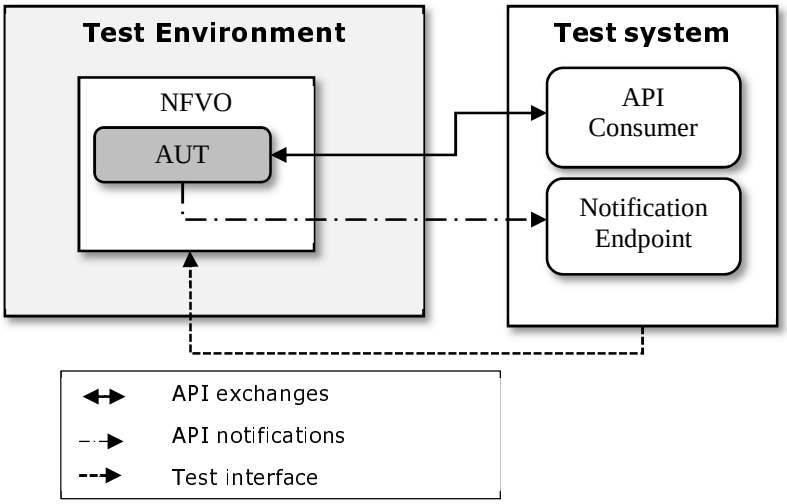

Figure 4.3.4-1: Configuration for tests of APIs with NFVO as Producer

4.3.5 Config_prod_VNFM_GRANT

The configuration config_prod_VNFM_GRANT shall be implemented to test APIs produced by FUTs which implement a VNFM for VNF LCM test cases where an Operation Grant is needed. The test environment of the virtual element is composed by the NFVI where the VE is executed and a NFVO component exposing the VNF Lifecyle Operation Granting API.

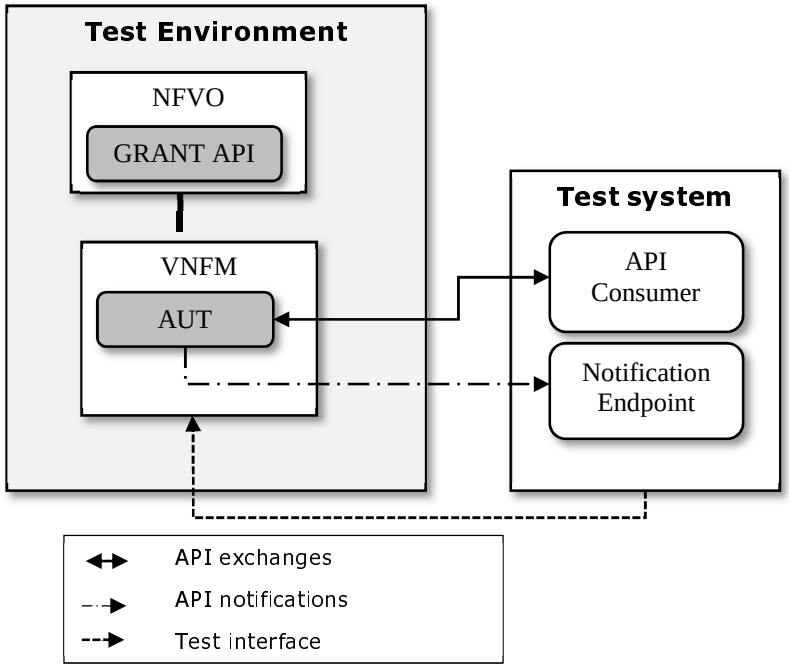

Figure 4.3.5-1: Configuration for tests of APIs with VNFM as Producer, where the VNFLifecycleOperationGranting API is required

4.4 Void

Void.

4.5 Generic Test Description

4.5.1 General

The machine readable language used for the test case implementations is the Robot Framework [i.2].

The Robot code for all test cases is available at: https://forge.etsi.org/rep/nfv/api-tests/2.6.1 and in the accompanying Docx Annexes , which contain the tabular representations of the test descriptions. The primary source for the Test Descriptions is the online repository at ETSI Forge.

A compliant NFV MANO API Conformance test systems shall implement the Test Descriptions as documented in the ETSI Forge repository and in the accompanying annexes which follow the requirements set in the following clauses.

Together with the normative test descriptions, informative implementation of the tests steps is made available as well in the form of "Keywords in Robot Framework" to facilitate adoption of the present work.

Within the Robot Framework files present in the online repository, the test descriptions (which are normative) and test steps implementations (which are informative) can be identified by the heading under which they are grouped, as follows:

- the normative tests descriptions are grouped under the "`*** Test Cases ***`" heading, while
- the informative test steps implementations are grouped under the "`*** Keywords ***`" heading.

NOTE: The text for the headings needs to conform to the Robot Framework language syntax.

4.5.2 Test Description format

Test Descriptions specify the information required to run an "API conformance test". Test Descriptions are defined per test, and table 4.5.2-1 defines the elements of the Test Description.

Table 4.5.2-1: Test Description entries

Elements	Definition
Test ID	The Test ID identifies uniquely the test. Naming convention: the Test ID is the number of the clause in the present the present document which includes the Test Descriptions.
Test Objective	The test objective indicates clearly which requirement is intended to be tested in the test. This part eases the understanding of the Test Description behaviour. This also eases the identification of the requirements, which were used as a basis for the test.
Pre-conditions	The pre-conditions field defines the conditions that the state of the SUT shall verify before undergoing the actual Test Description. In the process of the test, if the preconditions are not met, it leads to the assignment of an Inconclusive verdict. This field may include identifiers of other Tests within the present document which may be executed to bring the SUT into a state which verifies the mandated pre-conditions.
Reference	In the reference row, the Test Description writer indicates, the clauses of NFV SOL specification where the tested requirements are expressed. This information is critical, because it justifies the existence and the behaviour of the Test Description. The reference row may refer to several clauses, separated by semi-colon.
Config ID	The identifier of the applicable test configuration. A test configuration defines the components that shall be included in Test System and SUT to execute the tests and their connections.
Applicability	Only optional functions which are required for this specific Test Description are listed in this field.
Post-conditions	Definition of the events that the SUT is expected to perform or shall not perform, if any, according to the NFV SOL specifications and following the correct execution of the actions in the expected behaviour below. If the post-conditions concur in the definition of the verdict of the test, a step in the Test Case shall be explicitly provided. Multiple post conditions are separated by semi-colon which expresses a logical AND relation.

Elements	Definition
Test Case	A set of steps of the test, carried out by the Test System to verify conformance of the SUT to the Test Objective. The steps are written in the form of Keywords in the Robot Framework language [i.2]. The verdict of the Test shall be se to 'Pass' if all the applicable steps in the Test Case are successfully carried out and meet expected results criteria, it shall be set to 'Fail' if any of the steps fails. The Inconclusive verdict applies in all other cases.

The Test Descriptions are formatted as tables as shown in the example shown in table 4.5.2-2.

Table 4.5.2-2: Test Description entries examples

Test Purpose	
Test ID	A.B.C.D.E
Test Objective	Example Test Objective.
Pre-conditions	Precondition example.
Reference	Reference 1; reference 2.
Config ID	Configuration_example_id
Applicability	Applicability statement example.
Post-conditions	Precondition 1; Precondition 2; Precondition 3.
Test Case	
Step Number One Step Number Two Steps are described with use of below expressions: • GET all [resource] [expression] to query URI of [resource] to SUT by GET operation with [expression]. • GET [resource] [expression] to query individual [resource] to SUT by GET operation with [expression]. • Send [HTTP method] Request to [purpose]/[HTTP method] [resource] to send request to SUT by [HTTP method], i.e. POST, PUT, PATCH, DELETE. Same as "Send [HTTP method] Request to [purpose]". • Check HTTP Response Status [expression] to check if the status code in the response which corresponds to the previous step request from SUT matches [expression]. • Check HTTP Response Header [expression] to check if the contents of HTTP header in the response which corresponds to the previous step request from SUT matches [expression]. • Check HTTP Response Body [expression] to check if the JSON of the body in the payload of the response which corresponds to the previous step request from SUT matches [expression]. • Check Postcondition [resource] Exists to check if [resource] of the SUT is accessible by GET operation. • Check Postcondition [resource] is [expression] to check if [resource] matches [expression] by GET operation. • Check [notification name] Http Request Body [expression] Upon [notification name] received, to check if the JSON of the body in the payload of HTTP post request from SUT matches [expression]. • Trigger [condition] (external action) Test environment prepares [condition] for the particular test to run. • Check [resource] [condition] to check if the attribute of the [resource] in JSON in the payload of the response from SUT matches [condition] according to the "Post-Conditions" or "Pre-conditions" by GET operation.	

4.5.3 Scope of the tests

4.5.3.1 General

In addition to the material described in clause 4.5.1, the behaviours of the SUT shall be tested in terms of:

- expected handling of the messages received from Test System;
- expected actions inside the SUT.

These two items above shall be checked by examining information retrieved by other operations performed by Test Systems e.g. GET or appropriate notification messages. These checks of expected handling and internal actions shall be based on material provided in the referenced specifications including all error cases.

Further checking includes:

- the expected correct response messages sent back to the Test system.

The above item shall be checked by analysing received messages by Test Systems. These checks of response correctness shall be based on material provided in the referenced specifications, including all error cases.

4.5.3.2 General characteristics of the reference points

This clause describes the basic principles of ETSI GS NFV-SOL 002 [2], ETSI GS NFV-SOL 003 [1], and ETSI GS NFV-SOL 005 [3] which are considered in API conformance test.

From the API conformance test point of view, the reference points have below characteristics:

- Asynchronous API and State handling by notifications: applicable for Os-Ma-Nfvo, Or-Vnfm:
  - Due to the varieties of the reasons, e.g. consuming time of process handling, "VNF Lifecycle Management interface" defined in ETSI GS NFV-SOL 003 [1], clause 5 and "NS Lifecycle Management interface" defined in ETSI GS NFV-SOL 005 [3], clause 6 are designed as asynchronous way, i.e. API designed is based on the combination of the specific methods with task resources and state handling by related attributes signalled via notification messages.
  - For example, the operation "Instantiate VNF" is triggered by POST message with task "Instantiate VNF" sent from NFVO to VNFM , then VNFM sends "provisional" response "202 accepted" back to NFVO, which means "accepted request" (ETSI GS NFV-SOL 003 [1], clause 5.4.4). Based on the subscriptions registered in VNFM, the appropriate notification message will be sent toward NFVO with the state of the processing in the VNFM at that time. After finishing the process of instantiating VNF in the VNFM, the notification message with the notification endpoint resource which is set to "operationState=COMPLETED" shall be sent (ETSI GS NFV-SOL 003 [1], clause 5.4.18), which mean that the VNF has been instantiated. In order to verify such "Asynchronous API" characteristics, the Robot code implements "workflow test" which include series of operations required for "VNF Lifecycle Management interface" and "NS Lifecycle Management interface" operations.
- Linked resource: applicable for Os-Ma-Nfvo, Ve-Vnfm, Or-Vnfm:
  - According to ETSI GS NFV-SOL 003 [1], clause 4.3.3.1, ETSI GS NFV-SOL 005 [3], clause 4.3.3.1 and ETSI GS NFV-SOL 002 [2], clause 4.3.3.1, "link" pointing to the resources which may be fetched by API consumer, may be used on the reference point. It is used in order to avoid "big" response messages and to avoid repeating information request messages. Therefore the validity of such linked resources are also verified by test, as same as resources in the response message itself.
- Request and response data values: applicable for Os-Ma-Nfvo, Ve-Vnfm, Or-Vnfm:
  - According to the ETSI GS NFV-SOL 003 [1] and ETSI GS NFV-SOL 002 [2], clause 5.5, a VNF LCM request use the relevant values e.g. set in the valid VNFD, i.e. InstantiateVnfRequest, ScaleVnfRequest, ScaleVnfToLevelRequest, ChangeVnfFlavourRequest, TerminateVnfRequest HealVnfRequst, OperateVnfRequest and ChangeExtVnfConnectivityRequest.
  - According to the ETSI GS NFV-SOL 005 [3], clause 6.5, a NS LCM request uses the relevant values e.g. set in the valid NSD, i.e. InstantiateNsRequest, ScaleNsRequest, UpdateNsRequest, TerminateNsRequest and HealNsRequest.
  - Test system verify if each attribute of the response body uses relevant values e.g. set in the valid VNFD/NSD.

NOTE 1: This is not considered in the present document.

- Grant exchange: applicable for Or-Vnfm:
  - According to ETSI GS NFV-SOL 003 [1], clause 9.1, the VNFM executes Grant Request before certain LCM operations are executed for the direct mode. The test system verifies if each attribute of Grant request body uses relevant values e.g. set in the valid VNFD.

NOTE 2: The value validation is not considered in the present document.

NOTE 3: The Robot code implements "Individual Grant". The test for "Grant exchange in the VNF LCM operations" in the workflow test requires specific test configuration "Config_prod_VNFM_GRANT" specified in clause 4.3, but "Grant exchange" part in the VNF LCM is not considered by the Robot code in the present document.

- VIM registration: applicable for Or-Vnfm:
  - According to ETSI GS NFV-SOL 003 [1] clause 4.4.1.6, the VNFM uses VIM related information according to ETSI GS NFV-SOL 003 [1], clause C.5, the test system import the latest VIM registration before testing.

NOTE 4: The test system imports the values for the VIM related information via test configurations.

4.5.3.3 Basic behaviours of the API producer/consumer and verification steps

4.5.3.3.0 Introduction

Expected behaviours of the API producer specified in the SOL specifications targeted by the tests (i.e. ETSI GS NFV-SOL 002 [2], ETSI GS NFV-SOL 003 [1] and ETSI GS NFV-SOL 005 [3]) are categorized as follows:

- Producer sends event triggered notifications based on consumer subscriptions.
- Producer sends periodic notifications based on the consumer subscriptions.
- Producer executes the requested API operation.
- Consumer fetches the files/package info.

These four categories are expanded hereafter.

4.5.3.3.1 Producer sends event triggered notifications based on consumer subscriptions

Typical example for this case is "Fault management". The producer sends a notification to the consumer with mechanism "filter" and "callback". The Test System checks if the notification aligns to the conditions of the subscription.

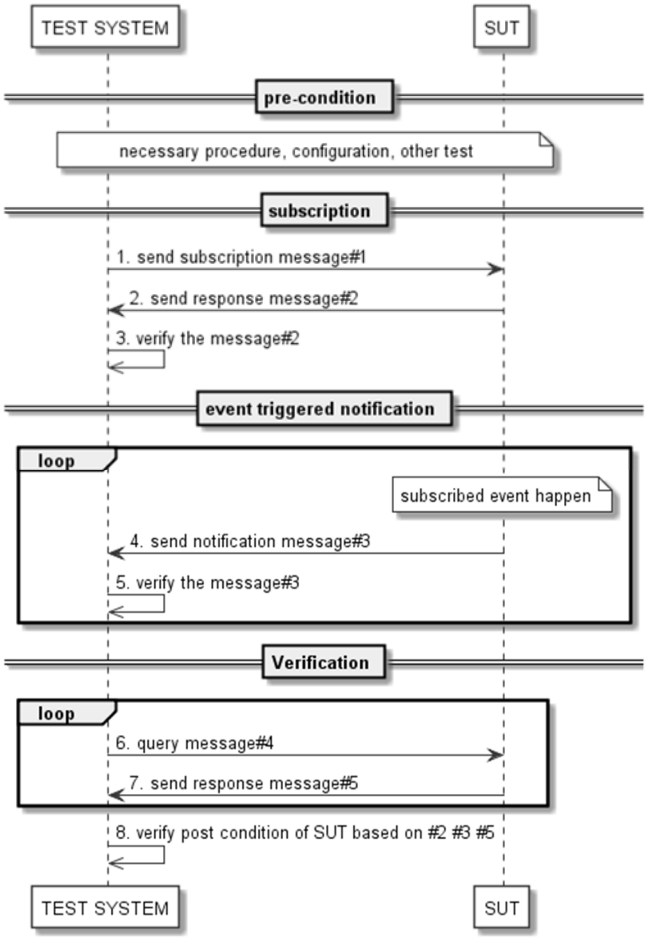

Figure 4.5.3.3.1-1: Verification procedure

The basic method of verification follows the steps as illustrated in figure 4.5.3.3.1-1:

Preconditions: Before sending message#1, required features/functionalities are executed, e.g. API authentication, etc. Other tests as pre-condition are executed beforehand if required:

1) Test System sends subscription message#1. Test System sets the data and attributes of the request body with correct/incorrect cardinalities and values.

2) Upon receiving the subscription message#1 from Test System, the SUT shall handle this subscription according to the requirements defined in SOL specifications, the response message#2 are sent to the Test System.

3) Test System verifies the received response message#2, in terms of expected response code, expected HTTP header, HTTP body, cardinality of the responded data, fulfilment of the requirements described in the description field, and verification aspects shown below, etc.:

   a) Verification aspects: 1-C, 2-C, 2-F, 2-G (see clause 4.5.4).

4) When the events which are matched to the subscribed criteria made by message#1 happen in SUT, notification message#3 sends to the Test System.

NOTE: How to make subscribed event in SUT to match the criteria is out of scope of the present document.

5) Test System verifies the received response message#3, in terms of expected response code, expected HTTP header, HTTP body, cardinality of the responded data, fulfilment of the requirements described in the description field, and verification aspects shown below, etc. steps 4 and 5 can be looped:

   a) Verification aspects: 1-C, 2-C, 2-F, 2-G (see clause 4.5.4).

6) After expected API messages exchange, the Test System may have some verification aspects which require retrieval of information from the SUT, e.g. linked resources embedded in the response message. In that case, the Test System may invoke a query message to the SUT.

7) As a reaction to the step 6, the SUT may send additional messages to the Test System. In this case, the verification is identical to the step 5.

8) In addition to the verification done by step 2, step 3 step 5 and step 7, the Test System verifies the expected post conditions for the SUT.

4.5.3.3.2 Producer sends periodic notifications based on the consumer subscriptions

Typical example for this case is "Performance management". The producer sends a periodic notification to the consumer as requested in the subscription. The Test System checks if the notification aligns to the conditions of the subscription.

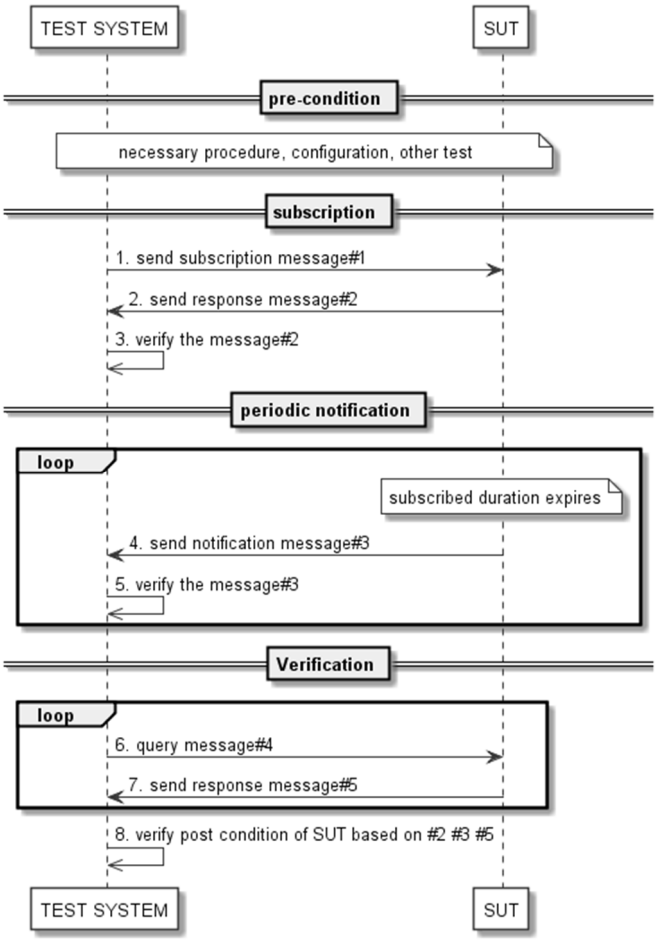

Figure 4.5.3.3.2-1: Verification procedure

The basic method of verification follows the steps as illustrated in figure 4.5.3.3.2-1:

Preconditions: Before sending message#1, required features/functionalities are executed, e.g. API authentication, etc. Other tests as pre-condition are executed beforehand if required:

NOTE: Even figure 4.5.3.3.2-1 shows the series of the message exchanges as sequence flow, the Robot code/Test Descriptions implements the tests per individual interface basis, i.e. the tests for subscription, the tests for notification.

1) Identical to the step 1 of the clause 4.5.3.3.1.

2) Identical to the step 2 of the clause 4.5.3.3.1.

3) Identical to the step 3 of the clause 4.5.3.3.1.

4) When the reporting period which is subscribed to the SUT by message#1 expires, notification message#3 sends to the Test System.

5) Identical to the step 5 of the clause 4.5.3.3.1.

6) Identical to the step 6 of the clause 4.5.3.3.1.

7) Identical to the step 7 of the clause 4.5.3.3.1.

8) Identical to the step 8 of the clause 4.5.3.3.1.

4.5.3.3.3 Producer executes the requested API operation

Typical example for this case is "Life Cycle Management". Based on the concept "Asynchronous API", the producer starts the operation by receiving a request message from the consumer, and sends some notifications back based on the operational status. The Test System checks if the procedure is correct, i.e. start with a request, send notifications with an appropriate status, complete operation and correct status of the resources and status of the producer.

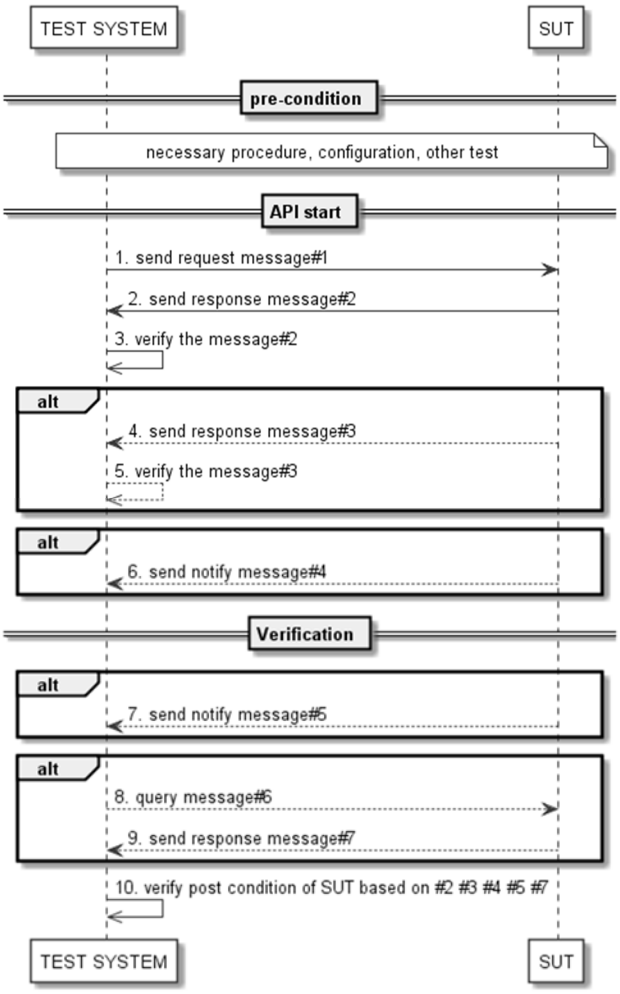

Figure 4.5.3.3.3-1: Verification procedure

The basic method of verification follows the steps as illustrated in figure 4.5.3.3.3-1:

Preconditions: Before sending message#1, required features/functionalities are executed, e.g. API authentication, etc. Other tests as pre-condition are executed beforehand if required:

1) Test System sends request message#1. Test System sets the data and attributes of the request body with correct/incorrect cardinalities and values.

2) Upon receiving the request message#1 from Test System, the response message#2 are sent to the Test System with an appropriate response code and HTTP body based on the result of the SUT executed procedure.

3) Test System verifies the received response message#2, in terms of expected response code, expected HTTP header, HTTP body, cardinality of the responded data, fulfilment of the requirements described in the description field, and verification aspects shown below, etc.:

   a) Verification aspects: 1-C, 2-A, 2-C, 2-F, 2-G (see clause 4.5.4).

4) Optionally, the API may have more than 1 message exchange in particular for messages sent from the SUT to the Test System. In that case this step follows step 2.

5) Identical to the step 3.

6) Optionally, the SUT may send notification messages to the Test System which shows the status of the processing of the messages in the SUT, e.g. STARTING/PROCESSING, etc., based on the consumer subscriptions.

7) After SUT completed the requested processing in the SUT, the SUT may send the notification message to the Test System which shows the status of the processing "COMPLETED".

8) After expected API messages exchange, the Test System may have some verification aspects which require retrieval of information from the SUT, e.g. linked resources embedded in the response message. In that case, the Test System may invoke a query message to the SUT.

9) As reaction to the step 8, the SUT may send additional messages to the Test System. In this case, the verification follows step 3.

10) In addition to the verification done by step 2, step 3, step 4, step 5 and step 7 (optional), the Test System verifies the expected post conditions for the SUT.

4.5.3.3.4 Consumer fetches the files/package info

Typical example for this case is fetching files of a VNFD, VNF package, NSD and PNFD. The Test System checks if the file or package info is correctly received.

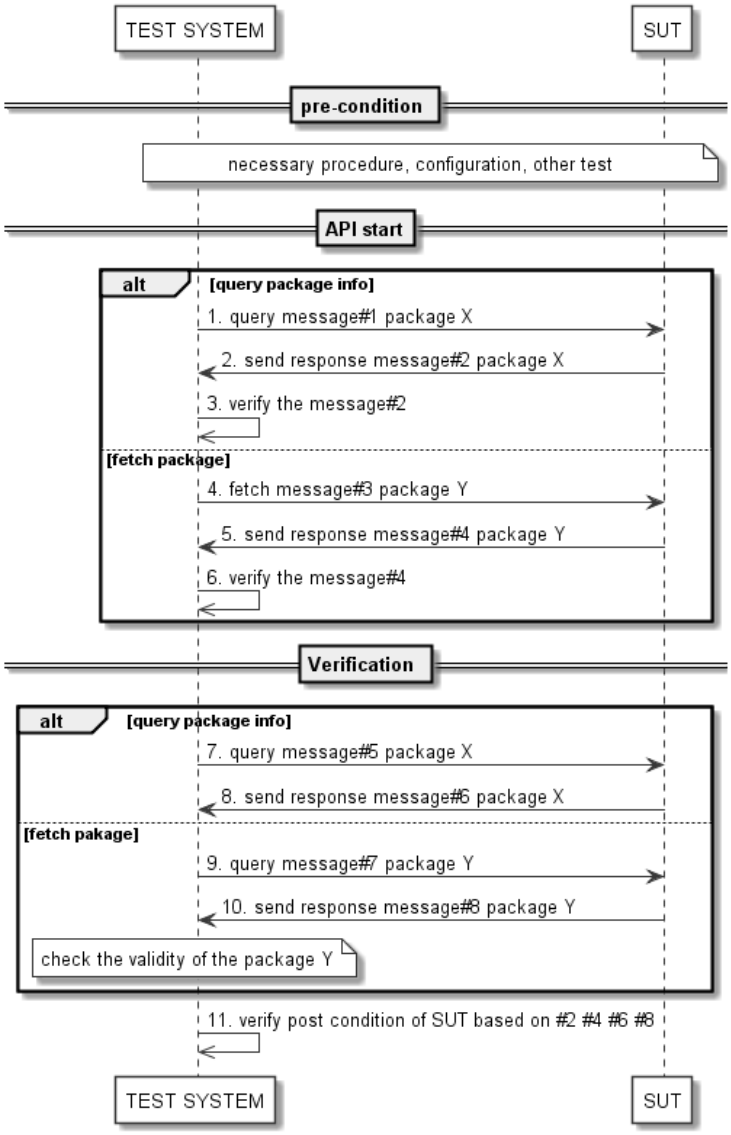

Figure 4.5.3.3.4-1: Verification procedure

The basic method of verification follows the steps as illustrated in figure 4.5.3.3.4-1:

Preconditions: Before sending message#1, required features/functionalities are executed, e.g. API authentication, etc. Other tests as pre-condition are executed beforehand if required:

NOTE 1: Even figure 4.5.3.3.1-1 shows the series of the message exchanges as sequence flow, the Robot code/Test Descriptions implements the tests per individual interface basis, i.e. the tests for query the package info, the tests for fetch the packages.

1) Test System sends query message#1 for information of the package X. Test System sets the data and attributes of the request body with correct/incorrect cardinalities and values.

2) Upon receiving the query message#1 from Test System, the SUT shall handle this subscription according to the requirements defined in SOL specifications, the response message#2 are sent to the Test System.

3) Test System verifies the received response message#2, in terms of expected response code, expected HTTP header, HTTP body, cardinality of the responded data, fulfilment of the requirements described in the description field, and verification aspects shown below, etc.:

   a) Verification aspects: 1-C, 2-C, , 2-F, 2-G (see clause 4.5.4).

4) Alternately to the step 1, Test System sends fetch message#3 to retrieve the package Y. Test System sets the data and attributes of the request body with correct/incorrect:

   a) Verification aspects: 2-F in clause 4.5.4.

5) Upon receiving the fetch message#3 from Test System, the SUT shall handle this request according to the requirements defined in SOL specifications, the response message#4 with package Y are sent to the Test System with an appropriate response code and HTTP body based on the result of the SUT executed procedure.

6) Identical to the step 3, for message#4.

7) After expected API messages exchange, the Test System may have some verification aspects which require retrieval of information from the SUT, e.g. linked resources embedded in the response message. In that case, the Test System may invoke a query message to the SUT. And in order to the verification of the package X information fetched by step 2, the Test System sends query message#5 to the SUT.

8) As reaction to the step 7, the SUT may send additional messages to the Test System. In this case, the verification is identical to the step 3.

9) The Test System may have some verification aspects which require retrieval of information from the SUT, e.g. linked resources embedded in the response message. In that case, the Test System may invoke a query message to the SUT. And in order to the verification of the package Y information fetched by step 5, Test System sends query message#7 to the SUT.

10) As reaction to the step 9, the SUT may send additional messages to the Test System. In this case, the verification is identical to the step 10.

11) In addition to the verification done by step 3 and step 6, the Test System verifies the expected post conditions for the SUT.

4.5.3.4 Workflow test considerations

The asynchronous API concept introduced in ETSI GS NFV-SOL 002 [2], ETSI GS NFV-SOL 003 [1] and ETSI GS NFV-SOL 005 [3] describes set of lifecycle operations and relations among them. Asynchronous API, though described in form of flow as informative in the base specifications, is one of the key and critical design concepts of NFV SOL specifications and for deployment of NFV systems. Therefore, the present document includes a set of workflow Test Descriptions to test such characteristics and, in particular, to include checks on the postconditions of the request for LCM operations.

NOTE 1: As an example, the request of instantiation of a VNF expects as a post condition the asynchronous notifications of the status change of the VNF throughout the instantiation process. A workflow Test Description includes checks on the actual notifications of such status changes.

NOTE 2: The workflow Test Descriptions for ETSI GS NFV-SOL 002 [2] aspects are not considered in the present document.

The workflow Test Descriptions are listed in annex B.

4.5.4 Verification

4.5.4.0 Introduction

Since the scope of the API conformance test includes syntax and semantics verifications, this clause defines the methodology of the verification executed by the Test System.

4.5.4.1 Common verification aspects

The common verification aspects checked by the Test System are defined in table 4.5.4.1-1.

Table 4.5.4.1-1: Common verification aspects

Verification aspects	Notes
1-A: URI handling	If the SUT receives an invalid URI by HTTP method with either an invalid path or invalid values for the resources, the SUT shall return the appropriate error response code to the Test System according to the SOL specification. The Test System tests the support of HTTP over TLS version 1.2. See and follow clause 4.1 of ETSI GS NFV-SOL 013 [5].
1-B: Attribute selectors	Expected attributes or sub-resources shall be sent from the SUT according to attribute selector parameter by a query operation from the Test System. The Test System tests all of the operations with parameters specified in table 5.3.2.1-1 of ETSI GS NFV-SOL 013 [5] if those work correctly. See and follow clause 5.3 of ETSI GS NFV-SOL 013 [5].
1-C: HTTP header fields	Expected HTTP headers shall be sent from the SUT. The Test System tests all of the operations with parameters if those work correctly. See and follow clause 4.2 of ETSI GS NFV-SOL 013 [5].
1-D: Error reporting	In case of some types of errors occurred in the SUT, the error response with type ProblemDetails data are returned to the Test System. The Test System tests all attributes of the ProblemDetails for error Response Codes explicitly specified in the each of Response body part of the tables for "Details of the request/response on the resources" in the baseline SOL specifications regarding the error experienced by the SUT. See note. See and follow clause 6 of ETSI GS NFV-SOL 013 [5].
1-E: API authorization	For the case of making an API call, OAuth2.0 is required for authentication between API consumer and API producer. For the case of the notification, OAuth2.0 or HTTP Basic authentication is used. From the API producer point of view, if the passed access token is invalid, API producer should return appropriate errors, e.g. 401 unauthorized, 403 insufficient scope. See and follow clause 8 of ETSI GS NFV-SOL 013 [5].
NOTE: The test system does not support the test for the exhaust error Response Codes specified in ETSI GS NFV-SOL 013 [5] for the "4xx/5xx" part of the each Response body filed in the tables for "Details of the request/response on the resources".	

4.5.4.2 Verification aspects for individual API

The verification aspects for individual APIs checked by the Test System are defined in table 4.5.4.2-1.

Table 4.5.4.2-1: Common verification aspects for each API

Verification aspects	Notes
2-A: Asynchronous API	As described in clause 4.5.3, one of the typical behaviours of Asynchronous API, the requested behaviours are not completed by a response message but as a combination of notifications. See note 1.
2-B: Void	Void.
2-C: HTTP Method support	The Test System verifies all of the Supported HTTP method, e.g. all types of the specified method with all types of the specified response codes.
2-D: Void	Void.
2-E: Void	Void.
2-F: Response messages validation	The Test System checks the syntax of the HTTP header and body, the cardinalities and the conditions described in the description part and/or NOTE part of the tables for definition of the data types in the SOL specifications. The "link" attributes of the data model are also subject for validation in term of intended values of the resources in the SUT. See note 2.
2-G: Validation of the resources in the SUT after API operation	The Test System checks if the related resources in the SUT are in the expected status after having received the response messages, by checking the notifications from the SUT or by initiating a GET request towards the SUT.

Verification aspects	Notes
2-H: Void	Void.
NOTE 1: The Robot code implements "workflow test", which includes series of operations required for VNF Lifecycle Management interface and NS Lifecycle Management interface. Special considerations on "workflow Test Descriptions" are described in clause 4.5.3.4. NOTE 2: The current version of the Robot code implements as following: – JSON schema validation is supported. – Conditional check for the requirements of the data type, i.e. requirements described in the description part of the data type table in the SOL specifications is not fully performed.	

4.5.4.3 Verification aspects not considered

The current version of the present document and test code is not considering the aspects for verification listed in table 4.5.4.3-1.

Table 4.5.4.3-1: Exception of the verification aspects for API conformance test

Verification aspects not considered by this version of document and test code	Notes
Multiple times notification test	Multiple times notification test based on the conditions subscribed. See clause 4.5.3.3.2.

5 Void

6 Void

7 Void

Annex A (informative): Known Issues

Table A-1 lists Test Descriptions that have known issues that could impact the understanding or the execution of the tests on "NotificationEndpoint".

Table A-1: Known issues

Clause IDs	Issue Description
5.3.2.17, 5.3.3.5, 6.3.4.6, 6.3.5.19, 7.3.1.34, 7.3.5.5	Test Descriptions for "NotificationEndpoint" test to dispatch the notifications on "Subscription" resources, i.e. does not test "NotificationEndpoint".

Table A-2 lists the missing Test Descriptions for "NotificationEndpoint".

Table A-2: Missing Test Descriptions on tests for "NotificationEndpoint"

Clause IDs	Description
5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.3.5	For the Os-Ma-Nfvo Reference Point "NSD Management interface" "NS Lifecycle Management interface" "NS Fault Management interface" "NS Performance Management interface" and "VNF Package Management interface", the Test Descriptions for notification endpoint are missing.
6.3.2, 6.3.3	For the Ve-Vnfm Reference Point "VNFIndicators Interface" and "VNF Performance Management Interface", the Test Descriptions for notification endpoint are missing.
7.3.3, 7.3.4, 7.3.5, 7.3.6, 7.3.7	For the Or-Vnfm Reference Point "VNF Package Management Interface" and "PerformanceManagement Interface" "VNF Fault Management Interface" "VNF Indicator Interface" and "Virtualised Resources Quota Available Notification interface", the Test Descriptions for notification endpoint are missing.

Annex B (informative): Workflow Test Descriptions

Table B.1 contains the exhaustive list of Test Descriptions that are workflows, as defined in clause 4.5.3.4.

Table B.1: List of workflow Test Descriptions

Test ID	Test Title
5.3.2.18.1	NS Instance Creation
5.3.2.19.1	NS Instance Deletion
5.3.2.20.1	Heal Flow of NS lifecycle management operations
5.3.2.21.1	Instantiate Flow of NS lifecycle management operations
5.3.2.22.1	Scale Flow of NS lifecycle management operations
5.3.2.23.1	Terminate Flow of NS lifecycle management operations
5.3.2.24.1	Update Flow of NS lifecycle management operations
7.3.1.19.1	Cancel a VNF LCM Operation - STARTING
7.3.1.19.2	Cancel a VNF LCM Operation - PROCESSING - ROLLING_BACK
7.3.1.20.1	Change external connectivity of VNF Workflow
7.3.1.21.1	Change VNF Flavour Workflow
7.3.1.22.1	Create VNF Instance Resource
7.3.1.23.1	Delete VNF Instance Resource
7.3.1.24.1	Fail a VNF LCM Operation Workflow
7.3.1.25.1	Heal a VNF Instance
7.3.1.26.1	VNF Instantiation
7.3.1.27.1	Modify info of a VNF Instance
7.3.1.28.1	Operate a VNF Instance
7.3.1.29.1	Retry VNF LCM Operation - Successful
7.3.1.29.2	Retry VNF LCM Operation - Unsuccessful
7.3.1.30.1	Rollback a VNF LCM Operation - Successful
7.3.1.30.2	Rollback VNF LCM Operation - Unsuccessful
7.3.1.31.1	VNF Instance Scale To Level
7.3.1.32.1	VNF Instance Scale Out
7.3.1.33.1	Terminate a VNF Instance

Annex C:
Void

Annex D (normative): Word format presentation of the test suite for the Os-Ma-Nfvo Reference Point

The Microsoft® Word format presentation of the high level tests for the Os-Ma-Nfvo is generated from the Robot Framework test suite using the tool robot2doc available at [i.3]. The format is provided to assist the readers that are not familiar with the Robot Framework syntax language.

The Microsoft® Word format presentation of the high level tests for the Os-Ma-Nfvo is the file Tests-SOL005-OsMaNfvo_261.docx contained in archive gs_nfv-tst010v020601p0.zip which accompanies the present document.

The normative test descriptions and the informative test steps implementations are available in the Robot Framework at https://forge.etsi.org/rep/nfv/api-tests/tree/2.6.1/SOL005.

Annex E (normative): Word format presentation of the test suite for the Ve-Vnfm Reference Point

The Microsoft® Word format presentation of the high level tests for the Ve-Vnfm is generated from the Robot Framework test suite using the tool robot2doc available at [i.3]. The format is provided to assist the readers that are not familiar with the Robot Framework syntax language.

The Microsoft® Word format presentation of the high level tests for the Ve-Vnfm is the file Tests-SOL002-VeVnfm_261.docx contained in archive gs_nfv-tst010v020601p0.zip which accompanies the present document.

The normative test descriptions and the informative test steps implementations are available in the Robot Framework at https://forge.etsi.org/rep/nfv/api-tests/tree/2.6.1/SOL002.

Annex F (normative): Word format presentation of the test suite for the Or-Vnfm Reference Point

The Microsoft® Word format presentation of the high level tests for the Or-Vnfm is generated from the Robot Framework test suite using the tool robot2doc available at [i.3]. The format is provided to assist the readers that are not familiar with the Robot Framework syntax language.

The Microsoft® Word format presentation of the high level tests for the Or-Vnfm is the file Tests-SOL003-OrVnfm_261.docx contained in archive gs_nfv-tst010v020601p0.zip which accompanies the present document.

The normative test descriptions and the informative test steps implementations are available in the Robot Framework at https://forge.etsi.org/rep/nfv/api-tests/tree/2.6.1/SOL003.

Annex G (informative): Change History

Version	Date	Information about changes
2.6.1.0.0.1	2020-03-30	Initial draft
2.6.1.0.0.2	2020-05-19	NFVTST(20)000018 - TST010ed261 - Removal of Annex A Known Issues NFVTST(20)000017 - TST010ed261 - Removal of TD clauses NFVTST(20)000026r3 - TST 010 - Code annexes templates and examples
2.6.1.0.0.3	2020-05-26	NFVTST(20)000020 - TST010ed261 - Approval of STF developments Merge Requests
2.6.1.0.0.4	2020-06-01	NFVTST(20)000022r3 - TST010ed261 4.5.4 Verification aspects update (CR format version of NFVTST(20)000001r5) NFVTST(20)000039 - TST010ed261 - Corrections from draft review and Plugtest NFVTST(20)000040 - TST010ed261 - Adjustments to Annex B
2.6.1.0.0.5	2020-07-13	NFVTST(20)00044r1 - TST010 NotificationEndpoint

History

Document history		
V2.4.1	March 2020	Publication
V2.6.1	September 2020	Publication