ETSI Plugtests Report V1.0.0 (2021-04)

ETSI Remote NFV&MEC API Plugtests
February 2021

Keywords
Testing, Interoperability, API, Conformance, NFV,
MEC, MANO, VNF, VIM, NFVI

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

Executive Summary

The NFV&MEC API Plugtests was organised by the ETSI Centre for Testing and Interoperability and run remotely for 1 month, from February 1st to the 28th 2021. The main goals of this remote event were to:

- Allow participants to self-evaluate the conformance of their API server implementations with Network Function Virtualisation and Multi-Access Edge Computing API Specifications,
- Validate and gather feedback on ETSI NFV and MEC API and Conformance Test Specification and associated Robot Test Suites

The NFV and MEC Test Suites were made available over the Test Automation Platform hosted by ETSI. Participants connected their implementations (Functions Under Test) to a secure network and could arrange as many individual test sessions as desired, according to their time-zone and availability. Participants could be completely autonomous when running the testing, or request support from the ETSI Plugtests Team whenever needed.

Overall, 29 NFV and MEC Test Suites were exercised and over 1200 test results recorded in the Test Automation Platform, with the following main highlights:

- Availability of several versions of the NFV Test Suites, allowing participants to choose the version of the API to test (or to test several versions). This included the latest 2.7.1 version made available in its current status of stable draft. This latest version includes support for automated verification of exchanged data related to NS Descriptors within the API Conformance Testing.
- Steady increase in the number and scope of tested NFV APIs and features, including this time APIs exposed by the VNFM over the Ve-Vnfm reference point (specified in NFV-SOL002). Moreover, the total number of NFV test results increased by 3% with respect of the previous API event.
- Improvement in the overall success rate of the API testing activities. For NFV the success rate has raised by 6% from previous event, while for MEC the success rate increased by 22%.
- Precious feedback on NFV and MEC API and Test Specifications was collected: over 70 issues were identified during the event and, for a high number of the issues raised on the Test Specifications, a fix coud be provided to participants so that the test could be re-run and new result recorded. In addition, 4 important findings have been collected and shared with NFV and MEC Industry Specification Groups for further discussion. All this feedback is reported in Section 10.
- Continuous support and active participation of different open source communities : AdvantEDGE, CNCF Kubernetes, ETSI Open Source MANO, OpenStack and StarlingX solutions were widely present in the testing through one or multiple distributions.

In addition to the API Conformance Testing, two experimental activities were hosted by external communities and during the event and offered to Network Function providers. The outcome of these two tracks is reported in Section 10.3.

- A track allowing to migrate legacy OSM descriptors to the latest, NFV-SOL006 conformant format, was conducted by the ETSI Open Source MANO Community.
- A track allowing to self-asses the level of conformance of containerized functions to the Cloud Native principles defined by the Cloud Native Computing Foundation (CFCF). The testing was complemented by a comparative analysis of the Cloud Native principles covered by the CNF Conformance Test Suite and the ETSI NFV-EVE011 Specification.

Over 30 organizations and 5 open source communities contributed to the event, providing Network Functions, Virtual Network Function Managers, Network Function Virtualisation Orchestrators, MEC Platforms and Services, Test Environments, Test Tools and Technical Support. A detailed list of all the different participating organizations and solutions is available in section 5.

The NFV&MEC API Plugtests in February 2021 provided a great opportunity for participants to get their API conformance tested in preparation of the NFV&MEC Interoperability Plugtests, planned for October 2021.

1 Introduction

This Plugtests focused on NFV and MEC API Conformance testing, allowing participants to self-assess the level of conformance of their Functions Under Test with ETSI NFV and MEC API Specifications. At the same time, running the ETSI NFV and MEC API Conformance Robot Test Suites against a significant number of different participating implementations, allows ETSI to validate the Test Suites and increase the quality of the Test Specifications,

In order to enable remote interaction among participating Functions Under Test, Test Environments and the Test Automation Platform hosted by ETSI, a dedicated VPN based network was used to interconnect local and remote systems in a reliable and secure way: the NFV Plugtests HIVE (Hub for Interoperability and Validation at ETSI). All the participating implementations, Functions Under Test, Test Environments and the Test Automation Platform were connected and/or accessible through the HIVE network

This document provides and overview of this remote Plugtests, including participation, test plans. test infrastructure, overall results, and major findings.

.

2 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 http://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.

[NFV003] ETSI GS NFV003 “Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV”

[NFV-TST010] ETSI GS NFV-TST010 V2.4.1, V2.6.1 and V2.7.1: “Network Function Virtualisation (NFV) Release 2; Testing; API Conformance Testing Specification”

[NFV-SOL001] ETSI GS NFV-SOL001 V2.7.1:” Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification”

[NFV-SOL002] ETSI GS NFV-SOL002 V2.4.1, V2.6.1 and V2.7.1: “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Ve-Vnfm Reference Point”

[NFV-SOL003] ETSI GS NFV-SOL002 V2.4.1, V2.6.1 and V2.7.1: “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point”

[NFV-SOL005] ETSI GS NFV-SOL005 V2.4.1, V2.6.1 and V2.7.1: “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point”

[NFV-SOL006] ETSI GS NFV-SOL006 V2.7.1: “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on YANG Specification”

[NFV-SOL013] ETSI GS NFV-SOL013 V2.7.1: “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; Specification of common aspects for RESTful NFV MANO APIs”

[NFV-EVE011] ETSI GS NFV-EVE011: “Network Functions Virtualisation (NFV) Release 3; Virtualised Network Function; Specification of the Classification of Cloud Native VNF implementations”

[MEC001] ETSI GS MEC001: “Multi-access Edge Computing (MEC); Terminology”

[MEC010-2] ETSI GS MEC010-2 V2.1.1: “Multi-access Edge Computing (MEC); MEC Management; Part 2: Application lifecycle, rules and requirements management”

[MEC011] ETSI GS MEC 011 V2.1.1: “Multi-access Edge Computing (MEC); Edge Platform Application Enablement”

[MEC012] ETSI GS MEC011 V2.1.1: “Multi-access Edge Computing (MEC); Radio Network Information API”

[MEC013] ETSI GS MEC012 V2.1.1: “Multi-access Edge Computing (MEC); Location API”

[MEC014] ETSI GS MEC013 V1.1.1: “Multi-access Edge Computing (MEC); UE Identity API”

[MEC015] ETSI GS MEC013 V1.1.1: “Multi-Access Edge Computing (MEC); Traffic Management APIs”

[MEC016] ETSI GS MEC013 V1.1.1: “Multi-access Edge Computing (MEC); UE application interface”

[MEC021] ETSI GS MEC013 V2.1.1: “Multi-access Edge Computing (MEC); Application Mobility Service API”

[MEC029] ETSI GS MEC013 V2.1.1: “Multi-access Edge Computing (MEC); Fixed Access Information API”

[MEC-DEC032] ETSI GS MEC032 V2.1.1: “Multi-access Edge Computing (MEC); API Conformance Test Specification;”

[NFV-ICS] NFV Implementation Conformance Statement https://nfvwiki.etsi.org/images/NFV_ICS.pdf

[MEC-ICS] MEC Implementation Conformance Statement https://nfvwiki.etsi.org/images/MEC_ICS.pdf

[FORGE] ETSI Forge https://forge.etsi.org

[NFV-ROBOT-TS] Robot Test Suite for NFV API Conformance https://forge.etsi.org/rep/nfv/api-tests

[MEC-ROBOT-TS] Robot Test Suite for MEC API Conformance https://forge.etsi.org/rep/mec/gs032p3-robot-test-suite

[NFV-ISSUE-TR] NFV API Conformance Robot TS Issue Tracker https://forge.etsi.org/rep/nfv/api-tests/issues

[MEC-ISSUE-TR] MEC API Conformance Robot TS Issue Tracker https://forge.etsi.org/rep/mec/gs032p3-robot-test-suite/issues

[NFV-API2021-TR] NFV API Test Results per Test Case https://nfvwiki.etsi.org/images/NFVMEC_API_Plugtests_2021_NFV_Test_Case_Results.pdf

[CNFC-CNF-TS] CNCF CNF Test Suite v0.10.2 https://github.com/cncf/cnf-conformance/releases/tag/v0.10.2

3 Abbreviations

For the purposes of the present document, the terms and definitions given in [NFV003], [NFV-TST010] and [MEC001] apply.

4 Technical and Project Management

4.1 Scope

The main goal of the remote NFV&MEC API Plugtests was to enable participants to run individual Test Sessions over the Test Automation Platform hosted by ETSI and supported by the Plugtests team, allowing to:

- validate the Robot Test Suites for the NFV API Conformance Test Specification [NFV-TST010] and MEC API Conformance Test Specification [MEC-DEC032]
- allow participants to assess the level of conformance of their Functions Under Test (VNFs, VNFMs, NFVOs, MEC Platforms and Services with NFV and MEC API Specifications: [NFV-SOL002], [NFV-SOL003], [NFV-SOL005], [MEC010-2], [MEC011], [MEC012], [MEC013], [MEC014], [MEC015], [MEC016], [MEC021] and [MEC029]

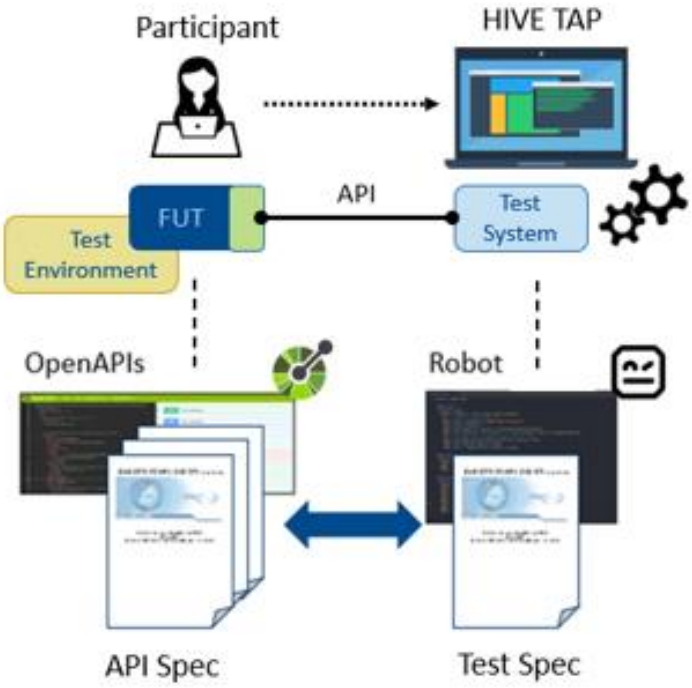

Figure 1. Remote NFV&MEC API Plugtests 2021 scope

In addition to the API Testing the following tracks were also offered to participants:

- OSM to [NFV-SOL006] Descriptor translation –Supported by the ETSI Open Source MANO (OSM) Community, this track was open to Network Function providers having legacy OSM Descriptors. The track allowed participants to migrate such Descriptors to the latest OSM Descriptor format, compliant with the standardized format specified in [NFV-SOL006].
- CNCF CNF Conformance testing – This experimental track supported by the CNCF, allowed participants to assess the level of conformance of their CNFs to the Cloud Native Principles defined by the CNCF. A special attention was given to the communalities and differences of CNCF principles with respect to ETSI NFV specifications, in concrete with [NFV-EVE011]

4.2 Timeline

The preparation of the Remote NFV&MEC API Plugtests 2021 started at the end of 2020 and run through different phases as described in the figure below.

Two events were scheduled in 2021, a Remote event during the month of February fully dedicated to NFV&MEC API Testing and a 1 weeklong (possibly) face to face event in October, dedicated to multi-vendor NFV&MEC interoperability. Measures were taken to fall back this second event into a remote (and possibly longer) format, in case the COVID-19 pandemic recovery would still not allow for face to face event in October 2021.

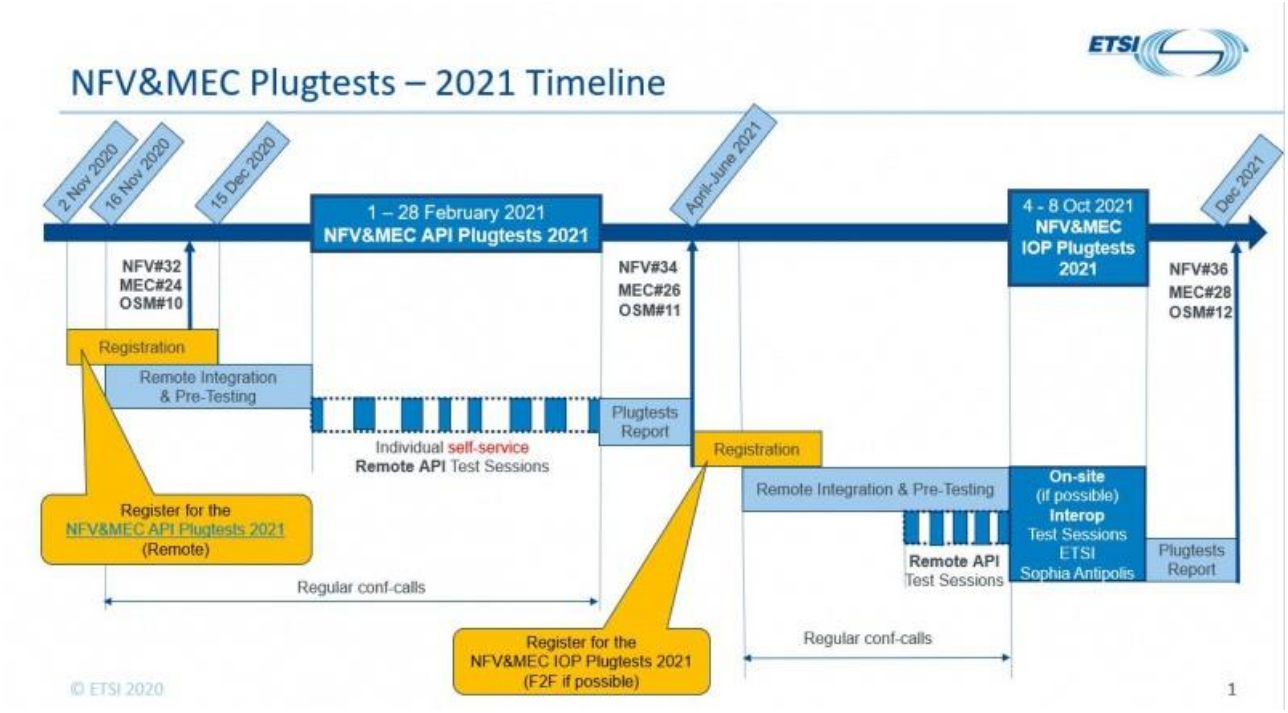

Figure 2. NFV&MEC Plugtests 2021 timeline

Registration to the NFV&MEC API Plugtests was open until mid-December 2020 for any organisation willing to join with a Function Under Test, or to support the testing. The possibility of participating as observers was offered to network operators and academia.

Following a phase of remote integration and pre-testing, participants were invited to schedule their individual test sessions at their best convenience, and to request support from the Plugtests Team when needed.

The remote API Plugtests was an extremely useful step to validate participating APIs and prepare for the Interoperability event planned for the second half of the year.

4.3 Communication Tools

Several electronic means were made available to enable inter and intra team communication during the preparation and testing phases of this Remote Plugtests.

4.3.1 Wiki

The NFV Plugtests Programme wiki was used to compile all the Plugtests related information:

- Participation
- Functions Under Test
- HIVE status

- Test Plans, additional tracks
- Test Automation Platform Documentation
- Plugtests Team contacts
- Slack access
- Conf-call calendar, agenda, and minutes

4.3.2 Regular conf-calls

Participants and organisers attended regular conf-calls during the preparation and testing phase of the event in order to discuss the event scope, connectivity to the HIVE, test plans, issues with the tools, findings, bugs... all the relevant findings discussed during these calls are compiled in chapter 10 Plugtests Outcome

4.3.3 Instant Messaging

Participants were requested to join a dedicated Slack Workspace allowing them to interact with other participants and organisers. Private and public channels were created to support the different testing activities. Participants could launch a video call from these channels anytime if they needed to share screens or discuss

5 Participation

5.1 Functions Under Test

The tables below summarise the different Functions Under Test provided by the Plugtests participants. In order to identify the APIs targeted for the testing, and to make sure the appropriate test suites would be available in the Test Automation Platform, each of the participating FUTs was requested to fill an Implementation Conformance Statement (ICS). The ICS templates can be found at [NFV-ICS] and [MEC-ICS].

5.1.1 NFs

The table below summarizes the Network Functions (Virtualised or Containerized) that participated to any of the available tracks:

- [NFV-SOL002] API Conformance,
- OSM Descriptors translation to [NFV-SOL006]
- CNCF CNF Conformance (experimental)

Organisation	Solution	Specs	Teams Locations	Short Description
A10 Networks	Thunder CGN/FW	SOL001 SOL006	Germany	VNF/CNF: CGN and FW application
Intracom Telecom	fs|cdn Anywhere	SOL006	Greece	CNF:5G vCDN
	mMTC Slicing	SOL006	Greece	CNF: 5G Network Slicing solution for IoT and Edge use cases, developed as an extension of LF EdgexFoundry
Mobileum	NTR	SOL001	India	CNF: Network Traffic Steering (NTR) connects to LTE core components like MME, HSS.
Telenity	Canvas SMSC	SOL006	Turkey	VNF: Short Messaging Service Center
	Canvas PROV	SOL006	Turkey	VNF: Provisioning and Screening for SMSC
	Canvas OPS	SOL006	Turkey	VNF: Operational Support and Reporitng for SMSC

Table 1. NFs Under Test

5.1.2 VNFMs

The table below summarizes the Virtualized Network Functions Managers (VNFMs) that participated to NFV SOL002 or SOL003 API Conformance Testing

Organisations	Name	Specs	Teams Locations	Short Description
Fujitsu	Openstack Tacker	SOL001 SOL003	Japan	Based on Openstack Wallby
Luxoft	SDL	SOL001 SOL003	Romania	NFVO and G-VNFM
NEC	Openstack Tacker	SOL002 SOL003	India Japan	Based on Openstack Victoria
Ubiqube	OpenMSA	SOL001 SOL003	Ireland, France India	Based on MSActivator 2.x Integrated Automation Platform

Table 2. VNMFs Under Test

5.1.3 NFVOs

Organisations	Name	Specs	Team(s) Location	Short Description
Canonical	Charmed OSM	SOL005 SOL006	EU Canada	Based on OSM Release NINE
Canonical Tata Elxsi Whitestack	OSM Release NINE	SOL005 SOL006	EU India Chile, Peru	Upstream OSM Release NINE supported by the OSM Community
Cisco	NFVO	SOL001 SOL003 SOL005 SOL006	US	Cisco NFVO
DZS Inc	RIFT.ware	SOL001 SOL003 SOL005	USA India	Carrier Grade NFVO and GVNFM tailored for Network Automation and 5G Slicing
Ericsson	Cloud Manager	SOL001 SOL003 SOL005	Hungary Ireland	Ericsson's NFVO solution supporting ETSI NFV SOL001 v2.5.1, SOL003 v2.4.1 and SOL005 v2.4.1 specifications
Fujitsu	Openstack Tacker	SOL001 SOL003 SOL005	Japan	Based on Openstack Release Wallby
Luxoft	SDL	SOL001 SOL003 SOL005	Romania	NFVO and G-VNFM
Tata Elxsi	TEOSM	SOL005 SOL006	India	Based on OSM Release NINE
Ubiqube	OpenMSA	SOL001 SOL003 SOL005	Ireland France India	Based on MSActivator 2.x Integrated Automation Platform
Whitestack	WhiteNFV	SOL005 SOL006	Chile Peru	Based on OSM Release NINE

Table 3. NFVOs Under Test

5.1.4 MEC Platforms / Services

Organisations	Name	Specs	Team(s) Location	Short Description
Fondazione LINKS	LocationAPISimulator	MEC013	Italy	Developer tool to support the creation of applications consuming MEC Location API.
Interdigital	AdvantEDGE	MEC012 MEC013	Canada USA	Open Source Mobile Edge Emulator Platform

Table 4. MEC Plat Under Test

5.2 Test Environments

Organisations	Name	LAB	Location	Short Description
Canonical	Charmed Openstack	Canonical @OSM Remote Lab	EU Canada	OpenStack Ussari
	Charmed Kubernetes			K8s v1.20
Wind River	WRCP 20.06 & WRO 20.10	Wind River CA @OSM Remote Lab	USA	Cloud Native K8s v1.18.1 OpenStack Train Compute Nodes
	WRCP 20.06	Wind River Kista @OSM Remote Lab	Sweden	Cloud Native K8s v1.18.1
Wind River Lenovo	Starling X	Lenovo Lab (SR630/SR650 Gen2)	USA	StarlingX R3 - K8s
XFlow	RHOSP 13	xFlow Lab	Pakistan	Red Hat Openstack 13

Table 5. Test Environments

5.3 Test Tools

Organisation	Tool
ETSI	HIVE TAP – Test Automation Platform
Canonical Tata Elxsi Whitestack	OSM Release NINE – 9.1 OSM to SOL006 Descriptor Translation with OSM Client. https://asciinema.org/a/fmA8NOKGA9rewOZCQlaua53WW
Vulk	CNCF CNF Conformance Test Tool. https://github.com/cncf/cnf-conformance/blob/v0.10.0/README-testsuite.md

Table 6. Test Tools

5.4 Open Source Communities

The Open Source communities listed below were actively involved in the Plugtests activities and their solutions were widely present in the Test Sessions through one or multiple distributions:

Project	Role	Details
AdvantEDGE	MEC Platform	https://github.com/InterDigitalInc/AdvantEDGE
CNCF Kubernetes	Test Environment	https://kubernetes.io/
ETSI Open Source MANO	MANO	https://osm.etsi.org
Open Stack	Test Environment	https://www.openstack.org
StarlingX	Test Environment	https://www.starlingx.io/

Table 7 Supporting Open Source communities

5.5 Observers

Organisation	Role
CAICT	Academia
DISMI	Academia
DOCOMO	Network Operator
NOS	Network Operator
NTT	Network Operator
Orange	Network Operator
STC	Network Operator
Telefonica	Network Operator

Table 8. Observers

5.6 Technical Support

The organisations below provided technical support and expertise to the Plugtests Team and contributed actively to prepare and run the Test Sessions during the Plugtests.

Organisation	Role
Baron	Technical Support
Nextworks	Technical Support
Sismondi	Technical Support

Table 9. Technical Support

6 Test Infrastructure

6.1 HIVE

The remote integration, pre-testing and test sessions were fully enabled by the NFV Plugtests Programme's HIVE network

Figure 3. NFV Plugtests HIVE network

The NFV HIVE (Hub for Interoperability and Validation at ETSI) network interconnects securely participants' remote labs and Functions under Test and allows for remote multi-party interoperability testing and validation activities. A total of 16 remote locations including several OSM Remote Labs participating to the Plugtests leveraged the HIVE network to make their Functions Under Test and Test Environments available for the test sessions.

As shown in the figure below, all the elements and actors in the Test Sessions: Functions Under Test, Test Environments, Test System (HIVE TAP), Participants and Plugtests Team experts were interconnected remotely through the HIVE network and several collaborative tools: WIKI, Slack, G2M ..

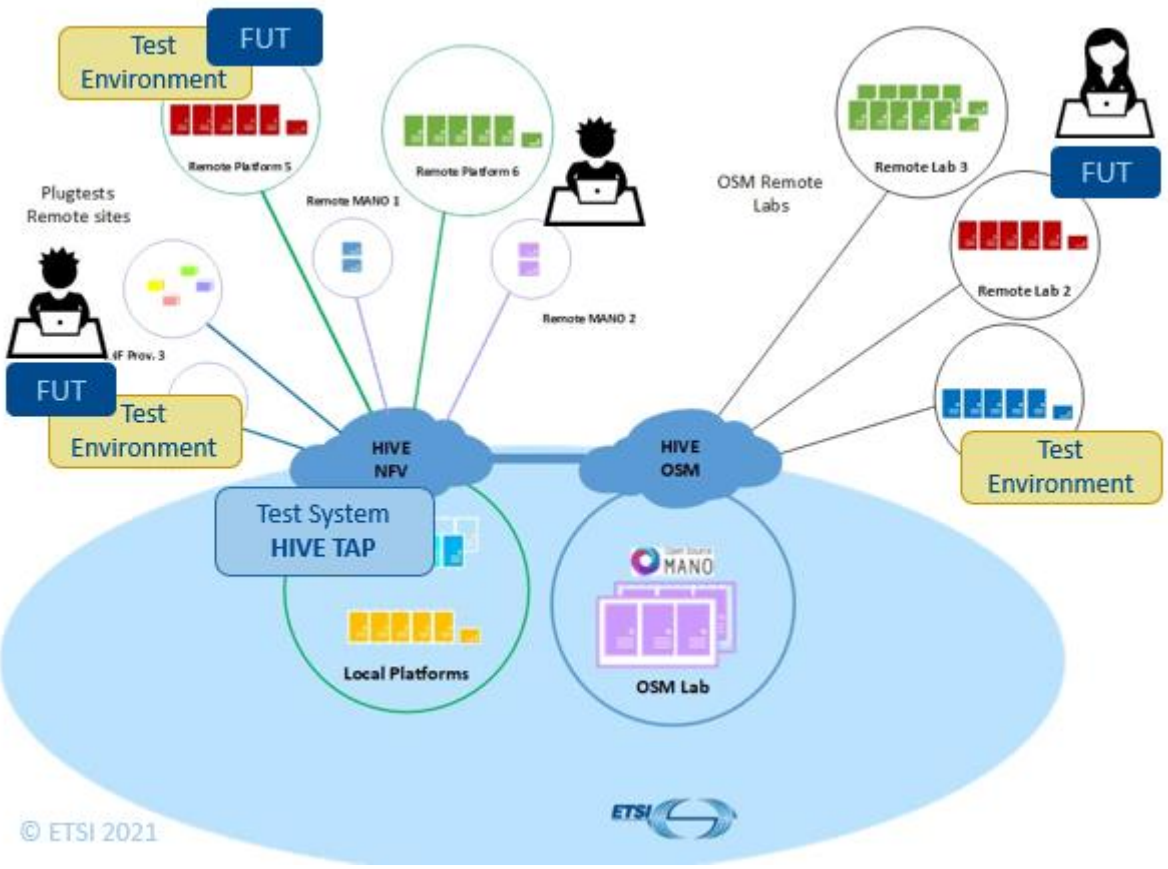

Figure 4. Remote Test Infrastructure

6.2 Test Automation Platform

The API Conformance Test Sessions relied on a Test System acting as API consumer and Notification Endpoint for the NFV and MEC APIs exposed by NFV and MEC components over different reference points of the respective architectural frameworks. The capabilities offered by the test system were:

- Sending configurable HTTP(S) requests
- Allowing custom payloads to be exchanged
- Uploading custom YAML, JSON and ZIP files to be used as request payloads (when applicable)
- Automatically applying headers validation on the response payloads
- Automatically applying schema validation on the response payloads
- Receiving and validating notifications

The test system was deployed as a set of Testing Tools in the HIVE Test Automation Platform (TAP), able to run the Robot Framework developed for [NFV-TST010] and [MEC032]. The Platform orchestrates test session executions in all the required steps, including:

- Selection of the appropriate Test Tool based on the API Under Test, base specification, and its version
- Configuration of the test system w.r.t Implementation Conformance Statements and implementation details,
- Test selection and execution, with detailed interaction with the user, and
- Test reporting and logging, to enable results collection and issues resolution.

The execution of the tests was triggered on demand by the participants, in a self-service fashion. A demo and a detailed presentation of the new platform was provided to participants during the preparation of the event.

The HIVE Test Automation Platform acts as a generic test orchestrator, able to transparently execute Test Suites implemented with different frameworks and tailored for different technologies. Ad-hoc components have been developed for NFV and MEC conformance testing, which can be reused in future activities and will be actively developed.

The Testing Tools that were made available for the event where developed on the bases of the Tools already available for the NFV&MEC Plugtests 2020, with several new features available:

- Execution of connectivity pre-checks, to validate the correct networking between the HIVE TAP and the FUT
- A more precise and user-friendly process for selection of individual test groups and test cases
- Possibility to commit or discard results
- Support for descriptors and packages upload

The usage of the HIVE TAP (with respect to the execution of sessions manually operated by a member of the Plugtests Team), led to a higher automation of the test execution which enabled a deeper learning of the Test Suites and the workflow of their execution. The learnings will be contributed back to ISG NFV and MEC (respectively in the TST and DECODE WGs) and a set of fixes has been made available in the code base during the event itself.

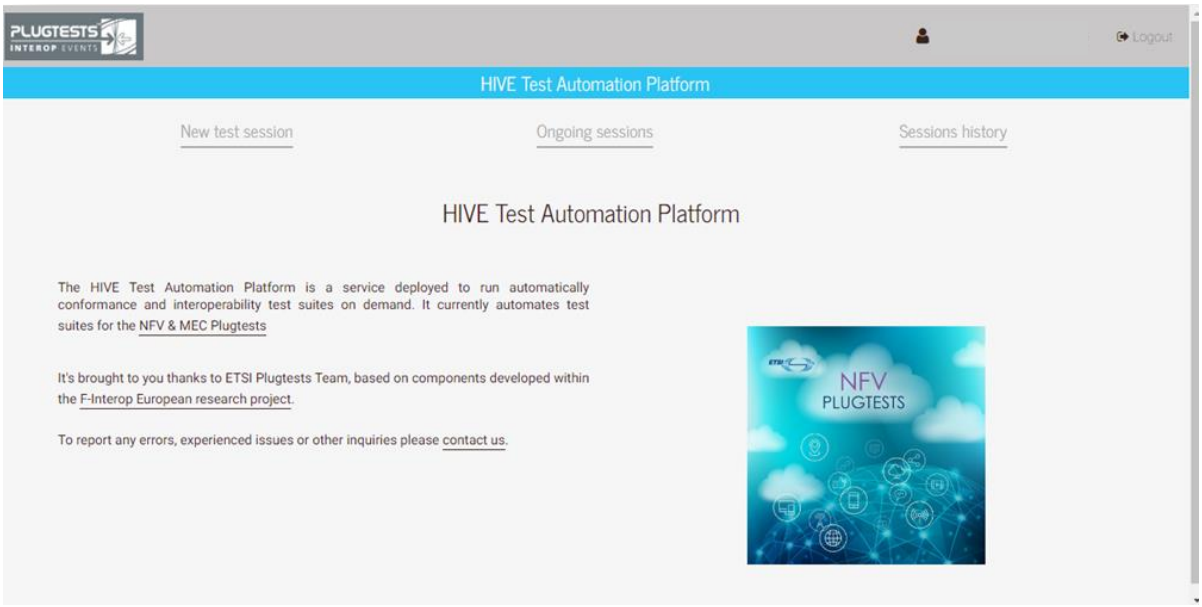

Figure 5. HIVE TAP - Test Automation Platform

7 API Testing Procedure

The NFV and MEC API Conformance Test Sessions aimed at validating the conformance of the participants FUTs to the [NFV-SOL002], [NFV-SOL003], [NFV-SOL005], [MEC012] and [MEC013] API specifications, while validating the API Conformance Robot Test Suites. The Test System was run on the HIVE TAP, which provided the connectivity to the participating FUTs. The Test System executed the Robot Framework Test Suites developed for the NFV API Conformance Test Specification [NFV-TST010] - in the published 2.4.1 and 2.6.1 versions and in the stable draft of version 2.7.1 - and made available via the ETSI Forge [NFV-ROBOT-TS]. For MEC, the Test System executed the Robot Framework Test Suites [MEC-ROBOT-TS] specified in [MEC-DEC032].

Each test session was executed on-demand and in a self-service fashion, to allow maximum flexibility and scalability of execution resources (e.g. with regards to time zone differences and number of parallel sessions).

Within each test session, the user would – via the workflow implemented in HIVE TAP – execute the following steps:

1. Log into the HIVE TAP, with credentials created for each participating team.

2. Select the API (i.e. NFV or MEC Interface) to be tested, e.g. NS Lifecycle Management over [NFV-SOL005].

3. Fill in the configuration of the test system, i.e. providing values for the variables defined in the Robot resource files. The variables were automatically collected from the Robot test suite and presented in a form for the user, who could fill them in individually or as a JSON data structure (enabling reuse of configuration settings).

4. After activation of the test session (which comprised initialization of a dedicated test environment and instantiation of the specifically configured test system), the user could select, execute, and skip individual groups of tests within the Test Suite for the selected API. Groups were defined by the individual Robot files in the test suite.

5. After the execution of all tests in the Robot File, the user was presented with the possibility to download the detailed test reports as produced by the Robot executor, in both human readable (HTML) and machine readable (XML) formats. The user was also given the possibility to commit the results for the calculation of aggregated statistics or to discard the results.

6. After execution of all Robot files, the user was presented with the possibility to restart or terminate the test sessions.

Committed results where automatically collected by the platform to allow the generation of aggregated and individual statistics.

8 Test Plans Overview

8.1 NFV API Conformance

This NFV API Conformance test plan was based on the Robot Framework Test Cases developed for [NFV-TST010] NFV API Conformance Test Specification, addressing FUT API Conformance to [NFV-SOL002], [NFV-SOL003] and [NFV-SOL005] specifications. In particular, for this Plugtests, three NFV API Conformance test specifications versions were made available to the participants for their tests:

- [NFV-TST010] v2.4.1, with NFV API conformance tests for [NFV-SOL002], [NFV-SOL003] and [NFV-SOL005] v2.4.1
- [NFV-TST010] v2.6.1 with NFV API conformance tests for [NFV-SOL002], [NFV-SOL003] and [NFV-SOL005] v2.6.1
- Stable version of [NFV-TST010] v2.7.1, with preliminary NFV API conformance tests for [NFV-SOL002], [NFV-SOL003] and [NFV-SOL005] v2.7.1

The Robot Framework test system acted as consumer for the NFV APIs produced by the FUTs, thus focusing only on testing the server-side of the NFV APIs under Test.

The following clauses summarise the test cases in scope for this Plugtests, grouped by FUT type. As none of the participants bringing NFs (as detailed in Table 1) were providing support or implementation of the [NFV-SOL002] APIs exposed by the VNFs (i.e. VNF Configuration and VNF Indicator APIs), the next clauses refer to VNFM and NFVO FUT types only.

The complete Test Specifications can be found in the [NFV-TST010] documents and the associated Robot Test Cases are available in the ETSI Forge [NFV-ROBOT-TS].

- V2.4.1 https://forge.etsi.org/rep/nfv/api-tests/tree/2.4.1
- V2.6.1 https://forge.etsi.org/rep/nfv/api-tests/tree/2.6.1
- V2.7.1 https://forge.etsi.org/rep/nfv/api-tests/tree/2.7.1-dev

8.1.1 VNFM

The VNFM APIs were tested following the test configuration shown in the figure below. In particular, two set of APIs were in scope for this NFV&MEC API Plugtests:

- NFV-SOL002 APIs, exposed by the VNFM and consumed by the test system (HIVE TAP) acting as VNF/EM
- NFV-SOL003 APIs, exposed by the VNFM and consumed by the test system (HIVE TAP) acting as NFVO

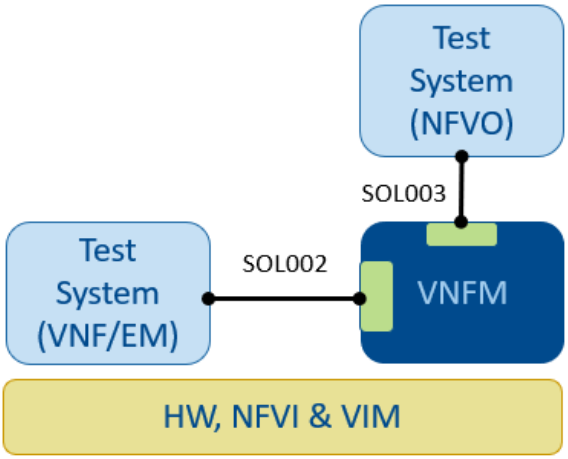

Figure 6: VNFM APIs Test Configuration

8.1.1.1 NFV-SOL002

The following subset of the [NFV-TST010] v2.4.1 and v2.6.1 Test Suites for [NFV-SOL002] APIs exposed by VNFMs was run during this Plugtests.

VNFM SOL002 API	Version	[NFV-TST010] Clause
VNF Life Cycle Management API	V2.4.1, v2.6.1	6.3.5 (Annex E for v2.6.1)

Table 10. VNFM SOL002 API tests suites

8.1.1.2 NFV-SOL003

The following subset of the [NFV-TST010] v2.4.1 and v2.6.1 Test Suites for [NFV-SOL003] APIs exposed by VNFMs was run during this Plugtests.

VNFM SOL003 API	Version	[NFV-TST010] Clause
VNF Life Cycle Management API	V2.4.1, v2.6.1	7.3.1 (Annex F for v2.6.1)
VNF Performance Management API	v2.6.1	7.3.4 (Annex F)

Table 11. VNFM SOL003 API tests suites

8.2.3 NFVO

The NFVO APIs were tested following the test configuration shown in the figure below. In particular, two set of APIs were in scope for this Plugtests:

- NFV-SOL003 APIs, exposed by the NFVO and consumed by the Robot Framework test system acting as VNFM
- NFV-SOL005 APIs, exposed by the NFVO and consumed by the Robot Framework test system acting as OSS/BSS

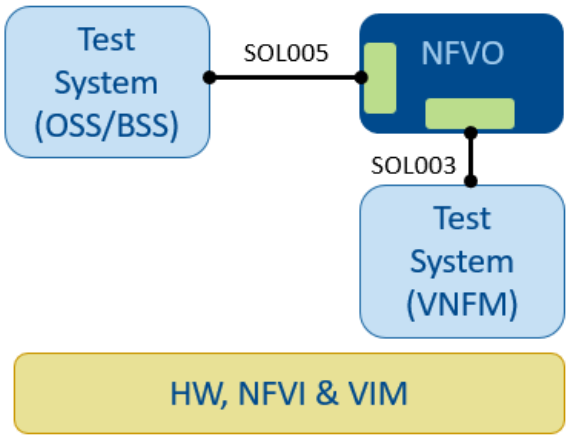

Figure 7: NFVO APIs Test ConfigurationSOL003

8.2.3.1 NFV-SOL003

The following subset of the [NFV-TST010] v2.4.1 and v2.6.1 Test Suites for [NFV-SOL003] APIs exposed by NFVOs was run during this Plugtests.

NFVO SOL003 API	Version	[NFV-TST010] Clause
VNF Package Management API	V2.4.1, v2.6.1	7.3.3 (Annex F for v2.6.1)
VNF Lifecycle Operation Granting API	v2.6.1	7.3.2 (Annex F)
Virtualised Resource Quota Available Notification API	v2.6.1	7.3.7 (Annex F)

Table 12. NFVO SOL003 API tests suites

8.2.3.2 NFV-SOL005

The following subset of the [NFV-TST010] v2.4.1, v2.6.1 and 2.7.1 Test Suites for [NFV-SOL005] APIs exposed by NFVOs was run during this Plugtests.

NFVO SOL005 API	Version	[NFV-TST010] Clause
NSD Management API	V2.4.1, v2.6.1, 2.7.1	5.3.1 (Annex D for v2.6.1 and v2.7.1)
NS Lifecycle Management API	V2.4.1, v2.6.1, 2.7.1	5.3.2 (Annex D for v2.6.1 and v2.7.1)
NS Fault Management API	v2.6.1	5.3.3 (Annex D)
NS Performance Management	v2.7.1	5.3.4 (Annex D)
VNF Package Management API	v2.6.1, 2.7.1	5.3.5 (Annex D)

Table 13. NFVO SOL005 API tests suites

8.2 MEC API Conformance

The test plan for the MEC API Conformance testing was based on [MEC-DEC032].

This Test Specification provides a database of test purposes for MEC APIs and implementation in TTCN-3 and Robot Framework as Abstract Test Suites. The Robot test suites were available for Plugtests participants to be executed over the HIVE TAP.

Based on the capabilities and selections of the participating FUTs, the API Tests prepared and executed during this Plugtests targeted [MEC012] and [MEC013] specifications. The full test suites are available at [MEC-ROBOT-TS].

The test suite structure followed [MEC-DEC032], Clause 5.

8.2.1 MEC012

The following subset of the [MEC-DEC032] Test Suites for [MEC012] APIs were run during the this Plugtests

MEC012 API	Version	[MEC-DEC032] Clause
RNIS API	V2.1.1	6.4.7

Table 14. MEC012 API tests suites

8.2.2 MEC013

The following subset of the [MEC-DEC032] Test Suites for [MEC013] APIs was run during the this Plugtests

MEC013 API	Version	[MEC-DEC032] Clause
RLOCLOOK API	V2.1.1	6.4.7
UEDISTLOOK API	V2.1.1	6.4.14
UEINFLOOK API	V2.1.1	6.4.16
UEINFSUB API	V2.1.1	6.4.17
UELOCSUB API	V2.1.1	6.4.19

Table 15. MEC013API tests suites

9 Results

9.1 NFV API Conformance Results

During the Remote NFV&MEC API Plugtests 2021, a total of 22 NFV Test Suites were executed among the different test sessions. The API conformance tests were executed for three different Functions Under Test (FUTs): VNFs, VNFMs and NFVOs. With respect to the previous event, tests for [NFV-SOL002] were executed, as well as for [NFV-SOL003] and [NFV-SOL005].

A total of 14 FUTs participated to the Remote API Plugtests, distributed in this way:

- 4 VNFMs testing [NFV-SOL002] and [NFV-SOL003] APIs
- 10 NFVOs testing [NFV-SOL003] and [NFV-SOL005] APIs

To facilitate the analysis, results are presented as follows:

Result	Meaning
PASS	Test Case run. Test Purpose successfully achieved.
FAIL	Test Case run. Test Purpose not achieved.
Total	Total number of Test Cases Run = PASS + FAIL

Table 16: Results Interpretation

Note that the tests cases for which no result was reported (i.e. when the test session run out of time) are not considered in the Total Results. Moreover, for test executions which required a re-run (i.e. to make some fixes in the Test Suite or in the FUT) only the best results were kept.

The table below provides the overall results (aggregated data) for all the NFV API Conformance tests run during the NFV&MEC Plugtests 2021, from all participating organisations.

Overall NFV Results	NFV API Conformance		Totals
	PASS	FAIL	Run
	611	585	1196

Table 17: NFV API Conformance Overall Results

For each remote Test Session, depending on the involved FUT and the features to be tested, the involved participant was able to select different number of test cases.

Overall, the test plan included more than 2000 NFV API Conformance test cases, organised in different groups as described in clause 8.2. The test plan was based on [NFV-TST010] versions 2.4.1, 2.6.1 and 2.7.1. Participants were free to select the version of the test suite according to their implementations.

With respect to the previous Plugtests event, a larger number of tests was made available. Through the Test Sessions run, a total of 1196 Test Results were executed and reported.

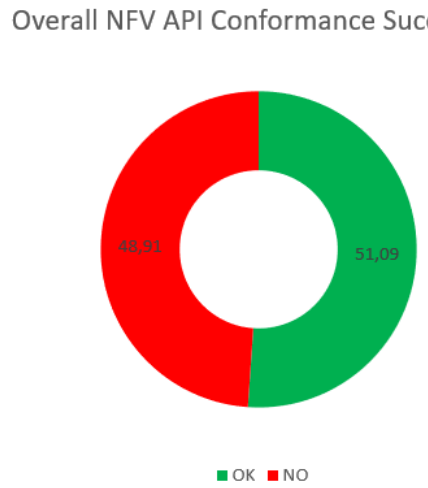

Figure 8. NFV API Conformance Overall Results (%)

The next clauses present more detailed results grouped by the base specifications (i.e. NFV-SOL Specifications) and their different versions, by FUT type and by test group and will allow to identify the areas and APIs with higher execution and conformance rates.

9.1.1 Results per NFV Specification

The tables and figures below provide an overview of the results for the NFV API Conformance per SOL specification. Overall, the [NFV-SOL005] APIs have been those with the higher number of Test Cases run while [NFV-SOL003] had the highest success rate.

	API Conformance		Totals	API Conformance (%)	
	PASS	FAIL	Results	% PASS	% FAIL
NFV-SOL002	8	8	16	50,0%	50,0%
NFV-SOL003	261	207	468	55,8%	44,2%
NFV-SOL005	342	370	712	48,0%	52,0%
TOTAL	611	585	1196	51,1%	48,9%

Table 18: Test Results Summary per NFV Specification

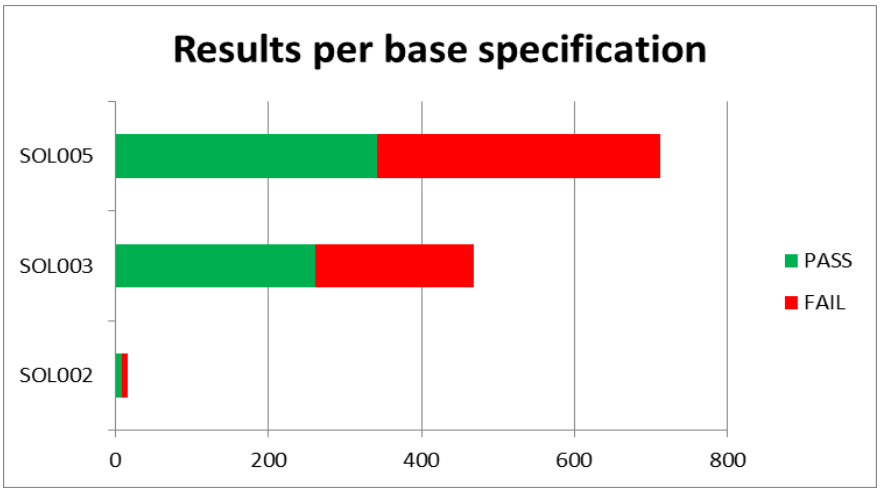

Figure 9. Test results per NFV Specification

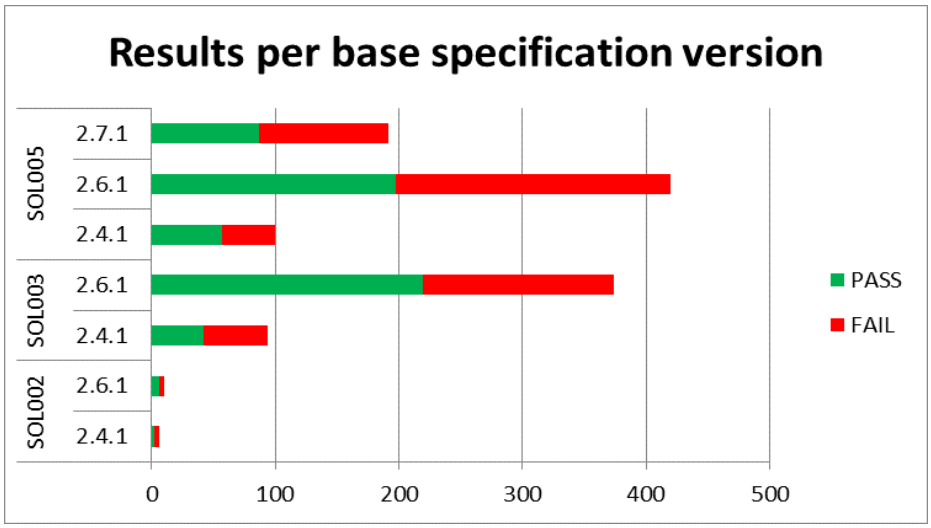

Figure 10. Test results per NFV Specification version

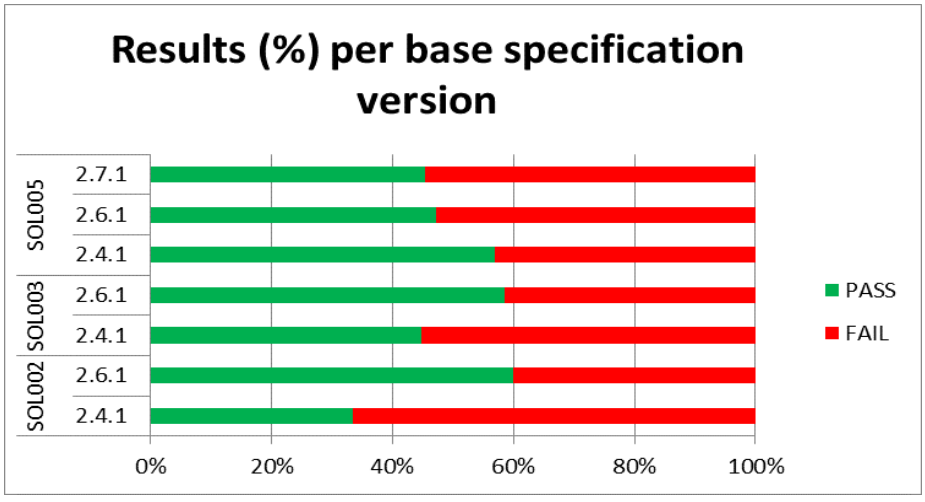

Figure 11. Test results per NFV Specification version - %

9.1.2 Results per NFV FUT Type

The tables and figures below summarize the results for the NFV API conformance tests per type of FUT involved in the test sessions, i.e. VNFM and NFVO. NFVO FUTs were those most tested.

	API Conformance		Totals	API Conformance (%)	
	PASS	FAIL	Run	% PASS	% FAIL
VNFM	114	121	235	48,5%	51,5%
NFVO	497	464	961	51,7%	48,3%
TOTAL	611	585	1196	51,0%	49,0%

Table 18: Test Results Summary per-FUT type

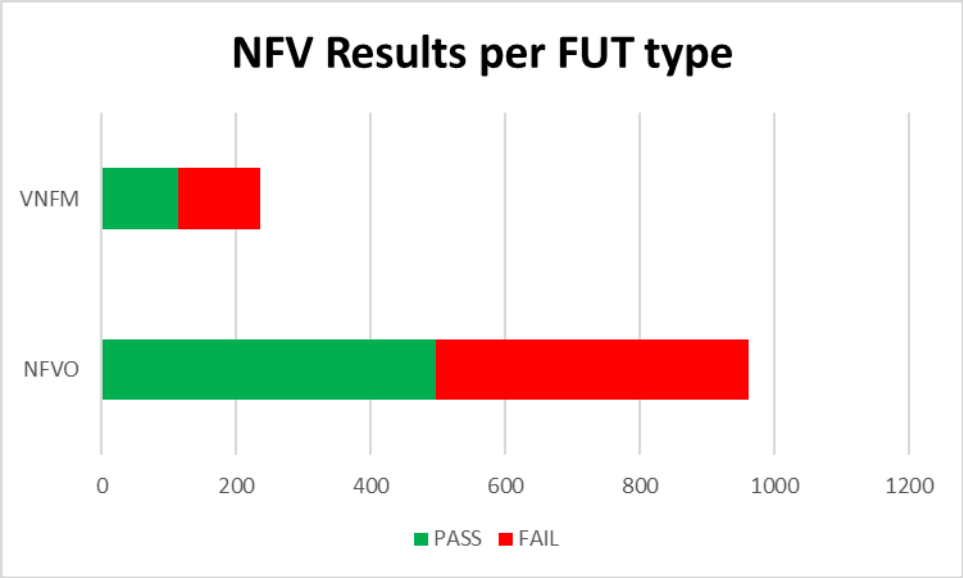

Figure 12. Test Results per-FUT type – Totals

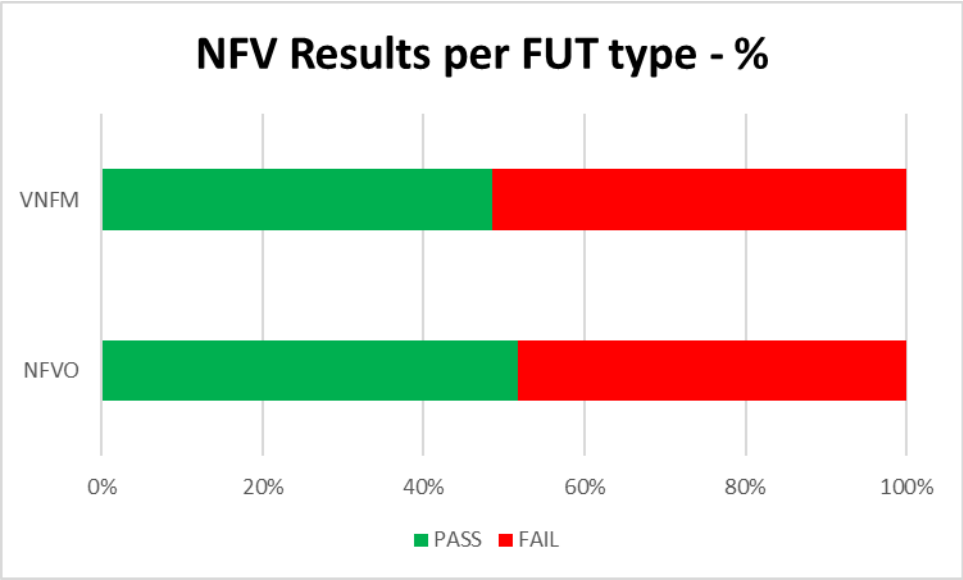

Figure 13. Test Results per- FUT type - %

9.1.3 Results per NFV API

The following clauses provide tables and figures which summarize the results for the NFV API conformance tests for the different APIs in each test configuration.

9.1.3.1 VNFM – NFV-SOL002

	API Conformance		Totals	API Conformance (%)	
	PASS	FAIL	Run	% PASS	% FAIL
VNF Lifecycle Management API	8	8	16	50%	50%
TOTAL	8	8	16	50%	50%

Table 19: VNFM NFV-SOL002 test results summary

9.1.3.2 VNFM – NFV-SOL003

	API Conformance		Totals	API Conformance (%)	
	PASS	FAIL	Run	% PASS	% FAIL
VNF Lifecycle Management API	72	77	149	48,3%	51,7%
VNF Performance Management API	34	36	70	48,6%	51,4%
TOTAL	106	113	219	48,4%	51,6%

Table 20: VNFM NFV-SOL003 test results summary

9.1.3.3 NFVO – NFV-SOL003

	API Conformance		Totals	API Conformance (%)	
	PASS	FAIL	Run	% PASS	% FAIL
Virtualised Resources Quota Available Notification API	20	6	26	76,9	23,1
VNF Package Management API	117	83	200	58,5%	41,5%
VNF Lifecycle Operation Granting API	18	5	23	78,3%	21,7%
TOTAL	155	94	249	62,2%	37,8%

Table 21: NFVO NFV-SOL003 test results summary

9.1.3.4 NFVO – NFV-SOL005

	API Conformance		Totals	API Conformance (%)	
	PASS	FAIL	Run	% PASS	% FAIL
NSD Management API	159	158	317	50,2%	49,8%
NS Fault Management API	4	17	21	19,0%	81,0%
NS Lifecycle Management API	61	56	117	52,1%	47,9%
VNF Package Management API	110	137	247	44,5%	55,5%
NS Performance Management API	8	2	10	80,0%	20,0%
TOTAL	342	370	712	48,0%	52,0%

Table 22: NFVO NFV-SOL005 test results summary

9.1.4 Results per NFV API Test Case

The full list of NFV API Conformance results per Test Case is provided in [NFV-API2021-TR].

9.2 MEC API Conformance Results

During the this Plugtests, a total of 7 MEC Test Suites were executed among the different test sessions. The API Conformance tests were executed for Functions Under Test (FUTs) of type “MEC Platform”. A total of 2 MEC Platforms participated to the MEC API Conformance sessions.

The table below provides the overall results (aggregated data) for all the MEC API Conformance tests run by all participating organisations.

Overall MEC Results	MEC API Conformance		Totals
	OK	NO	Run
	30	7	37

Table 23: MEC API Conformance overall results

For each remote Test Session, depending on the involved FUT and the features to be tested, the involved participant was able to select different number of test cases.

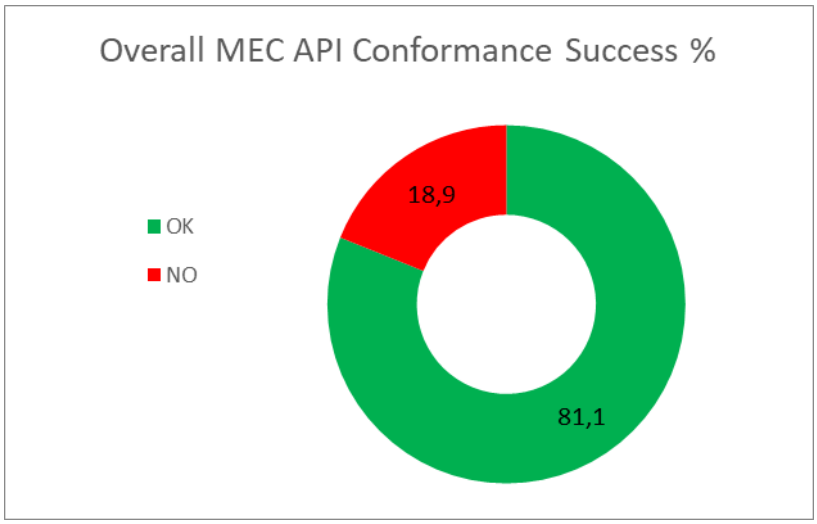

Figure 14. MEC API Conformance Overall results (%)

The next clauses present more detailed results per MEC Specification and per test group and will allow to identify the areas and APIs with higher execution and conformance rates.

9.2.1 Results per MEC Specification

The tables and figures below provide an overview of the results for the API conformance per MEC specifications, i.e. MEC012 and MEC013.

	API Conformance		Totals	Totals	
	OK	NO	Results	% OK	% NO
MEC012	16	6	22	72,7%	27,7%
MEC013	14	1	15	93,3%	6,7%
TOTAL	30	7	37	81,1%	18,9%

Table 24: Test Results summary per-MEC Specification

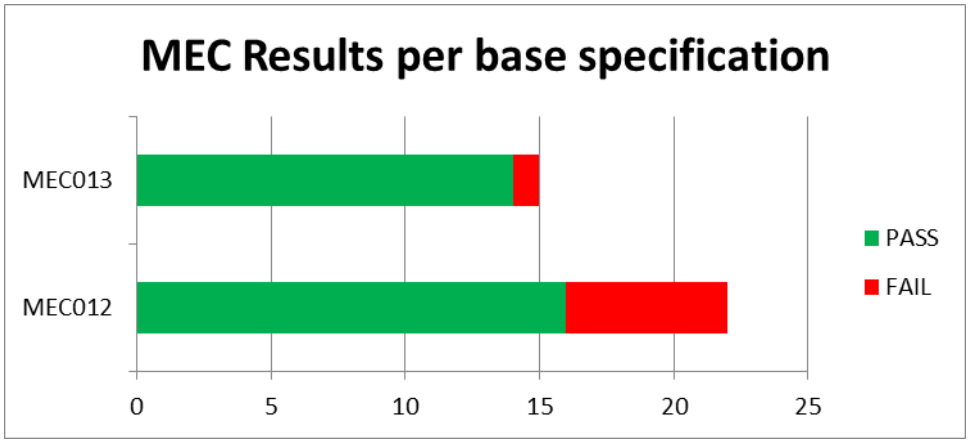

Figure 15. Test results MEC Specification – total

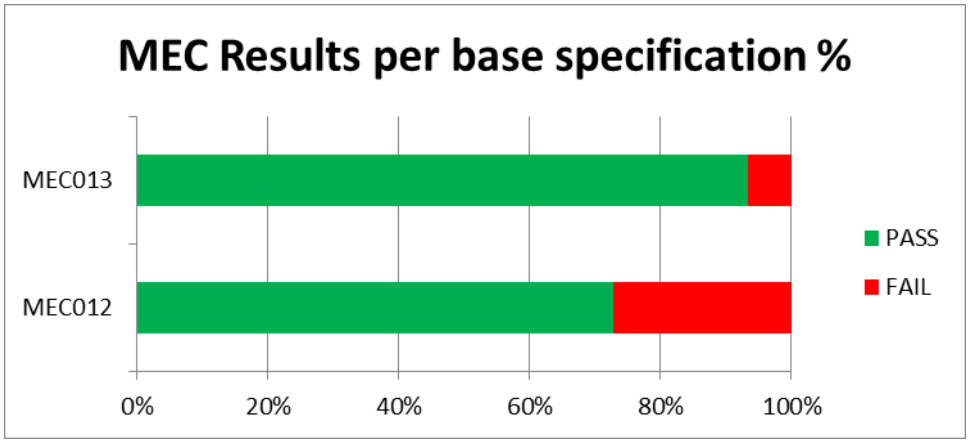

Figure 16. Test results per-MEC Specification - %

9.2.2 Results per MEC API Test Case

9.2.2.1 MEC012

API	GROUP	TESTNAME	PASS	FAIL	TOT
RNIS	RnisNotifications	Cell change notification	0	1	1
RNIS	RnisNotifications	RAB Establishment notification	0	1	1
RNIS	RnisNotifications	RAB modification notification	0	1	1
RNIS	RnisQuery BI BO	Request L2Meas info using non existing cell id	1	0	1
RNIS	RnisQuery BI BO	Request L2Meas info using wrong parameters	1	0	1
RNIS	RnisQuery BI BO	Request Plmn info using non existing application id	1	0	1
RNIS	RnisQuery BI BO	Request Plmn info using wrong parameters	1	0	1
RNIS	RnisQuery BI BO	Request RabInfo info using non existing cell id	1	0	1
RNIS	RnisQuery BI BO	Request RabInfo info using wrong parameters	1	0	1
RNIS	RnisQuery BI BO	Request S1Bearer info using non existing cell id	0	1	1
RNIS	RnisQuery BI BO	Request S1Bearer info using wrong parameters	0	1	1
RNIS	RnisQuery BV	Request L2Meas info	1	0	1
RNIS	RnisQuery BV	Request Plmn info	1	0	1
RNIS	RnisQuery BV	Request RabInfo info	1	0	1
RNIS	RnisQuery BV	Request S1Bearer info	0	1	1
RNIS	RnisSubscriptions BI BO	Create RNIS subscription using bad parameters	1	0	1
RNIS	RnisSubscriptions BI BO	Request RNIS subscription list using bad parameters	1	0	1
RNIS	RnisSubscriptions BV	Create RNIS subscription	1	0	1
RNIS	RnisSubscriptions BV	Get an Individual RNIS subscription	1	0	1
RNIS	RnisSubscriptions BV	Remove an Individual RNIS subscription	1	0	1
RNIS	RnisSubscriptions BV	Request RNIS subscription list	1	0	1
RNIS	RnisSubscriptions BV	Update an Individual RNIS subscription	1	0	1

Table 25: Test Results summary per-MEC Test Case (MEC012)

9.2.2.2 MEC013

API	GROUP	TESTNAME	PASS	FAIL	TOT
RLOCLOOK	PlatRadioNodeLocation	TC_MEC_SRV_RLOCLOOK_001_NF	1	0	1
RLOCLOOK	PlatRadioNodeLocation	TC_MEC_SRV_RLOCLOOK_001_OK	1	0	1
UEDISTLOOK	PlatUeDistanceLookup	TC_MEC_SRV_UEDISTLOOK_001_BR	1	0	1
UEDISTLOOK	PlatUeDistanceLookup	TC_MEC_SRV_UEDISTLOOK_001_OK	1	0	1
UEINFOLOOK	PlatUeInformationLookup	TC_MEC_SRV_UEINFOLOOK_001_BR	1	0	1
UEINFOLOOK	PlatUeInformationLookup	TC_MEC_SRV_UEINFOLOOK_001_NF	1	0	1
UEINFOLOOK	PlatUeInformationLookup	TC_MEC_SRV_UEINFOLOOK_001_OK	1	0	1
UEINFOSUB	PlatUeInformationSubscription	TC_MEC_SRV_UEINFOSUB_001_BR	1	0	1
UEINFOSUB	PlatUeInformationSubscription	TC_MEC_SRV_UEINFOSUB_001_OK	1	0	1
UEINFOSUB	PlatUeInformationSubscription	TC_MEC_SRV_UEINFOSUB_002_NF	1	0	1
UEINFOSUB	PlatUeInformationSubscription	TC_MEC_SRV_UEINFOSUB_002_OK	1	0	1
UELOCSUB	PlatUeLocationSubscription	TC_MEC_SRV_UELOCSUB_001_BR	0	1	1
UELOCSUB	PlatUeLocationSubscription	TC_MEC_SRV_UELOCSUB_001_OK	1	0	1
UELOCSUB	PlatUeLocationSubscription	TC_MEC_SRV_UELOCSUB_002_NF	1	0	1
UELOCSUB	PlatUeLocationSubscription	TC_MEC_SRV_UELOCSUB_002_OK	1	0	1

Table 26: Test Results summary per-MEC Test Case (MEC013)

10 Plugtests Outcome

During this Plugtests over 70 items were identified as potential issues and discussed with the participating community. This chapter compiles the outcome of these discussions, identifies some bugs, and provides some recommendations on NFV and MEC Specifications and the Robot Test Suites developed for the API Conformance Test Specifications.

10.1 Feedback on NFV Specifications

10.1.1 Potential inconsistency on disk and container format

A 33etrieved inconsistency among the NFV-SOL specifications has been identified related to the definition of disk and container format permitted values in the VnfPackageSoftwareImageInfo data type in [NFV-SOL003] and [NFV-SOL005]. In particular, [NFV-SOL005] in Table 9.5.3.2-1 (and the same in [NFV-SOL003] Table 10.5.3.2-1) specifies the permitted values disk and container format attributes as all uppercase, as can be seen in the following extract:

Container format indicates whether the software image is in a file format that also contains metadata about the actual software. Permitted values:

- *AKI: a kernel image format*
- *AMI: a machine image format*
- *ARI: a ramdisk image format*
- *BARE: the image does not have a container or metadata envelope*
- *DOCKER: docker container format*
- *OVA: OVF package in a tarfile*
- *OVF: OVF container format*

and

Disk format of a software image is the format of the underlying disk image. Permitted values:

- *AKI: a kernel image format*
- *AMI: a machine image format*
- *ARI: a ramdisk image format*
- *ISO: an archive format for the data contents of an optical disc, such as CD-ROM*
- *QCOW2: a common disk image format, which can expand dynamically and supports copy on write*
- *RAW: an unstructured disk image format*
- *VDI: a common disk image format*
- *VHD: a common disk image format*
- *VHDX: enhanced version of VHD format*
- *VMDK: a common disk image format*

However, two notes in the [NFV-SOL005] Table 9.5.3.2-1 (and the same in [NFV-SOL003] Table 10.5.3.2-1) state that the list of permitted values was taken from “Container formats” and “Disk Formats” defined by Openstack in:

https://docs.openstack.org/glance/pike/user/formats.html

In these Openstack reference definitions, the disk and container format permitted values are defined as all lowercase, so not aligned with [NFV-SOL005] Table 9.5.3.2-1 and [NFV-SOL003] Table 10.5.3.2-1 specifications.

While the Openstack reference is defined as Informative Reference in both [NFV-SOL005] and [NFV-SOL003], and therefore "*not necessary for the application of the present document, but they assist the user with regard to a particular subject area",* the following [NFV-SOL001] and [NFV-SOL006] VNFD type files:

- https://forge.etsi.org/rep/nfv/SOL001/raw/v2.6.1/etsi_nfv_sol001_vnfd_types.yaml
- https://forge.etsi.org/rep/nfv/SOL006/blob/v2.6.1/src/yang/etsi-nfv-vnf.yang

also include all lowercase definitions of the disk and container format permitted values.

34etrieved it is not clear if the API specifications in [NFV-SOL005] and [NFV-SOL003] should be aligned with the VNFD data types defined in [NFV-SOL001] and [NFV-SOL006].

Issue 8008 has been reported to NFV SOL Working Group to notify this potential 34etrieved34ncy.

10.1.2 Usability issue on hardcoded authorization header name

A usability issue with authentication and authorization in the [NFV-ROBOT-TS] has been reported. The issue concerned the lack of flexibility in the usage of custom HTTP header name to transmit the authentication tokens for API requests.

In details, the [NFV-ROBOT-TS] included the name of such HTTP header as hard-coded into the low-level code of the tests. As a result, implementations of the NFV APIs that made use of a different header name were not able to successfully execute any of the available tests. This has been reported as a blocking point for some communities avoiding them to run the NFV API Conformance tests at all.

From the specification perspective, the [NFV-SOL013] specifies the header name for authorization to be required, this being the name "Authorization". The very same name, in compliance with the base specification, was the one used in the low level code. Therefore, this raised issue was targeting an improvement in flexibility and usability of the Test Suites, and not the correctness of authentication mechanisms to be used.

Following guidance from the Plugtests Team, the issue has been therefore reported by a participant to NFV-TST WG as contribution NFVTST(21)000013. As a result of the discussion, the standardization group agreed to let the Plugtests Team apply a change to the executable test suites in order to allow users to configure such a parameter.

This modification would allow implementations to execute the functional tests and to verify the correctness of the NFV APIs behaviour. Nonetheless, the usage of an Authentication HTTP Header name different from the one specified in [NFV-SOL013] (i.e. "Authorization") is to be considered not conformant with the NFV API specifications.

The possibility to allow customization of such parameter is granted as an exception, given that it addresses a non-functional requirement (i.e. security mechanisms). The same is in fact applied by allowing the users to disable authentication mechanisms as a whole, as part of the parametrization of the test suites.

Documentation on the repercussion on the conformance to the NFV APIs will be added in [NFV-TST010] clarifying the aforementioned requirement: custominzing the authorization header name results in voiding conformance claims.

Therefore, in order to achieve conformance to the specifications, implementers of the NFV API shall work to allow their products and implementation to utilize the standard authorization header name (i.e. "Authorization").

10.1.3 Feedback on the NFV Robot Test Suite

Overall, the NFV&MEC API Plugtests allowed to identify and file 48 issues on the Robot Test Suites associated to [NFV-TST010] in the three available versions (v2.4.1, v2.6.1 and preliminary v2.7.1). The table below summarises all the issues and indicates the identifier of the issue in the ETSI Forge [NFV-ISSUE-TR] together with the affected SOL APIs.

Issue ID	Description	SOL002	SOL003	SOL005
183	"SOL002/vnflcm/{apiMajorVersion}/subscriptions/${subscriptionId}} API V2.4.1 GET method"	X		

182	“SOL003/vnflcm/{apiMajorVersion}/subscriptions API V2.4.1 GET method”		X	
181	“SOL002/vnflcm/{apiMajorVersion}/subscriptions API V2.4.1 POST and GET method”	X		
180	“SOL003 and SOL002 /vnf_lcm_op_occs/${vnfLcmOpOccId}/failAPI V2.4.1 POST method”	X	X	
179	“SOL002/vnflcm/{apiMajorVersion}/vnf_lcm_op_occs/${vnfLcmOpOccId} API V2.4.1 GET method”	X		
177	“SOL003/vnflcm/{apiMajorVersion}/vnf_lcm_op_occs/${vnfLcmOpOccId} API V2.4.1 GET method”		X	
176	“SOL002/vnflcm/{apiMajorVersion}/vnf_lcm_op_occs V2.4.1 GET method”	X		
175	“SOL003 and SOL002 /vnflcm/{apiMajorVersion}/vnf_instances/${vnfInstanceId} GET V2.4.1”	X	X	
174	“SOL002/vnflcm/{apiMajorVersion}/vnf_lcm_op_occs/${vnfLcmOpOccId}/fail API POST V2.6.1”	X		
173	“SOL002 and SOL003 /vnflcm/{apiMajorVersion}/subscriptions/${subscriptionId} DELETE V2.6.1”	X	X	
172	“SOL002 /vnflcm/{apiMajorVersion}/subscriptions POST V2.6.1”	X		
171	“SOL002 /vnflcm/{apiMajorVersion}/vnf_lcm_op_occs, /vnflcm/{apiMajorVersion}/vnf_lcm_op_occs/${vnfLcmOpOccId}, /vnflcm/{apiMajorVersion}/subscriptions and /vnflcm/{apiMajorVersion}/subscriptions/${subscriptionId} GET V2.6.1”	X		
170	“SOL002 /vnflcm/{apiMajorVersion}/vnf_instances POST, /vnflcm/{apiMajorVersion}/vnf_instances GET and /vnflcm/{apiMajorVersion}/vnf_instances/${vnfInstanceId} GET V2.6.1”	X		
169	“SOL005 NS LCM Individual NSInstance v2.7.1 – delete with conflict cannot be run due to previous test execution”			X
168	“TST issue in ‘/vnflcm/{apiMajorVersion}/vnf_lcm_op_occs SOL003 v2.6.1 api”		X	
167	“POST Terminate a vnfInstance Conflict returned http code 202 SOL003 VNFLifecycleManagementAPI v2.6.1”		X	
166	“Wrong data type for links SOL005 VNFPackageManagementAPI v2.6.1”			X
165	“Using the same parameter when checking full size and Content-length SOL003 v2.6.1 VNFPackageManagementAPI”		X	
164	“SOL005 NSDManagement-API Subscriptions NFVO_DUPLICATION==0 is used in both test cases.”			X
163	“SOL005 NSDManagement IndividualSubscription PUT & PATCH methods fail to evaluate Json”			X
162	“SOL005 IndividualNSDescriptor nsdUsageState is 35etrieved on a 404”			X
161	“SOL002 / IndividualVnfLcmOperationOccurence / 2.6.1 Multiple problems with PATCH methods”	X		

160	“SOL002/VNFLifecycleManagement/IndividualVNFInstance/2.6.1 DELETE queries with Content-Type: application/merge-patch+json”	X		
159	“FileNotFoundError: No such file or directory: \”schema//Subscriptions.schema.json\” SOL003 VNFLifecycleMangagementAPI v2.6.1”		X	
158	“\”=\” should not be included in Content range SOL003/SOL005 VNFPackageManagementAPI V2.6.1”		X	X
157	“Unknown parameter value appeared when testing SOL003 VNFPackageManagementAPI V2.6.1”		X	
156	“SOL005 Subscription end point is not available for POST response.”			X
155	“Typo in SOL005 NSDManagement-API NSDManagementKeywords.robot”			X
154	“Automated generation of callbackUri notification endpoint during SOL002/3/5 subscription operations”	X	X	X
153	“SOL003/VNFPackageArtifacts/2.6.1 Range is kept all along the test cases”		X	
152	“Definition of disk and container format conflicted with Specification v2.6.1 SOL003&SOL005 VNFPackageManagementAPI”		X	X
151	“When ${NFVO_RANGE_OK} =1 , test case ‘XXX and NFVO not supporting Range Requests’ should be skipped SOL003 v2.6.1 VNFPackage ManagementAPI”		X	
150	“Missing Set Suite Variable in keyword ‘Check Postcondition VNF Package Artifact Exist’ v2.6.1 SOL003&SOL005 VNFPackageManagement-API”		X	X
149	“SOL005 NS Subscription END point is returning 404 for GET response.”			X
148	“JSON schemas containing references to external files”			X
147	“Wrong parameter vlalue in ‘Get all VNF Packages with malformed authorization token’ SOL005 VNFPackges.robot V2.6.1”			X
146	“Misplaced referenced json requests in VNFPackageManagementKeywords.robot”			X
144	“Wrong method in Keywords ‘GET Scale vnfInstance’ SOL003 VNFLifecycleManagement-API v2.6.1”		X	
143	“Wrong parameter name in ‘IndividualVnfLcmOperationOccurence.robot’ SOL003 VNFLifecycleManagement-API v2.6.1”		X	
142	“Separator argument missing 2.6.1”		X	
140	“Wrong Keywords in SOL003 VNFPackageManagement-API VNFPackageContent.robot”		X	
139	“Separator argument missing in SOL005 NSDManagement-API ApiVersion.robot”			X
138	“Separator argument missing in SOL003 VNFPackageManagement-API ApiVersion.robot”		X	
137	“Separator argument missing in SOL003 VNFFaultManagement-API ApiVersion.robot”		X	

136	"Non utf-8 charachters present in schema json SOL003 VNFLifecyManagement-API"		X	
135	"SOL003 / VNF LCM – conflicting operations on vnf instances"		X	
134	"Inconsistency in MockServer configuration parameters"		X	X
133	"The parameter ${vnfLcmOpOccId} is defined repetitiously in the variables.txt file and the configuration.txt file"		X	

Table 27: Issues on the NFV-TST010 Robot Test Suites

10.2 Feedback on MEC Specifications

10.2.1 MEC013 inconsistency on endpoint definition

An inconsistency has been reported endpoint definitions in [MEC013]. In clause 7.2 the graphical representation of the resource tree is shown, indicating that all queries endpoint should be exposed as subresources of the /queries resource. As an example, the query on users is depicted as the resource /queries/users.

In the subsequent sections the enpoints are specified not to be subresources for the /queries resource. As an example, in clause 7.3.2, the request URI is specified as /location/v2/users, does not including the /queries result.

The same applies to subsequent resource definitions.

The ISG is invited to publish a revision of [MEC013] that fixes the inconsistency. For the sake of the tests, the latest changes applied to the document has been used, i.e. including the /queries/ segment in the lookup resources. This was also in line with the definitions provided in the OpenAPIs at https://forge.etsi.org/rep/mec/gs013-location-api.

10.2.2 Non existent attribute in a json body should be ignored

This issue on the [MEC-ROBOT-TS] has been reported at https://forge.etsi.org/rep/mec/gs032p3-robot-test-suite/issues/38. The content of the issue is copied here for convenience to the readers.

When sending a request including a JSON body in which one of the required attributes is not defined, the current expectation from the test suite is to receive a response with code 400 ("Invalid Request"). It is suggested that a more appropriate action would be to ignore the unknown parameter. This allows a less strict approach. It was not found in the specifications what should be the behavior as a result of this situation and ISG MEC is asked for recommendations or to clarify the applicable specifications.

From the point of view of a MEC API Producer implementation, the following is generally applied:

1. When unmarshalling a JSON string into an object, it is possible to ignore the undefined fields. The question then becomes *should the service be resilient and process the request (ignoring unknown fields) or should the service be strict and report an error*.

2. In the same manner, when receiving a request with invalid query parameters, a service can easily ignore these and only use the known params to process the request. The question then becomes: should the service be resilient and process the request (ignoring unknown query parameters) or should the service be strict and report an error.

The proposers have not found the definitive answer in MEC Specifications whether a service should be resilient or strict in replying with an error condition.

This is a specification interpretation error and ISG MEC is asked to provide guidance on the level API implementation, w.r.t. strictness or permissivity. For completeness, examples behaviours of strict and permissive implementations is detailed below.

A strict implementation:

- should perform JSON/Query param validation against the schema;
- should refuse a request with a malformed JSON/Query param;
- should return 400 or 404 error codes.

A permissive implementation:

- should not perform validation against the JSON schema/Query param list;
- can accept a request with malformed JSON/Query param (just ignores the unknown fields);
- can return a 200 response if valid fields are sufficient.

This issue will be contributed to ISG MEC as part of the Report from the Plugtests activities.

10.2.3 Feedback on the MEC Robot Test Suite

Overall, the Remote NFV&MEC API Plugtests 2021 allowed to identify and file 18 issues on the Robot Test Suites for [MEC-DEC032] version 2.1.1. The table below summarises all the issues and indicates the identifier of the issue in the ETSI Forge [MEC-ISSUE-TR] together with the affected MEC APIs.

Issue ID	Description	MEC API
24	Handle optionality for ProblemDetails presence and value check	ALL
25	Templates for Notification requests are not resolved	ALL
26	Plugtests 2021: Schemas not defined in few MEC013 test suites	MEC013
27	Plugtest 2021: MEC013 Location API using v1.1.1 schemas and endpoints, not v2.1.1	MEC013
28	Plugtest 2021: MEC012 RNISQuery BV RabInfo test error	MEC012
29	Plugtest 2021: MEC012 RNIS RNISQuery BV L2Meas using wrong schema	MEC012
30	Plugtest 2021: MEC012 RnisSubscription BV test suites validation errors	MEC012
31	Plugtest 2021: MEC012 RNISNotifications HTTPConnectionPool host name concatenation with port error	MEC012
32	Plugtest 2021: MEC013: Schemas have non-valid characters preventing the validation	MEC013
33	Plugtest 2021: MEC013: UELOCLOOK using non valid endpoint	MEC013
34	Plugtest 2021: MEC013: ClientCorrelator error in UELOCSUB	MEC013
35	Plugtest 2021: Spec discussion : Validity of returning a list with empty parameters rather than code 404	ALL
36	Plugtest 2021: MEC012: RNISQuery BV L2Meas info should not be an array	MEC012
37	Plugtest 2021: MEC012: Rnis Query BI BO RabInfo and PlmnInfo using wrong parameter errors in the URL path	MEC012
38	Plugtest 2021: Specification discussion: non-existent attribute in a json body should be ignored	ALL
39	Plugtest 2021: MEC012: RNIS BI-BO: Error code expected 404 for L2Meas with wrong query param	MEC012

40	Plugtest 2021: MEC013: UELOCLOOK_001_OK not finding a defined parameter	MEC013
41	Review the TPs and TCs on MEC 013	MEC013

Table 28: Issues on the MEC-DEC032 Robot Test Suites

10.3 Other outcome

10.3.1 OSM Descriptor translation to NFV-SOL006

As OSM Release NINE completed native adoption of [NFV-SOL006] as Data Model for the VNF and NS Descriptors, the OSM community provided a tool and supported participating NF providers to translate their legacy OSM Descriptors in the standardized [NFV-SOL006] format.

A short training and some collateral (documentation, video) were prepared by the OSM community and provided to participants willing to migrate their legacy descriptors. Overall, eight descriptors were successfully translated by three different VNF providers.

The adoption of standardized descriptor models [NFV-SOL001] and / or [NFV-SOL006] by NF providers, NFVOs and VNFMs, is becoming key for successful NFV API Testing as from v2.7.1 some NFV API tests include descriptor checks.

This move towards standardized descriptors should also improve interoperability and reduce integration efforts across NFs, VNFMS and NFVOs.

10.3.2 CNCF CNF Conformance Testing

During this Plugtests and experimental track lead by the CNCF was offered to containerized network functions providers to self-assess the level of conformance of their Telco workloads with the Cloud Native principles and best practices defined by the CNCF.

The CNF Test Suite is part of a larger set of CNCF Cloud Native Network Function initiatives which have the goal of increasing interoperability, reducing operational risks, and speeding up the development to production pipeline of CNF workloads. Related initiatives include the Cloud Native Network Function Working Group (CNF WG) and the CNF Testbed. The CNF WG creates and maintains the definitions, processes and best practices and cloud native principles. The work on the mechanics of the tests, implementation of tests which evaluate cloud native best practices, and the test framework itself occurs in CNF Test Suite.

Testing in this external experimental track used the latest CNF Test Suite release [CNCF-CNF-TS] and some participants chose to run development snapshots. Participants were given the choice to run the CNF Test Suite on their own Kubernetes cluster or to request a CNCF community test environment to run the testing. Two ETSI Plugtests participants used the provided Kubernetes clusters. At least two additional participants ran the CNF Test Suite on their own Kubernetes environment.

The CNF Test Suite found non-cloud native practices in real-world network functions, like inability to rollback to an earlier version and the inclusion of hard coded IP addresses. The CNF Test Suite team discussed with participants why those tests failed in the dedicated slack channel that was created for the Plugtests event.

Feedback from ETSI Plugtests participants was used to make enhancements to the CNF Test Suite, including supporting self-hosted Docker Image Registries, supporting a username/password protected Docker Image Registry, and supporting non standard Image URL ports. Additional enhancements requested by Plugtests participants have been prioritized for: adding support to run the CNF Test Suite on MacOS and adding more explanation to failed test results on how to better follow cloud native best practices.

For future Plugtests, a more structured Beta Test process could be used to initially gather the interest of participants, to provide managed Kubernetes clusters, and to obtain guided feedback from Beta Testers. This feedback would of course be shared with ETSI Plugtests organizers.

10.3.3 CNF Conformance comparison with NFV-EVE011

The CNCF CNF Conformance experimental track brough the opportunity to learn and compare the cloud native principles defined for Telco workloads assessed by the CNF Conformance Test Suite [CNCF-CNF-TS] with the principles defined for Cloud Native Network Functions by ETSI NFV in [NFV-EVE011].

The main findings of this comparison are listed in the table below:

CNF Test Categories	CNF Goals	Relation to [NFV-EVE011]
Compatibility Tests	CNFs should work with any Certified Kubernetes product and any CNI-compatible network that meet their functionality requirements. The CNF Conformance Suite validates this: On platforms: o Performing CNI Plugin testing which: ▪ Tests if CNI Plugin follows the CNI specification On workloads: o Performing K8s API usage testing by running API snoop on the cluster which: ▪ Checks alpha endpoint usage ▪ Checks beta endpoint usage ▪ Checks generally available (GA) endpoint usage	Related to clause 5.8
Statelessness Tests	The CNF conformance suite checks if state is stored in a custom resource definition or a separate database (e.g. etcd) rather than requiring local storage. It also checks to see if state is resilient to node failure: On workloads: o Resetting the container and checking to see if the CNF comes back up o Using upstream projects for chaos engineering (e.g Litmus)	Requirements in clause 5.5.3
Security Tests	CNF containers should be isolated from one another and the host. The CNF Conformance suite uses tools like OPA Gatekeeper, Falco, Sysdig Inspect and gVisor: On platforms: o Check if there are any shells On workloads: o Check if any containers are running in privileged mode o Check if any protected directories or files are accessed	Not addressed
Microservice Tests	The CNF should be developed and delivered as a microservice. The CNF Conformance suite tests to determine the organizational structure and rate of change of the CNF being tested. Once these are known we can determine whether the CNF	Requirements in clause 5.3.3 & 5.7.2

	is a microservice. See: Microservice-Principles: On workloads: o Check if the CNF have a reasonable start-up time. o Check the image size of the CNF.	
Scalability Tests	The CNF conformance suite checks to see if CNFs support horizontal scaling (across multiple machines) and vertical scaling (between sizes of machines) by using the native K8s kubectl: On workloads: o Test increasing/decreasing capacity o Test small scale autoscaling with kubectl o Test large scale autoscaling with load test tools like CNF Testbed o Test if the CNF control layer responds to retries for failed communication (e.g. using Pumba or Blockade for network chaos and Envoy for retries) (see scalability test usage documentation)	Requirements in clause 5.2.5
Configuration and Lifecycle Tests	Configuration and lifecycle should be managed in a declarative manner, using ConfigMaps, Operators, or other declarative interfaces. The Conformance suite checks this by: On workloads: o Testing if the CNF is installed using a versioned Helm v3 chart o Searching for hardcoded IP addresses, subnets, or node ports in the configuration o Checking for a liveness entry in the helm chart and if the container is responsive to it after a reset (e.g. by checking the helm chart entry) o Checking for a readiness entry in the helm chart and if the container is responsive to it after a reset o Checking if the pod/container can be started without mounting a volume (e.g. using helm configuration) that has configuration files o Testing to see if we can start pods/containers and see that the application continues to perform (e.g. using Litmus) o Testing by resetting any child processes, and when the parent process is started, checking to see if those child processes are reaped (i.e. monitoring processes with Falco or sysdig-inspect) o Testing if the CNF can perform a rolling update (also rolling	Requirements in clause 5.9.4

	downgrade) (i.e. kubectl rolling update) o Testing if the CNF can perform a rollback (i.e. kubectl_rollout_undo) o Testing if there are any (non-declarative) hardcoded IP addresses or subnet masks	
Observability Tests	In order to maintain, debug, and have insight into a protected environment, its infrastructure elements must have the property of being observable. This means these elements must externalize their internal states in some way that lends itself to metrics, tracing, and logging. The Conformance suite checks this: On workloads: o Testing to see if there is traffic to Fluentd o Testing to see if there is traffic to Jaeger o Testing to see if Prometheus rules for the CNF are configured correctly (e.g. using Promtool) o Testing to see if there is traffic to Prometheus o Testing to see if the tracing calls are compatible with OpenTelemetry o Testing to see if the monitoring calls are compatible with OpenMetric On platforms: o Testing to see if there is an OpenTelemetry compatible service installed o Testing to see if there is an OpenMetric compatible service installed	Not addressed
Installable and Upgradeable Tests	The CNF Conformance suite will check for usage of standard, in-band deployment tools such as Helm (version 3) charts. The Conformance suite checks this: On workloads: o Testing if the install script uses Helm v3 o Testing if the CNF is published to a public helm chart repository. o Testing if the Helm chart is valid (e.g. using the helm linter) o Testing if the CNF can perform a rolling update (i.e. kubectl rolling update)	Not addressed
Hardware Resources and Scheduling Tests	The CNF container should access all hardware and schedule to specific worker nodes by using a device plugin. The CNF Conformance suite checks this:	Not addressed

	On platforms: o Testing if the Platform supplies an OCI compatible runtime o Testing if the Platform supplies a CRI compatible runtime On workloads: o Checking if the CNF is accessing hardware in its configuration files o Testing if the CNF accesses hardware directly during run-time (e.g. accessing the host /dev or /proc from a mount) o Testing if the CNF accesses huge pages directly instead of via Kubernetes resources o Testing if the CNF Testbed performance output shows adequate throughput and sessions using the CNF Testbed (vendor neutral) hardware environment.	
Resilience Tests	Cloud Native Definition requires systems to be Resilient to failures inevitable in cloud environments. CNF Resilience should be tested to ensure CNFs are designed to deal with non-carrier-grade shared cloud HW/SW platform: On platforms: o Test for full failures in SW and HW platform: stopped cloud infrastructure/platform services, workload microservices or HW ingredients and nodes o Test for bursty, regular or partial impairments on key dependencies: CPU cycles by pausing, limiting, or overloading; DPDK-based Data plane networking by dropping and/or delaying packets. o Test if the CNF crashes when network loss occurs (Network Chaos) Tools to study/use for such testing methodology: The previously mentioned Pumba and Blocade, ChaosMesh, Mitmproxy, Istio for “Network Resilience”, kill -STOP -CONT, LimitCPU, Packet pROcessing eXecution (PROX) engine as Impair Gateway.	Requirements in clause 5.1.6
		EVE011 requirements on load balancing (clause 5.10) do not seem to be checked by any of the test categories in the CNF Test Suite

Table 28: Issues on the NFV-TST010 Robot Test Suites

History

Document history		
V1.0.0	13/04/2021	Publication