ETSI Plugtests Report V1.0.0 (2018-08)

3rd ETSI NFV Plugtests
Sophia Antipolis, France
29th May – 8th June 2018

ETSI

Keywords
Testing, Interoperability, NFV, MANO, VNF, VIM

ETSI

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

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

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

Important notice

Copyright Notification

Contents

Executive summary

The 3rd NFV Plugtests was organised by the ETSI Centre for Testing and Interoperability, and hosted by ETSI in Sophia Antipolis, France from 29th May to 8th June 2018, in co-location with the OPNFV Fraser Plugfest from 4th to 8th June.

The main goal of the NFV Plugtests is to run multi-vendor interoperability test sessions among different Functions Under Test (FUTs) provided by different participants. Over 45 organisations and 200 engineers were involved in the preparation of this two-week event forming an engaged and diverse community of NFV implementers participating with:

- 19 Virtual Network Functions (VNF), some of them providing also Element Manager (EM) and specific VNF Manager (VNFM) functionalities. These VNFs were combined in 27 different multi-vendor Network Services (NS)
- 10 Management and Orchestration (MANO) solutions, providing NFV Orchestrator (NFVO) and VNFM functionalities
- 9 NFV Platforms, including hardware, providing NFV infrastructure (NFVI) and Virtual Infrastructure Manager (VIM) functionalities

Different participating implementations are able to interact remotely since the moment they join the Plugtests Programme through the NFV HIVE (Hub for Interoperability and Validation at ETSI) which provides a secure framework to interconnect participants' labs and implementations and prepare for the Plugtests.

This 3rd NFV Plugtests offered participants not only interoperability test sessions, but also NFV API validation activities and the opportunity to prepare and showcase real use case demonstrations combining different participants' implementations.

During the interop test sessions, the FUTs described above were combined in different configurations evolving from simple single-site NS deployments to complex Multi-Site deployments of multi-vendor Network Services featuring EPA or Service Function Chaining.

The interoperability results for multi-vendor network services test sessions were slightly higher than the ones reported in 2nd NFV Plugtests (held only a few months before in January 2018). While the number of this type of test sessions was lower, the overall number of test cases run saw an increase of 11%, quite possibly due to the fact that the number of test cases for these sessions was higher, and therefore, the sessions longer. The small decrease in the number of this type of sessions was highly compensated for by the significant increase of sessions and results reported for parallel activities such as automated testing and API Validation sessions, as we see in the figures below.

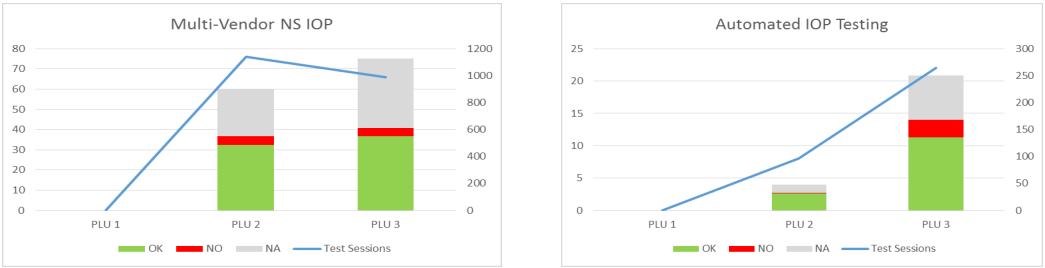

Figure 1a. NFV Plugtests overview: Multi-vendor NS and Automated Interop Testing

The figures show a significant increase in the number of automated test sessions (+175%) and the overall number of individual test cases that could be run automatically by a test system (+425%). This was due both to the increase in the number of test cases that were automated and by a growing interest in automation among Plugtests participants.

The NFV API validation track run by ETSI in parallel with the interoperability test sessions allowed participants to evaluate the alignment of their implementations with NFV SOL OpenAPIs. The scope of this API track, which was launched as experimental during the 2nd NFV Plugtests with a subset of NFV-SOL002 and NFV-SOL003 APIs, was extended to include NFV-SOL005, the Northbound Interface of NFV MANO.

The API track saw also a growing interest among participants, with a significant increase in the number of test sessions (+125%) and overall number of test cases run (+100%).

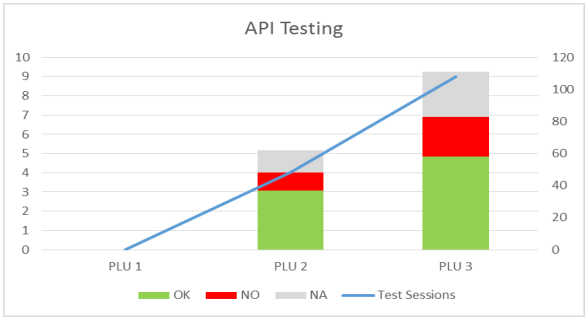

Figure 1.b. NFV Plugtests overview (API Testing)

The synergy with and across open source communities involved in the NFV Plugtests experienced also a visible growth with the arrival of SonataNFV and Open Air Interface, who joined ETSI OSM, Open Baton, Open Stack and OPNFV as supporting open source communities. Also, the co-location of the Plugtests with the OPNFV Fraser Plugfest during the second week provided additional opportunities of cross-community interaction showcased in some of the demos and cross project integration initiatives that were kicked off during the event.

The following clauses describe in detail the preparation of the 3rd NFV Plugtests, the participating implementations, the test plans and testing procedures, the overall results, as well as the lessons learnt and the feedback collected during the Plugtests on NFV Specifications and Open APIs.

The test plans and the present report are fed back to ETSI NFV Industry Specification Group.

1 Introduction

This Plugtests aimed at verifying interoperability among different implementations of the main components of the NFV Architectural Framework, which included:

- Virtual Network Functions (VNF), eventually providing additional Element Manager (EM) and specific VNF Manager (VNFM) functionalities
- Management and Orchestration (MANO) solutions, providing integrated NFV Orchestrator (NFVO) and VNFM functionalities
- NFV Platforms, including hardware, providing integrated NFV infrastructure (NFVI) and Virtual Infrastructure Manager (VIM) functionalities

Test and support VNFs were used to build the reference Network Services (NS) required to validate the proper behaviour of the Systems Under Test.

In order to enable remote integration and pre-testing among participants and allow them to get prepared for Plugtests, a dedicated VPN based network was built to interconnect local and remote implementations in a reliable and secure way: the NFV Plugtests HIVE: Hub for Interoperability and Validation at ETSI.

All of the participating implementations, Functions Under Test or Test/support Functions were connected and/or accessible through the HIVE network: most of the NFV platforms and MANO solutions run remotely on participants' labs, a subset of them were deployed locally at ETSI. VNF and NS Packages and images were made available in a local repository hosted at ETSI and uploaded to the different NFV platforms during the pre-testing phase. Some test and support VNFs were deployed locally on local infrastructure at ETSI, and also accessible through HIVE.

All the information required to organise, coordinate and manage the 3rd NFV Plugtests was compiled and shared with participants in a dedicated private WIKI. Most of the information presented in this document has been extracted from the NFV Plugtests wiki: https://wiki.plugtests.net/NFV-PLUGTESTS (login required).

.

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.

[NFV002] ETSI GS NFV 002: "Network Functions Virtualisation (NFV); Architectural Framework".

[NFV003] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for main concepts in NFV".

[IFA005] ETSI GS NFV-IFA 005: "Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification".

[IFA006] ETSI GS NFV-IFA 006: "Network Functions Virtualisation (NFV); Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification".

[IFA007] ETSI GS NFV-IFA 007: "Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification".

[IFA008] ETSI GS NFV-IFA 008: "Network Functions Virtualisation (NFV); Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specification".

[IFA010] ETSI GS NFV-IFA 010: "Network Functions Virtualisation (NFV); Management and Orchestration; Functional requirements specification".

[IFA013] ETSI GS NFV-IFA 013: "Network Functions Virtualisation (NFV); Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification".

[TST002] ETSI GS NFV-TST 002: “Network Functions Virtualisation (NFV); Testing Methodology; Report on NFV Interoperability Testing Methodology”

[TST007] ETSI GR NFV-TST 007: “Network Functions Virtualisation (NFV); Testing; Guidelines on Interoperability Testing for MANO”

[SOL002] ETSI GS NFV-SOL 002 V2.4.1: “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Ve-Vnfm Reference Point”

[SOL003] ETSI GS NFV-SOL 002 V2.4.1: “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point”

[SOL005] ETSI GS NFV-SOL 005 V2.4.1: “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point”

[1NFVPLU-TP] 1st ETSI NFV Plugtests Test Plan: https://portal.etsi.org/Portals/0/TBpages/CTI/Docs/1st_ETSI_NFV_Plugtests_Test_Plan_v1.0.0.pdf

[2NFVPLU-TP] 2nd ETSI NFV Plugtests Test Plan: https://portal.etsi.org/Portals/0/TBpages/CTI/Docs/2nd_ETSI_NFV_Plugtests_Test_Plan_v1.0.0.pdf

[3NFVPLU-TP] 3rd ETSI NFV Plugtests Test Plan: https://portal.etsi.org/Portals/0/TBpages/CTI/Docs/3rd_ETSI_NFV_Plugtests_Test_Plan_v1.0.0.pdf

[1NFVPLU-R] 1st ETSI NFV Plugtests Report: https://portal.etsi.org/Portals/0/TBpages/CTI/Docs/1st_ETSI_NFV_Plugtests_Report_v1.0.0.pdf

[2NFVPLU-R] 2nd ETSI NFV Plugtests Report: https://portal.etsi.org/Portals/0/TBpages/CTI/Docs/2nd_ETSI_NFV_Plugtests_Report_v1.0.0.pdf

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

3 Abbreviations

For the purposes of the present document, the terms and definitions given in [NFV003] and [TST002] apply.

4 Technical and Project Management

4.1 Scope

The main goal of the 3rd NFV Plugtests was to run multi-vendor interoperability test sessions allowing to validate ETSI NFV end-to-end capabilities such as management of descriptors and software images, life cycle management of network services, virtual network functions and virtual resources, fault and performance management, enhanced platform awareness, and Multi-Site deployments.

During the interoperability Test Sessions, the Systems Under Test (SUTs) were made of different combinations of the following Functions Under Test (FUTs):

- One or several NFV Platforms, including hardware and providing pre-integrated VIM and NFVI functionality
- One MANO solution, providing pre-integrated NFVO and generic VNFM functionality
- One Multi-Vendor Network Service, composed of VNFs from different providers, possibly including EM and specific VNFM management functionality.

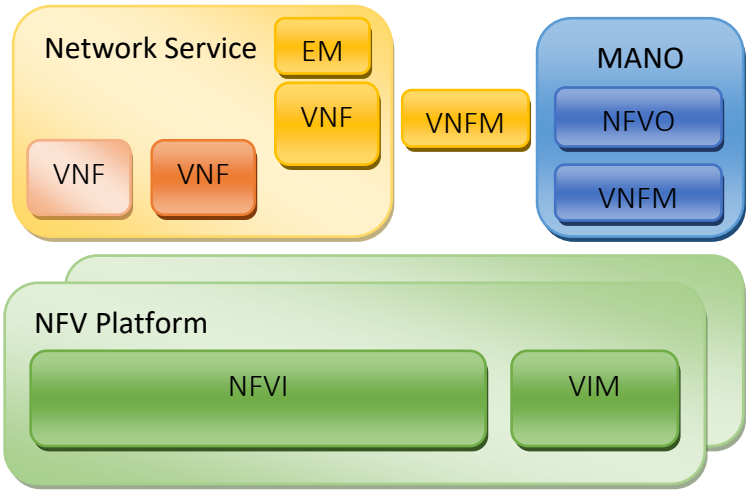

Figure 2a. IOP System Under Test

In addition and separately from the interoperability test sessions, a track on NFV API validation was run by ETSI allowing participants to test their API implementations against a subset of NFV SOL OpenAPIs.

4.2 Timeline

The 3rd NFV Plugtests preparation started soon after closure of the 2nd NFV Plugtests and run through different phases as described in the figure below.

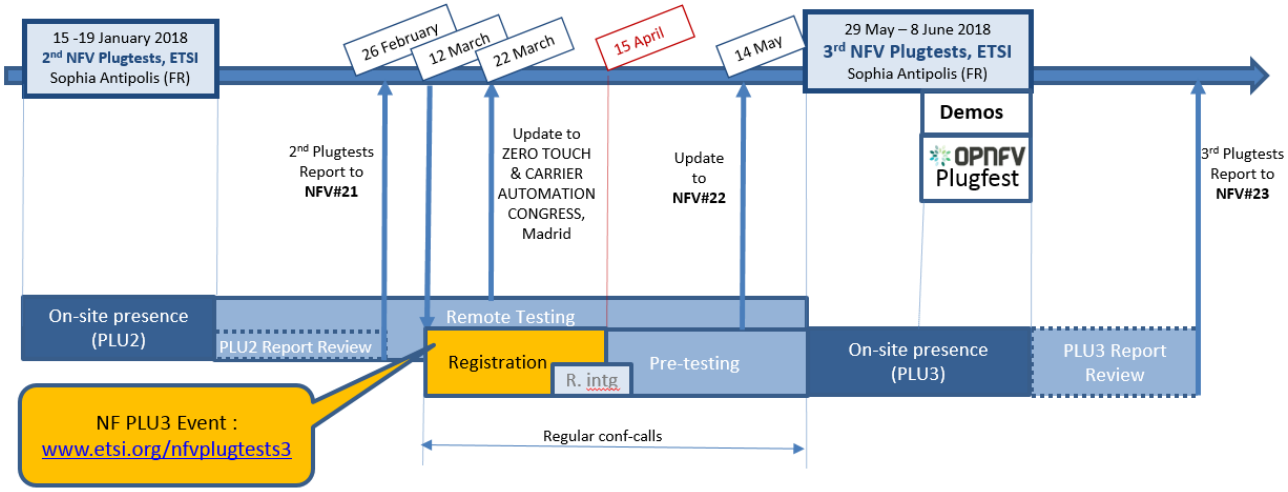

Figure 3. 3rd NFV Plugtests timeline

Registration to Plugtests was open from March to mid-April 2018 to any organisation willing to participate with a Function Under Test, or to support the testing. A number of observers (network operators, research, and academia) and Open Source communities (ETSI OSM, Open Baton, Open Stack, OPNFV, SonataNFV and Open Air Interface Software Alliance) participated to the test plan development and review. Overall, over 200 people were involved in the Plugtests either locally or remotely.

The following clauses describe the different phases of the Plugtests preparation. It is worth noting the preparation phases, weekly conf-calls were run among organisers and participants to discuss and track the remote-integration progress, anticipate and solve technical issues, review the test plan, and prepare for the face to face test sessions.

4.2.1 Remote Integration

During the Remote Integration phase, the following activities were run in parallel:

1) FUT Documentation

Participants documented their FUTs, by filling in or updating a form compiling the Interoperability Features Statement (IFS) and Technical Questions (TQ) concerning their implementations. The final IFS Templates for each type of FUT can be found in the annex of the 3rd NFV Plugtests Test Plan [3NFVPLU-TP]

Participants providing VNFs complemented their documentation with diagrams and resource requirements. As for the previous NFV Plugtests, some example VNFs and NSs were made available by the Plugtests team to facilitate the documentation of participating VNFs. These examples are documented in the Annex A of the 1st NFV Plugtests Report [1NFVPLU-R].

Participants providing MANO solutions developed and made available descriptor samples for the VNF and NS examples and supported the VNF providers in the creation of their own VNF and NS Descriptors.

Participants providing NFV Platforms created and documented tenants/projects and credentials for each participating MANO solution, and exposed and documented the North Bound Interfaces (NBI) of their VIM.

All the information described above was made available in the Plugtests WIKI, so that it could be easily maintained and consumed by participants.

2) Test Plan Development

The Test Plan development was led by ETSI Centre for Testing and Interoperability following the methodology and guidelines defined by ETSI NFV TST WG in TST002 and TST007, and building on the learnings and achievements of previous NFV Plugtests.

The Test Plan was developed and consolidated in an iterative way, taking into account input and feedback received from different stakeholders: ETSI NFV TST WG, supporting Open Source Communities and Plugtests participants. See details in Clause 8.

3) Connection to HIVE

The interconnection of different FUTs involved in the testing relies on HIVE: Hub for Interoperability and Validation at ETSI. NFV Plugtests Programme participants can keep and use their connection to HIVE between Plugtests events. New participants that joined the programme after the 2nd NFV Plugtests, were invited and helped to get their implementations available on HIVE.

At the end of this phase, up to 46 remote sites were connected to HIVE and each of them was allocated a dedicated network. The interconnection of remote labs allowed to run integration and pre-testing tasks remotely among any combination of participating FUTs, and helped to ensure an efficient use of the face to face Plugtests time and a smoother run of Interoperability Test Sessions.

A site-to-site connection to HIVE was mandatory for participants providing NFV Platforms and MANO Solutions, and highly recommended for participants providing VNF and VIM software. The latest could also rely on client-to-site connection to HIVE, as long as they had no software (i.e. support function) running locally in their labs and only required access to remote labs for trouble shooting and infrastructure access purposes

Additional details on the remote test infrastructure are provided in Clause 6.

4) Once the above steps were completed, FUTs could start cross-FUT remote integration, see 7.1 for details on the procedures.

4.2.2 Pre-testing

Once remote integration was completed, participants had the opportunity to run remote pre-testing among different combinations of VNF, MANO and NFV Platforms.

The pre-testing test plan was a subset of the tests run during the 1st NFV Plugtests, in order to allow participants to concentrate on advanced features and configurations during the on-site phase.

Additional details on the pre-testing plan and procedures are provided in Clause 7.

4.2.3 On-site

From 29th of May to 8th of June, participants sent representatives to ETSI to collaboratively run Interoperability Test Sessions. The 2 on-site Plugtests weeks were organised as follows:

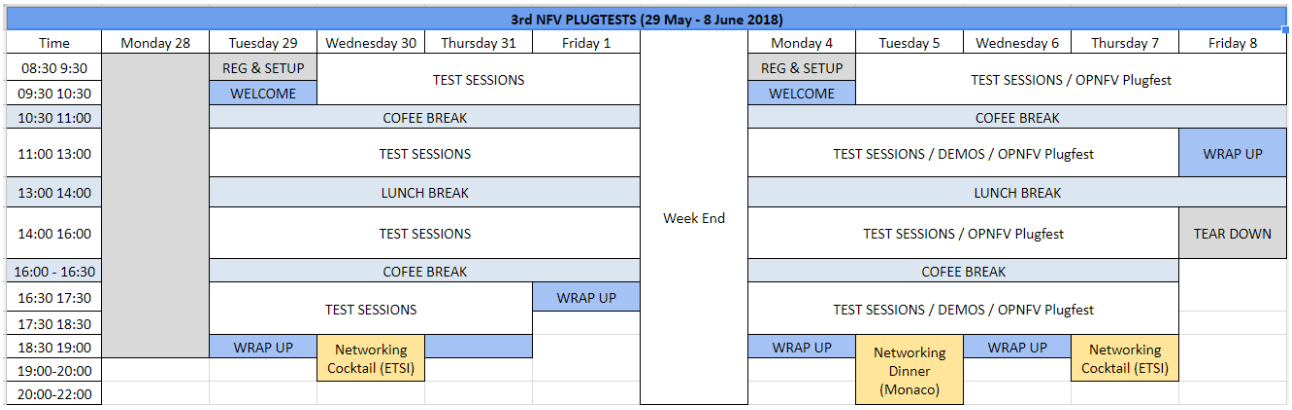

Figure 4. Plugtests on-site HL plan

The first week (4 days) concentrated on interoperability test sessions, in which participating FUTs were organised and combined in several parallel tracks, see details in Clause 4.3.2. An additional track was dedicated to NFV API testing.

During the second week, interoperability and API testing tracks were complemented with OPNFV Plugfest sessions and multi-vendor demos.

4.3 Tools

4.3.1 Plugtests Wiki

The NFV Plugtests Wiki was the main entry point for all the information concerning this event, from logistics aspects to testing procedures. Access to this Wiki is restricted to companies participating to the NFV Plugtests Programme.

4.3.2 Test Session Scheduler

The Test Session Scheduler allowed the Plugtests organisers to produce a daily schedule during the on-site phase. This tool has the following objectives:

- maximise the number of test sessions
- balance the amount of test sessions among participants
- take into account supported features of the participating FUTs
- minimise the number of participants not involved in a test session anytime.

The picture below shows a partial view a daily schedule. Each yellow box corresponds to a specific Test Session addressing a Multi-Vendor Network Service configuration including 2 or more VNFs from different providers, 1 MANO solution and 1 or more VIM&NFVI. For each of these sessions a Test Session Report was reported (see next clause). In addition to the pre-scheduled test sessions, participants were invited to request, run and report results for additional test sessions targeting either pre-testing or advanced configurations addressing more complex combinations of FUTs and features.

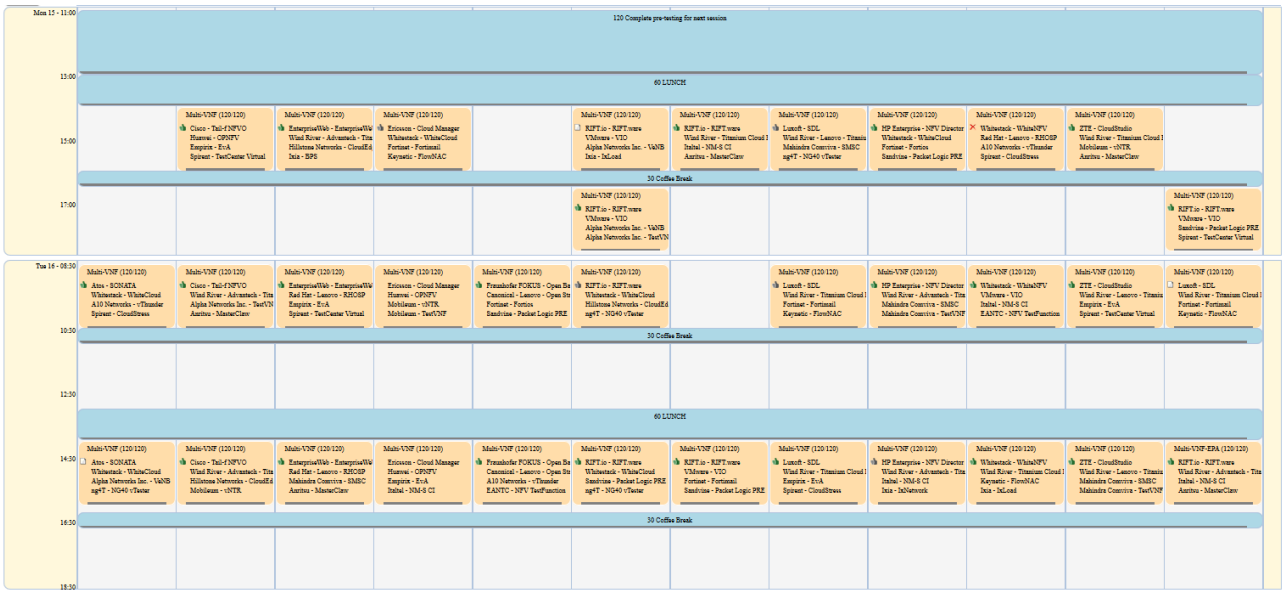

Figure 5. Daily Schedule example

4.3.3 Test Reporting Tool

The Test Reporting Tool guides participants through the Test Plan during the Test Sessions, and allows them to create Test Session Reports compiling detailed results for each test case in scope. It allows reporting on pre-scheduled Test Sessions, but also on Test Sessions organised on the fly among participants to prepare, complete or complement the scheduled testing (freestyle sessions).

Only the companies providing the FUTs and Test Functions (when applicable) for each specific Test Session have access to the Test Session Reports (TSR) contents and specific results. All the companies providing the FUTs for a Test Session, i.e. VNF provider(s), MANO provider and NFV Platform provider(s) are required to verify and approve the reported results at the end of the session.

3016	Freestyle			PreTesting	RIFT.io - RIFT.ware VMware - VIO Keynetic - FlowNAC	
3017	2018-01-18 11:00	120	Test Track 9	Multi-VNF-EPA	Whitestack - WhiteNFV Wind River - Titanium Cloud R4 Keynetic - FlowNAC Ixia - IxNetwork	
3018	2018-01-18 11:00	120	Test Track 6b	Multi-VNF	RIFT.io - RIFT.ware VMware - VIO Keynetic - FlowNAC Ixia - IxNetwork	
3020	Freestyle			Multi-VNF	Whitestack - WhiteNFV Red Hat - Lenovo - RHOSP A10 Networks - vThunder Spirent - TestCenter Virtual	
3022	2018-01-18 11:00	120	Ad-hoc Testing	Multi-VNF-EPA	Luxoft - SDL Wind River - Titanium Cloud R4 Fortinet - Fortios Spirent - TestCenter Virtual	
3023	Freestyle			Multi-Site	Cisco - Tail-f NFVO Wind River - Advantech - Titanium Edge Wind River - Lenovo - Titanium Cloud R4 Empirix - EvA Spirent - TestCenter Virtual	
3024	Freestyle			PreTesting	HP Enterprise - NFV Director Huawei - OPNFV A10 Networks - vThunder	
3025	2018-01-18 14:00	120	Test Track 8	Multi-VNF	HP Enterprise - NFV Director Huawei - OPNFV Hillstone Networks - CloudEdge Alpha Networks Inc. - VeNB	
3026	Freestyle			PreTesting	Whitestack - WhiteNFV Wind River - Titanium Cloud R4 Empirix - EvA	
3027	2018-01-19 08:30	120	Test Track 9	Multi-VNF	Whitestack - WhiteNFV Red Hat - Lenovo - RHOSP Mobileum - vNTR	

Figure 6. Test Reporting Tool (extract of the TSR list)

Another interesting feature of this tool is the ability to generate real-time statistics (aggregated data) of the reported results, per test case, test group, test session or overall results. These stats are available in real time for all participants and organisers and allow tracking the progress of the testing with different levels of granularity, which is extremely useful to analyse the results.

5 Participation

5.1 Functions Under Test

The tables below summarise the different Functions Under Test provided by the Plugtests participants, and the location from where they were provided / supported or connected to the HIVE network:

5.1.1 VNFs

Organisation	Solution	Team Location	Short Description
A10 Networks	vThunder	USA/Germany	Secure Application Delivery Services: ADC, CGN, Firewall
Anritsu	Master Claw vProbe	Romania/Czech Republic	Virtual Probe for Customer Service Assurance
Athonet	vEPC	Italy	Virtual EPC
Citrix	VPX + MAS	UK / India	Virtual ADC + Management, configuration, anaytics
EANTC	NFV Test Function	Germany	Traffic generator
Fortinet	FortiGate	France	FW, Security
Hillstone Networks	CloudEdge	China/USA	NGFW, Security
IT Aveiro	Heimdall Web	Portugal	SFC-MITM VNF to accelerate SSL/TLS processing (cloud-based)
IT Aveiro	Heimdall Hybrid Web	Portugal	SFC-MITM VNF to accelerate SSL/TLS processing (hybrid: part cloud, part edge)
Italtel	iRPS	Italy	Centralized Routing Engine
Keysight	IxLoad	USA/Romania	Application traffic simulator
Keysight	IxNetwork	USA/Romania	Traffic generator
Keysight	BPS	USA/Romania	Application traffic simulator
Mobileum	vNTR	India	Virtual NTR, Roaming
ng4T	NG40 vTester	Germany	Simulator, functional, capacity and load vTester
OpenAirInterface SA Eurecom	OAI-EPC	France	EPC Network Functionality
Spirent	Avalanche Virtual	California, USA	L4-7 Traffic generation and analysis
Spirent	TestCenter Virtual	California, USA	Traffic Generator, Traffic Analyzer, Capture, Control Plane emulator

Table 1. VNFs Under Test

5.1.2 NS

The VNFs under test were combined in multi-vendor network services as follows:

NS Name	VNFs
NS1:A10 vThunder+FortiGate+STCv	vThunder (A10) FortiGate (Fortinet) TestCenter Virtual (Spirent)
NS2:A10 vThunder+STCv	vThunder (A10) TestCenter Virtual (Spirent)
NS3:A10 vThunder+IxLoad	vThunder (A10) IxLoad (Keysight)
NS4:A10 vThunder+EANTC	vThunder (A10) TestVNF (EANTC)
NS5:ng4T+OAI	vEPC (OAI) vTester (ng4T)
NS6:Athonet+IxLoad	vEPC (Athonet) IxLoad (Keysight)
NS7: Italtel+Anritsu	iRPS (Italtel) MasterClaw (Anritsu)
NS8: IT Aveiro+FortiGate	Heimdall Web (IT Aveiro) FortiGate (Fortinet)
NS9: Hillstone+Mobileum	CloudEdge (Hillstone) vNTR (Mobileum)
NS10:Citrix+STCv	vPX (Citrix) STCv (Spirent)
NS11:Athonet+ng4T	vEPC (Athonet) vTester (ng4T)
NS12:Anritsu+ng4T	MasterClaw (Anritsu) vTester (ng4T)
NS13:A10 vThunder+ng4T	vThunder (A10) vTester (ng4T)
NS14: Athonet+STCv	vEPC (Athonet) STCv (Spirent)
NS15: Citrix+EANTC	vPX (Citrix) TestVNF (EANTC)
NS16: FortiGate+STCv	FortiGate (Fortinet) STCv (Spirent)
NS17: FortiGate+Mobileum+STCv	Fortigate (Fortinet) vNTR (Mobileum) STCv (Spirent)
NS18: Italtel+IxLoad	iRPS (Italtel) IxLoad (Keysight)
NS19: Anritsu+Spirent	MasterClaw (Anritsu) STCv (Spirent)
NS20: Anritsu+IxLoad	MasterClaw (Anritsu) IxLoad (Keysight)
NS21: Mobileum+STCv	vNTR (Mobileum) STCv (Spirent)
NS22: Mobileum+IxLoad	vNTR (Mobileum) IxLoad (Keysight)
NS23: FortiGate+EANTC	Fortigate (Fortinet)

	TestVNF (EANTC)
NS24: Hillstone+IxLoad	CloudEdge (Hillstone) IxLoad (Keysight)
NS25: Italtel+EANTC	iRPS (Italtel) TestVNF (EANTC)
NS26: Mobileum+OAI+ng4T	vNTR (Mobileum) vEPC (OAI) vTester (ng4T)
NS27: vThunder+STCv+nginx	vThunder (A10) STCv (Spirent) nginx (SONATA)

Table 2. NSs Under Test

5.1.3 MANOs

Organisations	Name	Location	Short Description
Altice Labs Atos Demokritos iMEC	SONATA	Spain/ Netherlands	Open Source SONATA NFVO + Generic VNFM
Cisco	Network Services Orchestrator (NSO) and Elastic Services Controller (ESC)	USA	NFVO + Generic VNFM
EnterpriseWeb	EnterpriseWeb	USA/Canada	Microservice-based NFVO + Generic VNFM
Ericsson	Cloud Manager	New Jersey, USA	MANO Orchestrator, Generic VNFM,
Huawei	CloudOpera Orchestrator NFV	Xi'an/China	NFVO + Generic VNFM
Luxoft	SDL	Romania	NFVO + Generic VNFM
Netcracker	Hybrid Orchestrator	Waltham, MA	NFVO + Generic VNFM
RIFT.io	RIFT.ware	USA	Carrier Grade NFVO + Generic VNFM
Ubiqube	OpenMSA	Ireland	Network and Security Automation Framework
Whitestack	WhiteNFV	USA	OSM distribution (NFVO + Generic VNFM)

Table 3. MANOs Under Test

5.1.4 VIM&NFVIs

Organisations	Name	HW	Location	Short Description
Huawei	OPNFV Compass	OPNFV POD @Huawei	Singapore	OPNFV: OpenStack Ocata + Pike, ODL Nitrogen
Nokia	NCIR18	AirFrame OR	Espoo, Finland	OpenStack Pike
Red Hat	Red Hat OpenStack Platform	Lenovo POD - SR630/SR650 and OCP servers - G8272/NE2572 and G8052 switches	NC, USA	OpenStack Newton

Red Hat	Red Hat OpenStack Platform	QCT @ETSI	France	OpenStack Newton
SUSE	OPNFV XCI (Open SUSE)	POD @UNH IOL	NH, USA	OpenStack Queens + ODL Oxygen
Whitestack	WhiteCloud	Intel remote POD	USA	OpenStack Pike + ODL Carbon
Wind River	Titanium Cloud R4 – Core Configuration	Wind River @OSM Remote Lab	Hudson, MA, USA	OpenStack Newton + OF 1.3
Wind River	Titanium Cloud R5 – Edge Configuration	Advantech remote POD	Taipei Taiwan	OpenStack Pike + OF 1.3
Wind River	Titanium Cloud R5 – Core Configuration	Lenovo POD - SR630/SR650 and OCP servers - G8272/NE2572 and G8052 switches	NC, USA	OpenStack Pike + OF 1.3

Table 4. VIM&NFVIs Under Test

5.2 Test Functions

In addition to the Test VNFs included in the VNF section, the following Test Functions were available during the Plugtests to support or complement the scope of the Test Sessions. Their use was optional.

Organisation	Name	Team Location	Short Description
EANTC	Auto LCM Validation	Berlin, Germany	TD automation tool
Ericsson	OPNFV Dovetail		VIM&NFVI validation tool
Spirent	Lifecycle Validation	California, USA	Automatic Lifecycle Validation tool
Huawei	OPNFV Bottlenecks	Shanghai, China	VIM&NFVI validation tool

Table 5. Test Functions

5.3 Technical Support

The organisations below provided technical support and expertise to the Plugtests Team and contributed actively to the test plan development and technical arrangements to prepare and run the Plugtests.

Organisation	Role
Nextworks	Technical Support

Table 6. Technical Support

5.4 Observers

The following organisations joined the NFV Plugtests as observers and contributed with technical advice and test plan review:

Organisation	Role
BUPT	Beijing University of Posts and Telecommunications
Cable Labs	Non-profit R&D consortium for cable providers
CAICT	China Academy of Information and Communications Technology
DOCOMO	Telecommunications service provider
Orange	Telecommunications service provider

Table 7. Observers

5.5 Open Source Communities

The Open Source communities listed below were actively involved in the Plugtests preparation and contributed to the Test Plan review. Their solutions were widely present in the Test Sessions through multiple distributions:

Organisation	Role	Details
Open Source MANO	MANO	https://osm.etsi.org
SonataNFV (5GTango)	MANO	https://5gtango.eu
Open Stack	VIM&NFVI	https://www.openstack.org
OPNFV	VIM&NFVI	https://www.opnfv.org
Open Air Interface	VNF (vEPC)	http://www.openairinterface.org

Table 8. Supporting Open Source communities

6 Test Infrastructure

6.2 HIVE

The remote integration, pre-testing and on-site phases were enabled by the NFV Plugtests Programme's HIVE network

Figure 7. 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 46 remote locations including several OSM Remote Labs participating to the Plugtests leveraged the HIVE network to make their Functions Under Test available for the test sessions.

During the on-site phase, a local network deployed in the Plugtests room allowed participants to access the remote labs, the local VNF Repository and some support functions running on ETSI servers. One participating NFV Platform (VIM&NFVI) was also deployed locally in the ETSI Lab and connected to HIVE.

The figure below describes how the NFV and OSM HIVE networks are interconnected to support the NFV Plugtests activities.

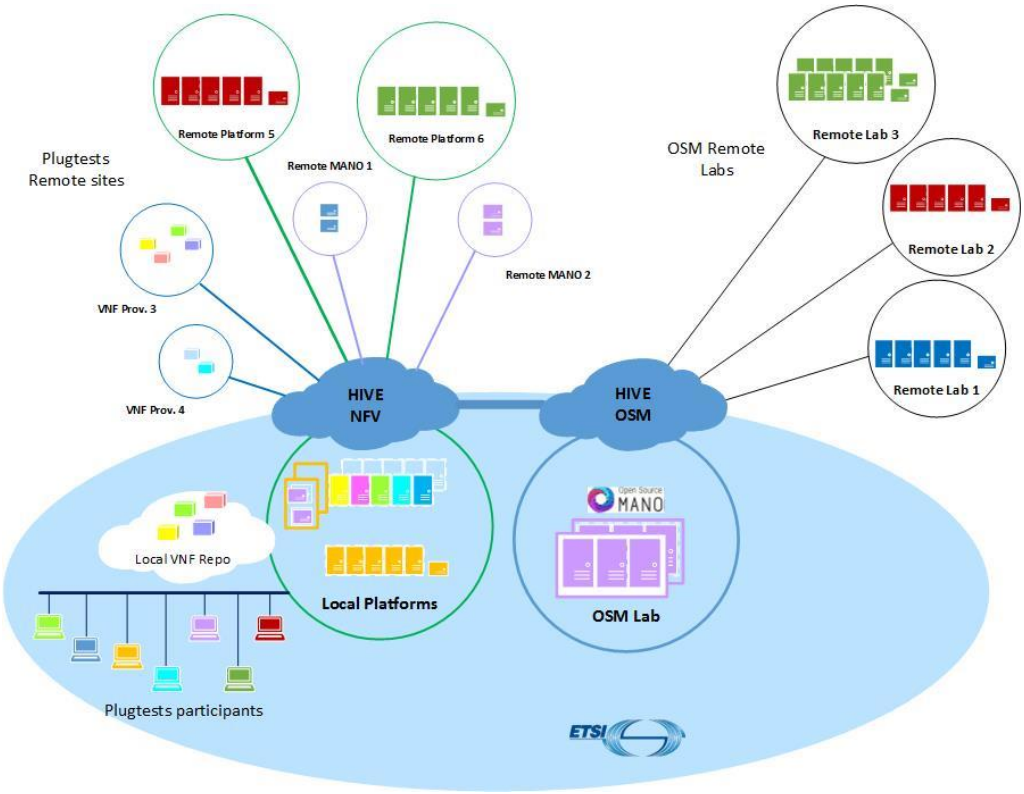

Figure 8. Remote Test Infrastructure

6.2 API Test System

The API Track required a Test System acting as API consumer for a number of APIs belonging to different NFV components and over different reference points. The capabilities required for the test system where:

- Sending configurable HTTP(S) requests
- Allowing custom payloads to be exchanged
- Automatically applying schema validation on the response payloads

The test system was implemented via the Postman HTTP testing environment, using as base input the OpenAPI descriptions provided by ETSI NFV.

7 Test Procedures

7.1 Remote Integration Procedures

Remote integration procedures for different FUTs and FUT combinations were documented in the WIKI (see next chapters) and the progress captured in a multi-dimensional tracking matrix which was reviewed regularly during the preparation calls. The tracking matrix was similar to the one shown below.

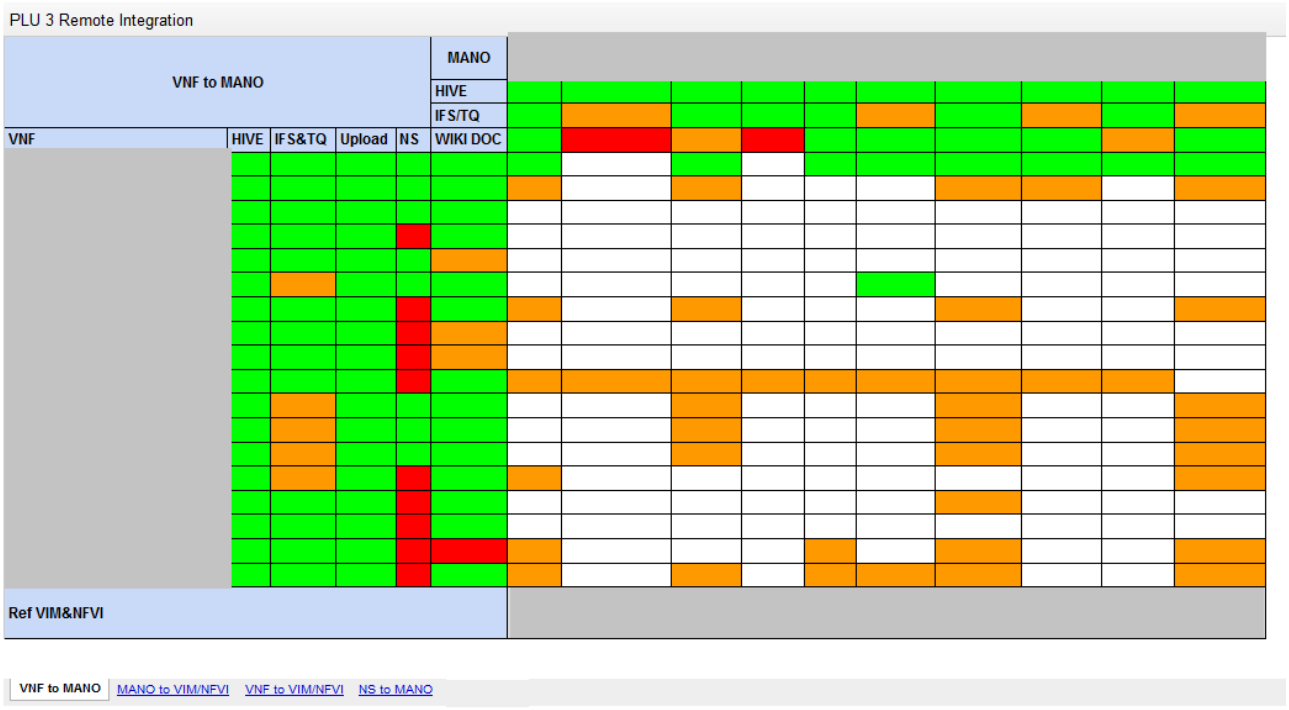

Figure 9. Pre-testing tracking matrix

The progress on each of the dimensions of the remote integration activities (VNF to MANO, MANO to VIM&NFVI VNF to VIM&NFVI and NS to MANO) was tracked following the colour code described below:

Pre-testing Matrix Colour Code	
	Not Started / Not Reported
	Not Applicable
	Pending
	Ongoing
	Completed

Table 9. Remote integration matrix colour code

7.1.1 Per-FUT Procedures

The next sections describe the independent procedures for each type of FUT:

7.1.1.1 VNF

1. **HIVE:** Connect to **HIVE** (site to site or client to site)
2. **IFS&TQ:** Fill in the VNF Interoperability Feature Statements and Technical Questions form
3. **Upload**: Upload VNF Package(s), descriptors, artifacts, and SW image(s) to the central VNF Repository.
4. **NS:** Team up with other VNF providers to build a multi-vendor NS
5. **WIKI DOC**: Provide a schema describing your VNF / NS

7.1.1.2 MANO

1. **HIVE:** Connect to **HIVE**. site-to-site VPN connectivity is required for MANO solutions
2. **IFS&TQ:** Fill in the MANO Interoperability Feature Statements and Technical Questions form
3. **WIKI DOC**: provide VNF and NS Descriptor examples for the Ref VNF&NS examples.

7.1.1.3 VIM&NFVI

1. **HIVE:** Connect to **HIVE** or ship HW to ETSI
2. **VIM Ready**: Install, start and configure VIM
3. **IFS&TQ:** Fill in the VIM&NFVI Interoperability Feature Statements and Technical Questions form
4. **WIKI DOC**: provide additional information required for MANO solutions to integrate: NBI IP@, projects, credentials, VNF Management IP pool, etc...

7.1.2 Cross-FUT Procedures

7.1.2.1 MANO to VIM&NFVI

1. Test connectivity from MANO to VIM&NFVI & from VIM&NFVI to MANO
2. Connect MANO to VIM (get NBI IP, project and credentials on VIM&NFVI wiki page)
3. Verify VIM resources can be accessed from MANO
4. Specify the reference VNF in the pre-testing table
5. Upload Ref VNF to VIM
6. On-board, instantiate and terminate Reference VNF from MANO (* see pre-testing sessions below)

7.1.2.2 VNF to VIM&NFVI

1. Upload VNF image to VIM
2. Verify physical network connectivity in NFVI will allow for VNF/NS deployment
3. Manual creation of VM's and network infrastructure required for this VNF (to prepare MANO On-board automatic execution)
4. Manual execution Instantiate VNF steps (to prepare MANO automatic execution)
5. Manual execution Scale in/out VNF steps (to prepare MANO automatic execution)
6. Manual execution Terminate VNF steps (to prepare MANO automatic execution)

7.1.2.3 VNF to MANO

1. Identify the reference VIM&NFVI(s) that will be used for pre-testing
2. Create VNFD/NSD for MANO, and upload to VNF Repository (VNF folder under NFVPLU3)
3. On-board VNFD/NSD to MANO
4. Upload VNF image(s) to Ref VIM
5. Instantiate VNF/NS from MANO
6. Scale in/out
7. Terminate VNF/NS

7.1.2.4 NS to MANO

1. Identify the reference VIM&NFVI(s) that will be used for pre-testing
2. Create multi-vendor NSDs for MANO, and upload to VNF Repository (NS folder under NFVPLU3)
3. On-board NSD to MANO
4. Upload VNF image(s) to Ref VIM
5. Instantiate NS from MANO
6. Scale in/out
7. Terminate NS

7.2 Pre-testing Procedure

Following remote integration, participants were invited to run pre-testing sessions and formally capture the results in the Test Reporting Tool. The recommendation was to start pre-testing with a Single-VNF Network Service configuration involving one VNF, one MANO solution and one VIM&NFVI and increment complexity gradually.

Running and reporting results for a pre-testing session required all the involved parties (VNF, MANO, VIM&NFVI) to have successfully achieved remote integration (connection to HIVE, descriptors created and uploaded, images uploaded, credentials shared, VIM ready, etc...).

Participants were encouraged to find suitable testing partners in the pre-testing matrix (those that had completed remote integration) and arrange a convenient timeframe for pre-testing, leveraging Slack to facilitate discussions among the different teams.

Once the arrangements made the pre-testing procedure was as follows:

1) Open the Test Reporting Tool, go to the “Reports” TAB

2) Create a “Freestyle” report, select configuration = Pre-testing

3) Select the MANO, VIM&NFVI and VNF involved in the pre-testing session

Once the above completed, a Test Session Report would open, and participants would follow the Interoperability Testing Procedure remotely (see next clause)

7.3 Interoperability Testing Procedure

During the on-site part of the Plugtests, a daily Test Session Schedule was produced with the Plugtests Scheduler. Test Sessions were organised in several parallel tracks, ensuring that all participants had at least one pre-scheduled Test Session every day. Pre-scheduled Test Sessions addressed the multi-vendor Network Service configuration, involving one MANO solution, one or more VIM&NFVIs and one network serviced with at least 2 VNFs from different providers.

Pre-scheduled test sessions could be completed with additional sessions addressing other test configurations / test groups, such as Multi-site, EPA, Service Function Chaining, or specific VNFM deployments (see 8.3 for details)

Participants could choose to run the above mentioned additional testing as “freestyle” test sessions, and create a test report on the fly, or to ask the Plugtests team to schedule the additional sessions for them.

During each test session the procedure for interoperability testing was as follows:

1) The MANO, VIM&NFVI(s), and VNFs representative would sit together. Each VNF was expected to have previously run a single-VNF NS test session with the same MANO and VIM(s)

2) One representative of the team opened the Test Session Report and the Test Plan.

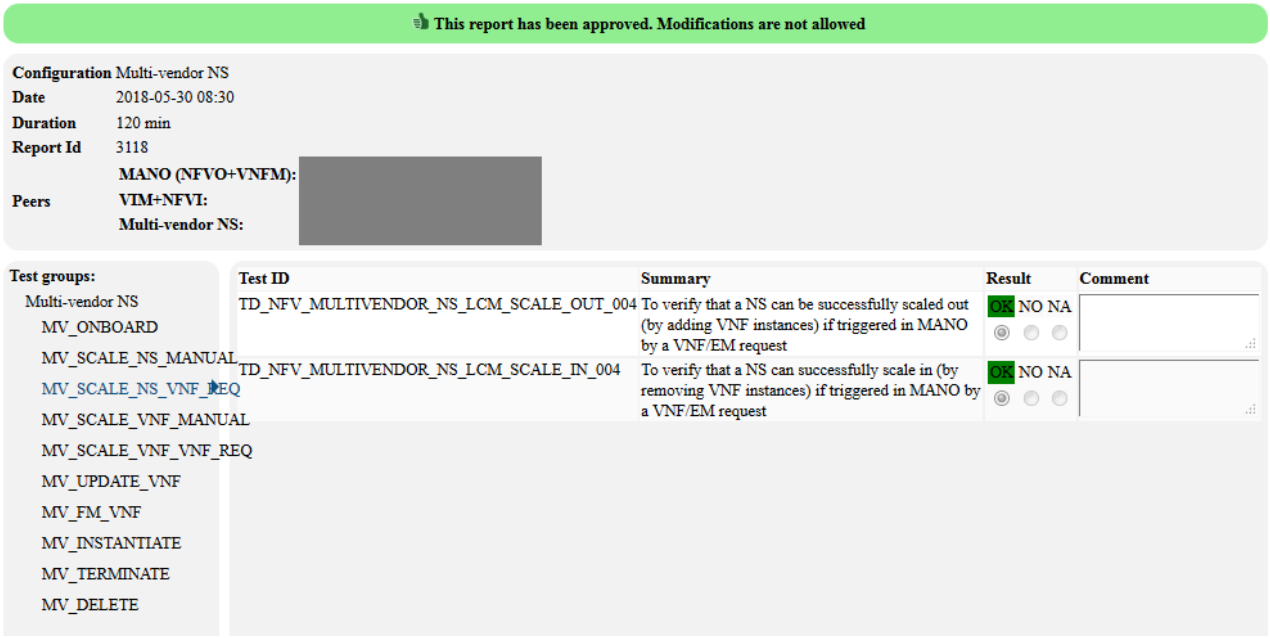

Figure 10. Test Session Report

3) For each Test in each group of the Test Plan:

a. The corresponding Test Description was applied to the target SUT Configuration:

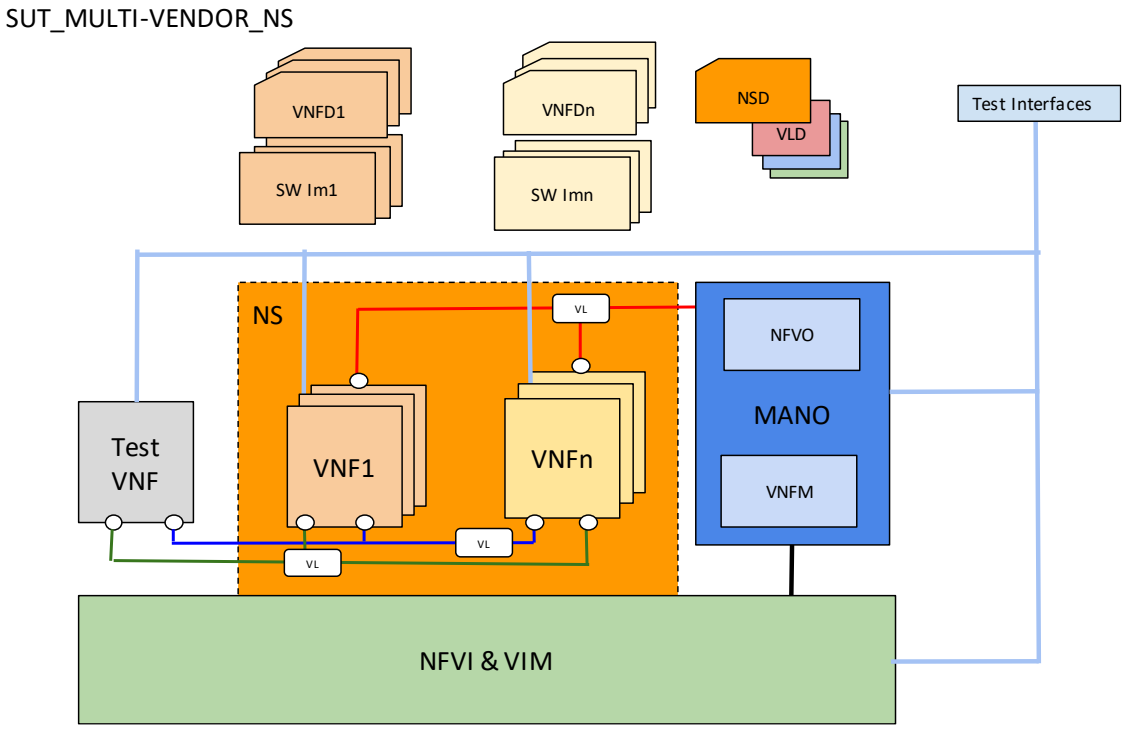

Figure 11. SUT Configuration example.

b. VNFs, MANO and VIM&NFVI providers jointly executed the different steps specified in the Test Description and evaluated interoperability through the different IOP Checks prescribed in it.

Interoperability Test Description				
Identifier	TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_004			
Test Purpose	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered in MANO by a VNF/EM request			
Configuration	SUT_MULTI-VENDOR_NS			
References	ETSI GS NFV-IFA005 V2.3.1 (clause 7.3.1.2, 7.4.1.2, 7.5.1.2, 7.7) ETSI GS NFV-IFA006 V2.3.1 (clause 7.3.1.2, 7.4.1.2, 7.5.1.2, 7.7) ETSI GS NFV-IFA007 V2.3.1 (clause 7.2.4) ETSI GS NFV-IFA008 V2.3.1 (clause 7.2.4)			
Applicability	* [IFS_NFV_MANO_16] MANO supports scaling out/in request from VNF/EM * [IFS_NFV_MANO_14] MANO supports scaling by adding/removing VNF instances * [IFS_NFV_VNF_4] VNF can scale out/in by adding/removing VNF instances * [IFS_NFV_VNF_8] VNF/EM can request scaling to MANO			
Pre-test conditions	* NS is instantiated (TD_NFV_MULTIVENDOR_NS_LCM_INSTANTIATE_001)			
Test Sequence	**Step**	**Type**	**Description**	**Result**
	1	Stimulus	Trigger the VNF/EM to send a scale out (by adding VNFs) request to MANO	
	2	IOP Check	Verify that the scale out (by adding VNF instance(s)) procedure has been started in MANO	
	3	IOP Check	Verify that the requested resources have been allocated by the VIM according to the descriptors	

	4	IOP Check	Verify that the additional VNF instance(s) have been deployed	
	5	IOP Check	Verify that the additional VNF instance(s) are running and reachable through the management network	
	6	IOP Check	Verify that the additional VNF instances(s) have been configured according to VNFD (i.e by obtaining a result from the management interface)	
	7	IOP Check	Verify that the additional VNF instances(s), VL(s) and VNFFG(s) are connected according to the Descriptors	
	8	IOP Check	Verify that NS has been scaled out by running the end-to-end functional test	
IOP Verdict				

Figure 12 Test Description example

c. The Test Result was reported to the Test Session Report, as follows:

i. OK: all IOP Checks were successful

ii. NOK: at least one IOP Check failed. A comment was requested.

iii. NA: the feature was not supported by at least 1 of the involved FUTs. A comment was requested to clarify the missing feature.

4) Once all the tests in the Test Session Report were executed and results recorded, the VNFs, MANO and VIM&NFVI providers reviewed the Report and approved it.

7.4 API Testing Procedure

The goal of the API track was to validate the compliance to interfaces as defined in ETSI NFV SOL specifications. The targeted specifications were again [SOL002] -for the Ve-Vnfm reference point- and [SOL003] – for the Or-Vnfm reference point and was extended to include [SOL005] for the Os-ma-nfvo reference point

The testing focused on checking the formal correctness of the exchanged HTTP payloads for one or more of the available interfaces within a reference point.

Participation to the API track was optional and required the FUT to implement at least one of the APIs under test. Given the reference points in scope, the API track targeted VNF/EM, VNFM and NFVO implementations.

The preparation of the API track was shared with participants during the remote integration phase and conf-calls, where participants were invited to provide recommendations and feedback.

Out of the 17 interfaces in the targeted specifications, the participants were given the possibility to test 15 Interfaces for which OpenAPI definitions were available at that time in the ETSI [FORGE].

Depending on the FUT provided, the availability of NFV API(s) implementation and their interest, each participant was asked to register its participation in the track and the interfaces to be tested during the remote integration part of the event.

For each interface, the tests focused on the compliance of the API producer (i.e. REST server), not on the API consumers (i.e. REST clients). The available interfaces are listed in the following table.

Interface	Specification	Producer	Web editor
SOL003-VirtualisedResourcesQuotaAvailableNotification-API	SOL003	NFVO	Link
SOL003-VNFFaultManagementNotification-API	SOL003	NFVO	Link
SOL003-VNFIndicatorNotification-API	SOL003	NFVO	Link

Interface	Specification	Producer	Web editor
SOL003-VNFLifecycleOperationGranting-API	SOL003	NFVO	Link
SOL003-VNFPackageManagement-API	SOL003	NFVO	Link
SOL003-VNFPerformanceManagementNotification-API	SOL003	NFVO	Link
SOL002-VNFConfiguration-API	SOL002	VNF	Link
SOL002-VNFIndicator-API	SOL002	VNF/EM	Link
SOL002-VNFIndicatorNotification-API	SOL002	VNFM	Link
SOL003-VNFFaultManagement-API	SOL003	VNFM	Link
SOL003-VNFIndicator-API	SOL003	VNFM	Link
SOL003-VNFLifecycleManagement-API	SOL003	VNFM	Link
SOL003-VNFPackageManagementNotification-API	SOL003	VNFM	Link
SOL003-VNFPerformanceManagement-API	SOL003	VNFM	Link
SOL005-NSDManagement-API	SOL005	NFVO	Link

Table 10. Interfaces in scope

Tests were run jointly by a participant organization (providing the FUT) and the Plugtests team (providing the experimental test system). The test sessions for the API track were scheduled upon request in parallel with the interoperability test sessions.

Following the prerequisites defined above, three test configurations were defined and shared with participants. Each configuration involved the test system – playing the role of the API consumer – and the FUT in the role of the API producer. See details in Clause 9.

Test execution and test results where operated and recorded manually by the Plugtests Team. Before and during the API test session the set of predefined tests were compiled to accommodate different participants' targets. The resulting test suites (in form of "Postman Collections") were made available in the Plugtests wiki.

8 IOP Test Plan Overview

8.1 Introduction

This 3rd NFV Plugtests Test Plan was developed by the NFV Plugtests team lead by ETSI Centre for Testing and Interoperability and following the interoperability testing methodology defined by ETSI NFV in [TST002] and building on the test plans of previous NFV Plugtests [1NFVPLU-TP] and [2NFVPLU-TP].

The following clauses summarise the interoperability test cases in scope for this Plugtests, and how they were grouped to optimise test session scheduling, duration and results collection and analysis. The five main interoperability test groups identified for the 3rd NFV Plugtests were: Single Vendor NS (covering also Single Vendor Scale To Level sub-group), Multi-Vendor NS (covering also Multi-Vendor NS with EPA and Multi-Vendor NS with SFC sub-groups), Multi-Site, Specific VNFM, Automatic LCM validation.

8.2 Single Vendor NS

8.2.1 Test Configuration

The Single Vendor NS group makes use of the SUT_SINGLE-VENDOR_NS configuration as described in the Test Plan [3NFVPLU-TP].

It involves one MANO solution, one VIM&NFVI and one VNF. The Test VNF used to verify proper NS deployment and functionality was considered as optional and expected to be provided by the VNF provider. Participants were encouraged to schedule and run this test group remotely (i.e. before the face-to-face event) as a mean to carry out pre-testing sessions to prepare for more complex sessions on-site.

This Single Vendor NS test configuration was also used for the purpose of verifying and testing the NS and VNF scaling to level operations, as a kind of enhanced interoperability tests cases with respect to the more generic scale in and scale out ones.

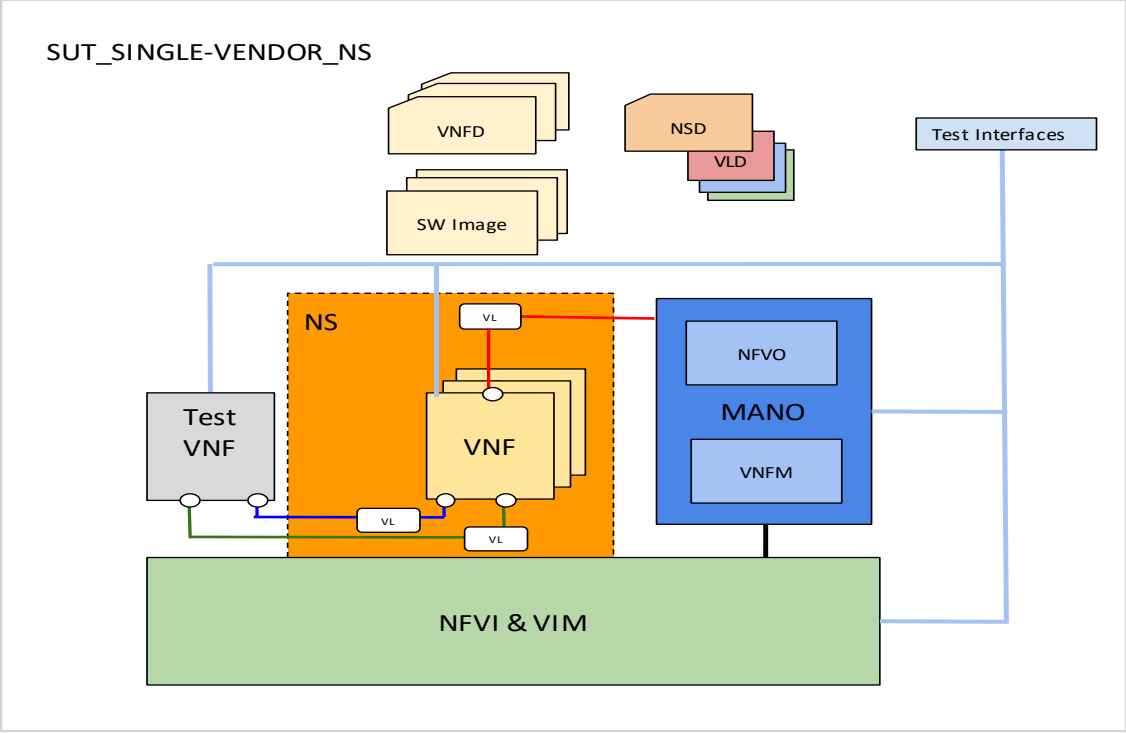

Figure 13: Single Vendor NS SUT Configuration

8.2.2 Test Cases

8.2.2.1 Base

Test Id	Test Purpose
TD_NFV_SINGLEVENDOR_ONBOARD_VNF_PKG_001	To on-board a VNF Package
TD_NFV_SINGLEVENDOR_ONBOARD_NSD_001	To onboard a NSD
TD_NFV_SINGLEVENDOR_NS_LCM_INSTANTIATE_001	To verify that an NS can be successfully instantiated
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_OUT_001	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered by a MANO operator
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_IN_001	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered by a MANO operator
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_OUT_002a	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered automatically in MANO by a VNF Indicator notification
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_IN_002a	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered automatically in MANO by a VNF Indicator notification
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_OUT_002b	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered automatically in MANO by querying a VNF Indicator
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_IN_002b	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered automatically in MANO by querying a VNF Indicator
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_OUT_003	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered automatically in MANO by a VIM KPI
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_IN_003	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered automatically in MANO by a VIM KPI
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_OUT_004	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered in MANO by a VNF/EM request
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_IN_004	To verify that a NS can successfully scale in (by removing VNF instances) if triggered in MANO by a VNF/EM request
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_OUT_VNF_001	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_IN_VNF_001	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_OUT_VNF_002a	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered automatically in MANO by a VNF Indicator notification

TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_IN_VNF_002a	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered automatically in MANO by a VNF Indicator notification
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_OUT_VNF_002b	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered automatically in MANO by querying a VNF Indicator
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_IN_VNF_002b	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered automatically in MANO by querying a VNF Indicator
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_OUT_VNF_003	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered automatically in MANO by a VIM KPI
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_IN_VNF_003	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered automatically in MANO by a VIM KPI
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_OUT_VNF_004	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered in MANO by a VNF/EM request
TD_NFV_SINGLEVENDOR_NS_LCM_SCALE_IN_VNF_004	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered in MANO by a VNF/EM request
TD_NFV_SINGLEVENDOR_NS_LCM_UPDATE_STOP_VNF_001	To verify that a VNF running in a NS can be successfully stopped by MANO
TD_NFV_SINGLEVENDOR_NS_LCM_UPDATE_START_VNF_001	To verify that a stopped VNF in a NS can be successfully re-started by MANO
TD_NFV_SINGLEVENDOR_FM_VNF_ALARM_001	Verify that a VNF fault alarm event is detected by the MANO when a fault occurs on a VNF part of a NS
TD_NFV_SINGLEVENDOR_FM_VNF_CLEAR_001	Verify that a VNF fault alarm clearance event is detected by the MANO when a fault is cleared on a VNF part of a NS by resolving the causing problem
TD_NFV_SINGLEVENDOR_PM_VR_CREATE_MONITOR_001	To verify that performance metrics of one or more virtualised resources that are allocated to a NS instance can be monitored
TD_NFV_SINGLEVENDOR_PM_VR_CREATE_THRESHOLD_001	To verify that performance metrics of one or more virtualised resources that are allocated to a NS instance can be monitored using thresholds based notifications
TD_NFV_SINGLEVENDOR_PM_VR_DELETE_MONITOR_001	To verify that monitoring of performance metrics of one or more virtualised resources that are allocated to a NS instance can be stopped
TD_NFV_SINGLEVENDOR_PM_VR_DELETE_THRESHOLD_001	To verify that performance monitoring thresholds created for one or more virtualised resources that are allocated to a NS instance can be deleted
TD_NFV_SINGLEVENDOR_PM_VNF_KPI_CREATE_MONITOR_001	To verify that a VNF indicator related to a NS instance can be monitored
TD_NFV_SINGLEVENDOR_PM_VNF_KPI_DELETE_MONITOR_001	To verify that monitoring of a VNF indicator related to a NS instance can be stopped

TD_NFV_SINGLEVENDOR_PM_VNF_KPI_CREATE_THRESHOLD_001	To verify that a VNF indicator related to a NS instance can be monitored using thresholds based notifications
TD_NFV_SINGLEVENDOR_PM_VNF_KPI_DELETE_THRESHOLD_001	To verify that performance monitoring thresholds created for one or more VNF indicator related to a NS instance can be deleted
TD_NFV_SINGLEVENDOR_NS_LCM_TERMINATE_001	To verify that a NS can be successfully terminated
TD_NFV_ SINGLEVENDOR_TEARDOWN_DELETE_NSD_001	To delete a NSD
TD_NFV_SINGLEVENDOR_TEARDOWN_DELETE_VNF_PKG_001	To delete a VNF Package

Table 11: Single Vendor NS: Test Cases (Base)

8.2.2.2 Scale to Level

Test Id	Test Purpose
TD_NFV_SCALE-LEVEL_ONBOARD_VNF_PKG_001	To on-board a VNF Package
TD_NFV_ SCALE-LEVEL_ONBOARD_NSD_001	To onboard a NSD
TD_NFV_SCALE-LEVEL_NS_LCM_INSTANTIATE_001	To verify that an NS can be successfully instantiated
TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_001	Verify that an NS can be successfully scaled to another existing instantiation level by changing the number of VNF instances when triggered by a MANO operator
TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_002	Verify that an NS can be successfully scaled to another existing instantiation level by changing the number of VNF instances when triggered automatically by a VNF indicator
TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_003	Verify that an NS can be successfully scaled to another existing instantiation level by changing the number of VNF instances when triggered automatically by a VIM KPI
TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_VNF_001	Verify that a VNF in a NS can be successfully scaled to another existing instantiation level by changing the number of VNFC instances when triggered by a MANO operator
TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_VNF_003	Verify that a VNF in a NS can be successfully scaled to another existing instantiation level by changing the number of VNFC instances when triggered automatically by a VIM KPI
TD_NFV_SCALE-LEVEL_NS_LCM_TERMINATE_001	To verify that a NS can be successfully terminated
TD_NFV_SCALE-LEVEL_TEARDOWN_DELETE_NSD_001	To delete a NSD
TD_NFV_SCALE-LEVEL_TEARDOWN_DELETE_VNF_PKG_001	To delete a VNF Package

Table 12xx: Single Vendor NS: Test Cases (Scale to level)

8.3 Multi-Vendor NS

8.3.1 Test configuration

The Multi-Vendor NS group leverages the SUT_MULTI-VENDOR_NS configuration as described in the Test Plan [3NFVPLU-TP].

It involves one MANO solution, one VIM&NFV and at least two VNF from 2 different providers. This group was mandatory for the 3rd NFV Plugtests participants and aimed at testing multi-vendor Network Services in the wider set of combinations possible.

With this SUT configuration other two interoperability test groups have been validated during the 3rd NFV Plugtests. First, the Multi-Vendor NS with EPA that included test cases (optional for Plugtests participants) to validate the interoperability in the case of EPA supported by the MANO the VIM&NFVI and at least one of the VNFs. As second additional interoperability test group, the Multi-Vendor NS with SFC also leveraged on this test configuration, to validate the usage of SFC based on the NSH protocol, which had to be supported at least by the VIM&NFVI and at least one of the VNFs.

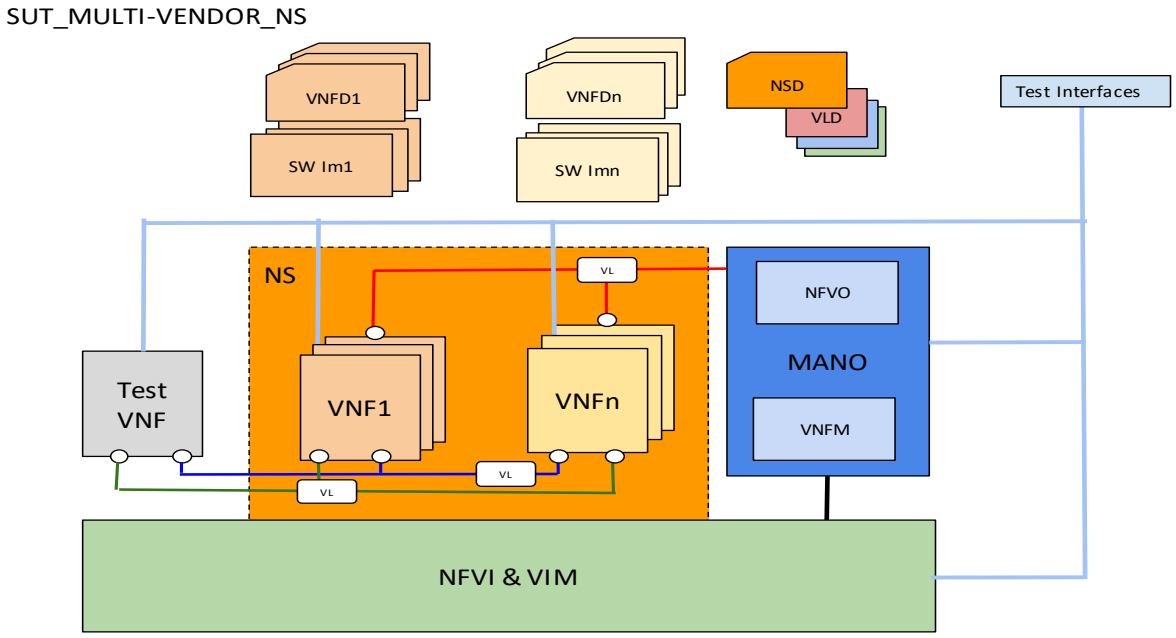

Figure 14: Multi-Vendor NS SUT Configuration

8.3.1 Test Cases

8.2.2.1 Base

Test Id	Test Purpose
TD_NFV_MULTIVENDOR_ONBOARD_VNF_PKG_001	To on-board a VNF Package
TD_NFV_MULTIVENDOR_ONBOARD_NSD_001	To onboard a NSD
TD_NFV_MULTIVENDOR_NS_LCM_INSTANTIATE_001	To verify that an NS can be successfully instantiated
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_001	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered by a MANO operator
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_001	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered by a MANO operator

TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_002a	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered automatically in MANO by a VNF Indicator notification
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_002a	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered automatically in MANO by a VNF Indicator notification
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_002b	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered automatically in MANO by querying a VNF Indicator
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_002b	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered automatically in MANO by querying a VNF Indicator
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_003	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered automatically in MANO by a VIM KPI
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_003	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered automatically in MANO by a VIM KPI
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_004	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered in MANO by a VNF/EM request
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_004	To verify that a NS can successfully scale in (by removing VNF instances) if triggered in MANO by a VNF/EM request
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_VNF_001	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_VNF_001	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_VNF_002a	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered automatically in MANO by a VNF Indicator notification
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_VNF_002a	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered automatically in MANO by a VNF Indicator notification
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_VNF_002b	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered automatically in MANO by querying a VNF Indicator
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_VNF_002b	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered automatically in MANO by querying a VNF Indicator
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_VNF_003	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered automatically in MANO by a VIM KPI
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_VNF_003	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered automatically in MANO by a VIM KPI
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_VNF_004	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC

	instances (VMs)) when triggered in MANO by a VNF/EM request
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_VNF_004	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered in MANO by a VNF/EM request
TD_NFV_MULTIVENDOR_NS_LCM_UPDATE_STOP_VNF_001	To verify that a VNF running in a NS can be successfully stopped by MANO
TD_NFV_MULTIVENDOR_NS_LCM_UPDATE_START_VNF_001	To verify that a stopped VNF in a NS can be successfully re-started by MANO
TD_NFV_MULTIVENDOR_FM_VR_ALARM_001	Verify that a fault alarm event propagates to the MANO when a virtualised resource that is required for the NS fails.
TD_NFV_MULTIVENDOR_FM_VR_CLEAR_001	Verify that a fault clearance event propagates to the MANO when a failed virtualised resource that is required for the NS is recovered
TD_NFV_MULTIVENDOR_PM_VR_CREATE_MONITOR_001	To verify that performance metrics of one or more virtualised resources that are allocated to a NS instance can be monitored
TD_NFV_MULTIVENDOR_PM_VR_CREATE_THRESHOLD_001	To verify that performance metrics of one or more virtualised resources that are allocated to a NS instance can be monitored using thresholds based notifications
TD_NFV_MULTIVENDOR_PM_VR_DELETE_MONITOR_001	To verify that monitoring of performance metrics of one or more virtualised resources that are allocated to a NS instance can be stopped
TD_NFV_MULTIVENDOR_PM_VR_DELETE_THRESHOLD_001	To verify that performance monitoring thresholds created for one or more virtualised resources that are allocated to a NS instance can be deleted
TD_NFV_MULTIVENDOR_PM_VNF_KPI_CREATE_MONITOR_001	To verify that a VNF indicator related to a NS instance can be monitored
TD_NFV_MULTIVENDOR_PM_VNF_KPI_DELETE_MONITOR_001	To verify that monitoring of a VNF indicator related to a NS instance can be stopped
TD_NFV_MULTIVENDOR_PM_VNF_KPI_CREATE_THRESHOLD_001	To verify that a VNF indicator related to a NS instance can be monitored using thresholds based notifications
TD_NFV_MULTIVENDOR_PM_VNF_KPI_DELETE_THRESHOLD_001	To verify that performance monitoring thresholds created for one or more VNF indicator related to a NS instance can be deleted
TD_NFV_MULTIVENDOR_NS_LCM_TERMINATE_001	To verify that a NS can be successfully terminated
TD_NFV_ MULTIVENDOR_TEARDOWN_DELETE_NSD_001	To delete a NSD
TD_NFV_MULTIVENDOR_TEARDOWN_DELETE_VNF_PKG_001	To delete a VNF Package

Table 13: Multi-Vendor NS Test Cases (Base)

8.3.2.1 EPA – Enhanced Platform Awareness

Test Id	Test Purpose
TD_NFV_EPA_ONBOARD_VNF_PKG_001	To on-board a VNF Package
TD_NFV_EPA_ONBOARD_NSD_001	To onboard a NSD

TD_NFV_EPA_NS_LCM_INSTANTIATE_001	To verify that an NS can be successfully instantiated with EPA requirements
TD_NFV_EPA_NS_LCM_SCALE_OUT_001	To verify that a NS can be successfully scaled out with EPA requirements (by adding VNF instances) if triggered by a MANO operator
TD_NFV_EPA_NS_LCM_SCALE_IN_001	To verify that a NS can be successfully scaled in with EPA requirements (by removing VNF instances) if triggered by a MANO operator
TD_NFV_EPA_NS_LCM_SCALE_OUT_VNF_001	To verify that a VNF in a NS can be successfully scaled out with EPA requirements (by adding VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_EPA_NS_LCM_SCALE_IN_VNF_001	To verify that a VNF in a NS can be successfully scaled in with EPA requirements (by removing VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_EPA_NS_LCM_TERMINATE_001	To verify that a NS can be successfully terminated
TD_NFV_ EPA_TEARDOWN_DELETE_NSD_001	To delete a NSD
TD_NFV_EPA_TEARDOWN_DELETE_VNF_PKG_001	To delete a VNF Package

Table 14: Multi-Vendor NS Test Cases (EPA)

8.3.2.1 SFC – Service Function Chaining

Test Id	Test Purpose
TD_NFV_SFC_ONBOARD_VNF_PKG_001	To on-board a VNF Package
TD_NFV_SFC_ONBOARD_NSD_001	To onboard a NSD
TD_NFV_SFC_NS_LCM_INSTANTIATE_001	To verify that an NS with NSH based SFC can be successfully instantiated
TD_NFV_SFC_NS_LCM_TERMINATE_001	To verify that a NS can be successfully terminated
TD_NFV_SFC_TEARDOWN_DELETE_NSD_001	To delete a NSD
TD_NFV_SFC_TEARDOWN_DELETE_VNF_PKG_001	To delete a VNF Package

Table15: : Multi-Vendor NS Test Cases (SFC)

8.4 Multi Site

8.4.1 Test configuration

The multi-site group leverages the SUT_MULTI-SITE configuration as described in the Test Plan [3NFVPLU-TP]. It involves one MANO solution, at least two different VIM&NFVIs and at least two VNF.

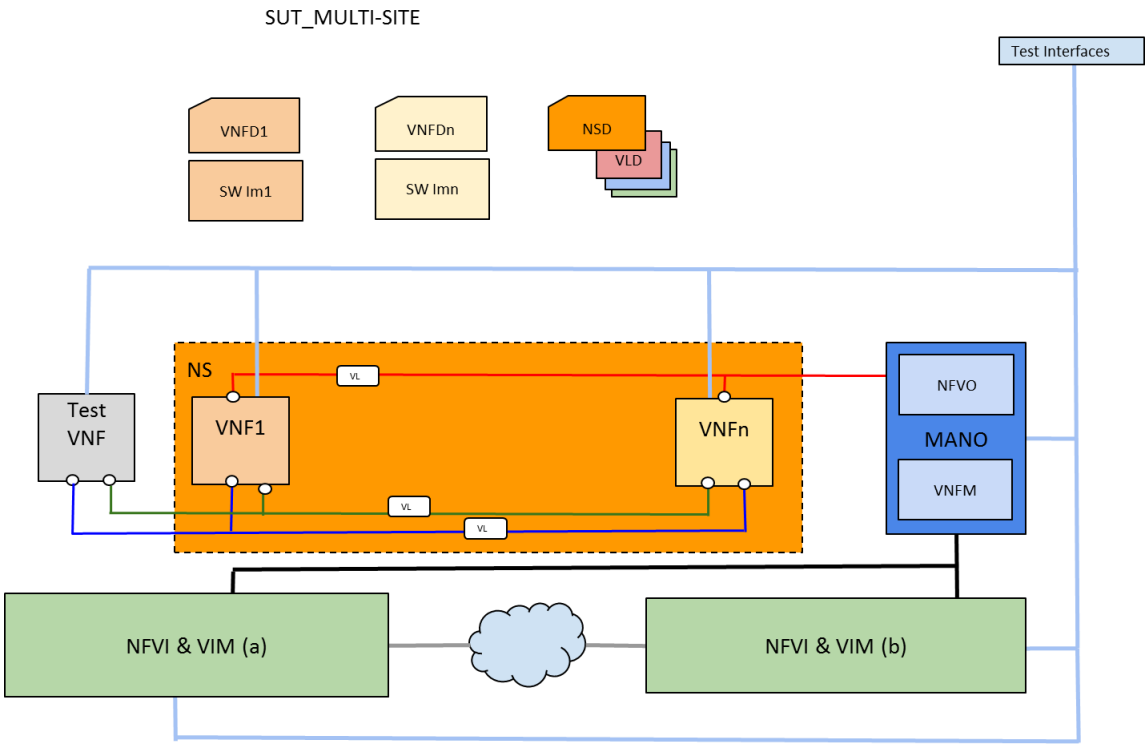

Figure 15: Multi-SITE SUT Configuration

8.4.2 Test cases

Test Id	Test Purpose
TD_NFV_MULTISITE_ONBOARD_VNF_PKG_001	To on-board a VNF Package
TD_NFV_MULTISITE_ONBOARD_NSD_001	To on-board a Multi-Site NSD
TD_NFV_MULTISITE_NS_LCM_INSTANTIATE_001	To verify that an NS can be successfully instantiated across different sites
TD_NFV_MULTISITE_NS_LCM _SCALE_OUT_001	To verify that a multi-site NS can be successfully scaled out (by adding VNF instances) if triggered by a MANO operator
TD_NFV_MULTISITE_NS_LCM_SCALE_IN_001	To verify that a multi-site NS can be successfully scaled in (by removing VNF instances) if triggered by a MANO operator
TD_NFV_MULTISITE_NS_LCM_SCALE_OUT_VNF_001	To verify that a VNF in a multi-site NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_MULTISITE_NS_LCM_SCALE_IN_VNF_001	To verify that a VNF in a multi-site NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_MULTISITE_FM_VR_ALARM_001	Verify that a fault alarm event propagates to the MANO when a virtualised resource that is required for the NS fails.
TD_NFV_MULTISITE_FM_VR_CLEAR_001	Verify that a fault clearance event propagates to the MANO when a failed virtualised resource that is required for the NS is recovered
TD_NFV_MULTISITE_FM_VNF_ALARM_001	Verify that a VNF fault alarm event is detected by the MANO when a fault occurs on a VNF part of a NS

TD_NFV_MULTISITE_FM_VNF_CLEAR_001	Verify that a VNF fault alarm clearance event is detected by the MANO when a fault is cleared on a VNF part of a NS by resolving the causing problem
TD_NFV_MULTISITE_PM_VR_CREATE_MONITOR_001	To verify that performance metrics of one or more virtualised resources that are allocated to a NS instance can be monitored
TD_NFV_MULTISITE_PM_VR_CREATE_THRESHOLD_001	To verify that performance metrics of one or more virtualised resources that are allocated to a NS instance can be monitored using thresholds based notifications
TD_NFV_MULTISITE_PM_VR_DELETE_MONITOR_001	To verify that monitoring of performance metrics of one or more virtualised resources that are allocated to a NS instance can be stopped
TD_NFV_MULTISITE_PM_VR_DELETE_THRESHOLD_001	To verify that performance monitoring thresholds created for one or more virtualised resources that are allocated to a NS instance can be deleted
TD_NFV_MULTISITE_PM_VNF_KPI_CREATE_MONITOR_001	To verify that a VNF indicator related to a NS instance can be monitored
TD_NFV_MULTISITE_PM_VNF_KPI_DELETE_MONITOR_001	To verify that monitoring of a VNF indicator related to a NS instance can be stopped
TD_NFV_MULTISITE_PM_VNF_KPI_CREATE_THRESHOLD_001	To verify that a VNF indicator related to a NS instance can be monitored using thresholds based notifications
TD_NFV_MULTISITE_PM_VNF_KPI_DELETE_THRESHOLD_001	To verify that performance monitoring thresholds created for one or more VNF indicator related to a NS instance can be deleted
TD_NFV_MULTISITE_NS_LCM_TERMINATE_001	To verify that a Multi-Site NS can be successfully terminated
TD_NFV_MULTISITE_TEARDOWN_DELETE_NSD_001	To delete a NSD
TD_NFV_MULTISITE_TEARDOWN_DELETE_VNF_PKG_001	To delete a VNF Package

Table 16: Multi-Site Test Cases

8.5 Specific VNFM

8.5.1 S-VNFM-D

8.5.1.1 Test Configurations

The S-VNFM-D group leverages the SUT_S-VNFM-D configuration as described in the Test Plan [3NFVPLU-TP].

This configuration involved one MANO solution, one VIM&NFVI and one VNF providing its own VNF Manager. The Specific VNFM and the MANO solutions were requested to both support the direct resource management mode.

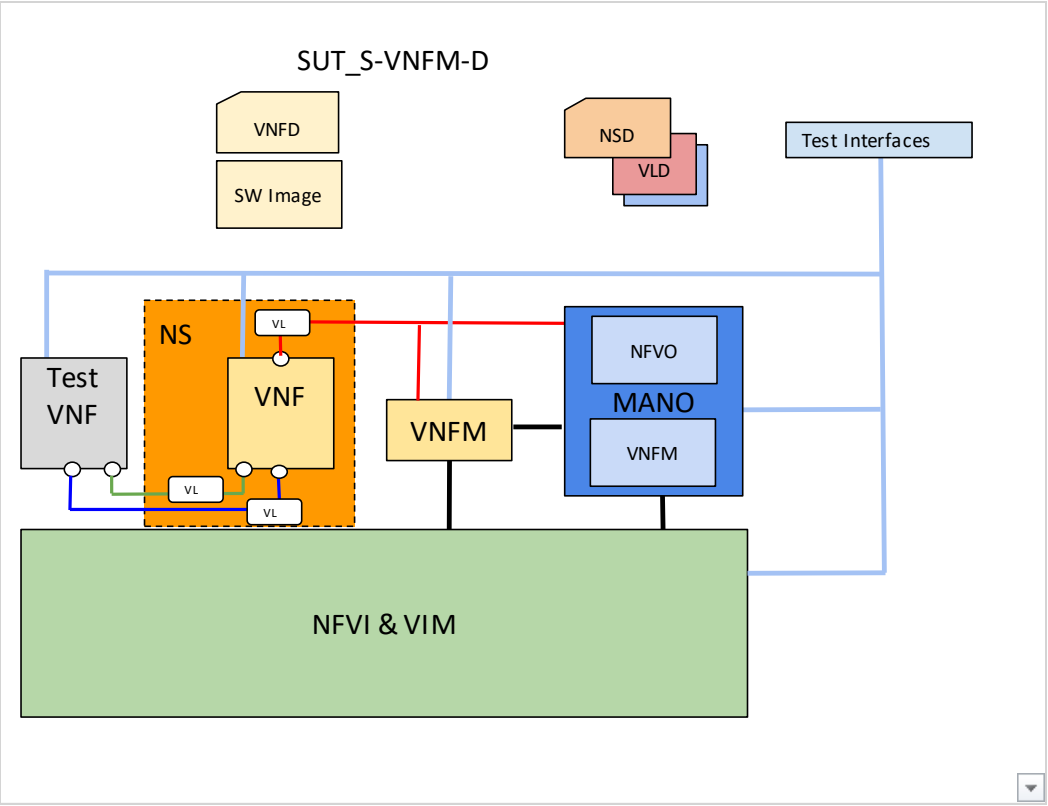

Figure 16: S-VNFM-D SUT Configuration

8.5.1.2 Test Cases

Test Id	Test Purpose
TD_NFV_S-VNFM-D_ONBOARD_VNF_PKG_001	To on-board a VNF Package
TD_NFV_S-VNFM-D_ONBOARD_NSD_001	To on-board a NSD
TD_NFV_S-VNFM-D_NS_LCM_INSTANTIATE_001	To verify that an NS can be successfully instantiated
TD_NFV_S-VNFM-D_NS_LCM_SCALE_OUT_001	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered by a MANO operator
TD_NFV_S-VNFM-D_NS_LCM_SCALE_IN_001	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered by a MANO operator
TD_NFV_S-VNFM-D_NS_LCM_SCALE_OUT_VNF_001	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_S-VNFM-D_NS_LCM_SCALE_IN_VNF_001	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_S-VNFM-D_NS_LCM_SCALE_OUT_VNF_001	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_S-VNFM-D_NS_LCM_SCALE_IN_VNF_001	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_S-VNFM-D_PM_VNF_VR_CREATE_MONITOR_001	To verify that the performance metrics of a virtualised resource that is allocated to a VNF instance inside a NS instance can be monitored through external VNFM

TD_NFV_S-VNFM-I_PM_VNF_VR_CREATE_THRESHOLD_001	To verify that the performance metrics of a virtualised resource that is allocated to a VNF instance inside a NS instance can be monitored using thresholds based notifications through external VNFM
TD_NFV_S-VNFM-D_PM_VNF_VR_DELETE_MONITOR_001	To verify that the monitoring of performance metrics of a virtualised resource that is allocated to a VNF instance inside a NS instance can be stopped through external VNFM
TD_NFV_S-VNFM-D_PM_VNF_VR_DELETE_THRESHOLD_001	To verify that a performance monitoring threshold created for a virtualised resource that is allocated to a VNF instance inside a NS instance can be deleted through external VNFM
TD_NFV_S-VNFM-D_PM_VNF_KPI_CREATE_MONITOR_001	To verify that a VNF indicator related to a NS instance can be monitored through external VNFM
TD_NFV_S-VNFM-D_PM_VNF_KPI_DELETE_MONITOR_001	To verify that monitoring of a VNF indicator related to a NS instance can be stopped through external VNFM
TD_NFV_S-VNFM-D_PM_VNF_KPI_CREATE_THRESHOLD_001	To verify that a VNF indicator related to a NS instance can be monitored using thresholds based notifications through external VNFM
TD_NFV_S-VNFM-D_PM_VNF_KPI_DELETE_THRESHOLD_001	To verify that performance monitoring thresholds created for one or more VNF indicator related to a NS instance can be deleted through external VNFM
TD_NFV_S-VNFM-D_FM_VNF_VR_ALARM_001	Verify that a VNF fault alarm notification propagates via the specific VNFM to the MANO when a VNF fault is triggered by a failed virtualised resource
TD_NFV_S-VNFM-D_FM_VNF_VR_CLEAR_001	Verify that a VNF fault alarm clearance notification propagates via the specific VNFM to the MANO when a VNF fault is cleared by resolving the causing problem on the failed virtualised resource
TD_NFV_S-VNFM-D_FM_VNF_ALARM_001	Verify that a VNF fault alarm notification propagates via the VNFM to the MANO when a fault occurs on a VNF part of a NS
TD_NFV_S-VNFM-D_FM_VNF_CLEAR_001	Verify that a VNF fault alarm clearance notification propagates via the VNFM to the MANO when a fault is cleared on a VNF part of a NS by resolving the causing problem
TD_NFV_VNFM-D_NS_LCM_TERMINATE_001	To verify that a NS can be successfully terminated
TD_NFV_VNFM-D_TEARDOWN_DELETE_NSD_001	To delete a NSD
TD_NFV_VNFM-D_TEARDOWN_DELETE_VNF_PKG_001	To delete a VNF Package

Table 17: S-VNFM-D Test Cases

8.5.2 S-VNFM-I

8.5.2.1 Test Configuration

The S-VNFM-I group leverages the SUT_S-VNFM-I configuration as described in the Test Plan [3NFVPLU-TP].

This configuration involved one MANO solution, one VIM&NFVI and one VNF providing its own VNF Manager. The Specific VNFM and the MANO solutions were requested to both support the indirect resource management mode

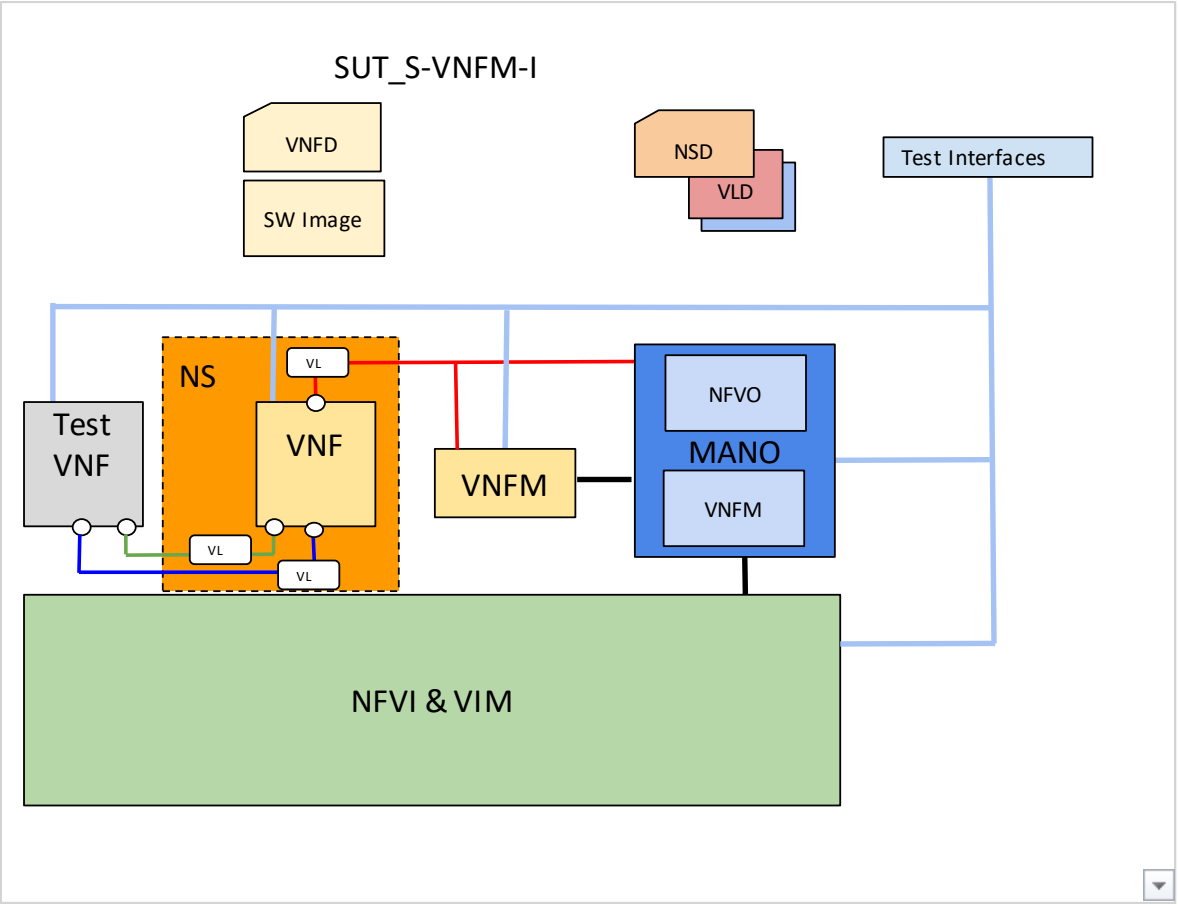

Figure 17: S-VNFM-I SUT Configuration

8.5.2.2 Test Cases

Test Id	Test Purpose
TD_NFV_S-VNFM-I_ONBOARD_VNF_PKG_001	To on-board a VNF Package
TD_NFV_S-VNFM-I_ONBOARD_NSD_001	To on-board a NSD
TD_NFV_S-VNFM-I_NS_LCM_INSTANTIATE_001	To verify that an NS can be successfully instantiated
TD_NFV_S-VNFM-I_NS_LCM_SCALE_OUT_001	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered by a MANO operator
TD_NFV_S-VNFM-I_NS_LCM_SCALE_IN_001	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered by a MANO operator
TD_NFV_S-VNFM-I_NS_LCM_SCALE_OUT_VNF_001	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_S-VNFM-I_NS_LCM_SCALE_IN_VNF_001	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_S-VNFM-I_PM_VR_CREATE_MONITOR_001	To verify that performance metrics of one or more virtualised resources that are allocated to a NS instance can be monitored
TD_NFV_S-VNFM-I_PM_VR_CREATE_THRESHOLD_001	To verify that performance metrics of one or more virtualised resources that are allocated to a NS instance can be monitored using thresholds based notifications
TD_NFV_S-VNFM-I_PM_VR_DELETE_MONITOR_001	To verify that monitoring of performance metrics of one or more virtualised resources that are allocated to a NS

	instance can be stopped
TD_NFV_S-VNFM-I_PM_VR_DELETE_THRESHOLD_001	To verify that performance monitoring thresholds created for one or more virtualised resources that are allocated to a NS instance can be deleted
TD_NFV_S-VNFM-I_PM_VNF_KPI_CREATE_MONITOR_001	To verify that a VNF indicator related to a NS instance can be monitored through external VNFM
TD_NFV_S-VNFM-I_PM_VNF_KPI_DELETE_MONITOR_001	To verify that monitoring of a VNF indicator related to a NS instance can be stopped through external VNFM
TD_NFV_S-VNFM-I_PM_VNF_KPI_CREATE_THRESHOLD_001	To verify that a VNF indicator related to a NS instance can be monitored using thresholds based notifications through external VNFM
TD_NFV_S-VNFM-I_PM_VNF_KPI_DELETE_THRESHOLD_001	To verify that performance monitoring thresholds created for one or more VNF indicator related to a NS instance can be deleted through external VNFM
TD_NFV_S-VNFM-I_FM_VR_ALARM_001	Verify that a fault alarm event propagates to the MANO when a virtualised resource that is required for the NS fails.
TD_NFV_S-VNFM-I_FM_VR_CLEAR_001	Verify that a fault clearance event propagates to the MANO when a failed virtualised resource that is required for the NS is recovered
TD_NFV_VNFM-I_NS_LCM_TERMINATE_001	To verify that a NS can be successfully terminated
TD_NFV_VNFM-I_TEARDOWN_DELETE_NSD_001	To delete a NSD
TD_NFV_VNFM-I_TEARDOWN_DELETE_VNF_PKG_001	To delete a VNF Package

Table 18: S-NFM-I Test Cases

8.6 Auto LCM Validation

8.6.1 Test Configuration

The Auto LCM Validation group leverages the SUT_AUTO-LCM-VALIDATION configuration as described in the Test Plan [3NFVPLU-TP].

It involves one MANO solution, one VIM&NFVI and at least two VNFs. The Test System is in charge of automating the triggers and IOP checks as described in the Test Plan

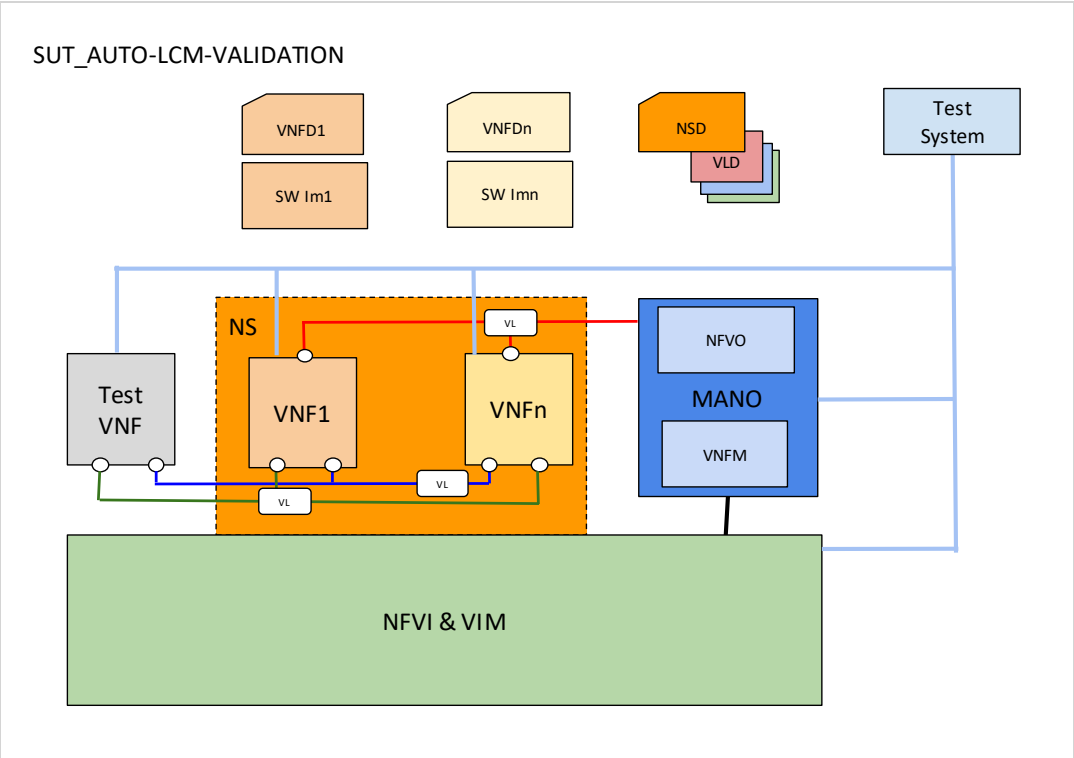

Figure 18: AUTO-LCM-VALIDATION SUT Configuration

8.6.2 Test Cases

Test Id	Test Purpose
TD_NFV_AUTOLCMV_ONBOARD_NSD_001	To on-board a NSD
TD_NFV_AUTOLCMV_NS_LCM_INSTANTIATE_001	To verify that an NS can be successfully instantiated
TD_NFV_AUTOLCMV_NS_LCM_SCALE_OUT_001	To verify that a NS can be successfully scaled out (by adding VNF instances) if triggered by a MANO operator
TD_NFV_AUTOLCMV_NS_LCM_SCALE_IN_001	To verify that a NS can be successfully scaled in (by removing VNF instances) if triggered by a MANO operator
TD_NFV_AUTOLCMV_NS_LCM_SCALE_OUT_VNF_001	To verify that a VNF in a NS can be successfully scaled out (by adding VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_AUTOLCMV_NS_LCM_SCALE_IN_VNF_001	To verify that a VNF in a NS can be successfully scaled in (by removing VNFC instances (VMs)) when triggered by a MANO operator
TD_NFV_AUTOLCMV_NS_LCM_UPDATE_STOP_VNF_001	To verify that a VNF running in a NS can be successfully stopped by MANO
TD_NFV_AUTOLCMV_NS_LCM_UPDATE_START_VNF_001	To verify that a stopped VNF in a NS can be successfully re-started by MANO
TD_NFV_AUTOLCMV_FM_VR_ALARM_001	Verify that a fault alarm event propagates to the MANO when a virtualised resource that is required for the NS fails.
TD_NFV_AUTOLCMV_FM_VR_CLEAR_001	Verify that a fault clearance event propagates to the MANO when a failed virtualised resource that is required for the NS is recovered

TD_NFV_AUTOLCMV_NS_LCM_TERMINATE_001	To verify that a NS can be successfully terminated
TD_NFV_AUTOLCMV_TEARDOWN_DELETE_NSD_001	To delete a NSD

Table 19: Auto LCM Validation Test Cases

9 API Test Plan Overview

9.1 Introduction

The test plan was created according to the APIs and test cases that the participants required. Out of the total 17 APIs specified by ETSI NFV (of which 14 were available in a machine-readable format) 4 APIs were tested during the Plugtests.

For each API in the test plan, a subset of supported operations was used to create a total of 29 test cases. Each test case focuses on the test of one operation. The verdict of the test is “pass” if the response is complying with NFV SOL specifications, in terms of:

- Response code
- Response headers
- Response body

The notification system provided in several interfaces was informally tested in correspondence of the requests and responses. The outcome of the notification was noted as a comment in the test reports.

The test system provided a CallbackURI for notifications capable of logging all messages and their bodies, for debugging and off-line analysis. During the execution of the tests the entire sessions were recorded in a Pcap file to enable further analysis.

The tests configurations for the API track generally include the following components:

- The Function Under Test, i.e. the API producer, depicted in Yellow
- The Test environment (TEST ENV), i.e. the set of components required to support the correct execution of the FUT
- The Test Systems, composed of
    - The Postman scripts in the role of the API consumer, interoperating with the FUT using the API under test and
    - The Notification Endpoint, the reference of which was communicated to the FUT as CallbackUri for Notifications.

9.2 SOL002 – VNF/EM

9.2.1 Test Configuration

The test configuration as described in the figure below was defined to test the interfaces exposed by a VNF/EM towards the VNFM, such as VNF Configuration API and VNF Indicator API. The test system acts as the VNFM.

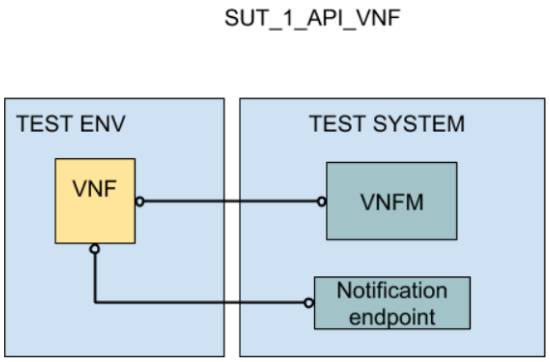

Figure 19: Test configuration SUT_1_API_VNF

9.2.2 Test Cases

9.5.2.1 VNF Configuration API (Ve-Vnfm)

TD ID	Operation	Resource	Method
TD_API_20	Read VNF/VNFC configuration from VNF.	/configuration	GET
TD_API_21	Modify VNF/VNFC configuration	/configuration	PATCH
TD_API_22	Modify VNF/VNFC configuration with partial JSON	/configuration	PATCH

Table 20. VNF Configuration API TCs

9.5.2.2 VNF Indicator API (Ve-Vnfm)

TD ID	Operation	Resource	Method
TD_API_23	Query Multiple indicators	/vnfind/v1/indicators	GET
TD_API_24	Query multiple indicators related to a VNF instance.	/indicators/{Id}	GET
TD_API_25	Read individual indicator related to a VNF instance.	/indicators/{Id}/{Id}	GET
TD_API_26	Subscribe	/subscriptions	POST
TD_API_27	Query all subscriptions	/subscriptions	GET
TD_API_28	Read individual subscription	/subscriptions/{Id}	GET
TD_API_29	Terminate Subscription	/subscriptions/{Id}	DELETE

Table 21. VNF Indicator API TCs

9.3 SOL003 - VNFM

9.3.1 Test Configuration

The test configuration as described below was defined to test the interfaces exposed by a VNFM towards the NFVO such as VNF Fault Management API, VNF Indicator API, VNF Lifecycle Management API, and VNF Performance Management API. The test system is acting as the NFVO.

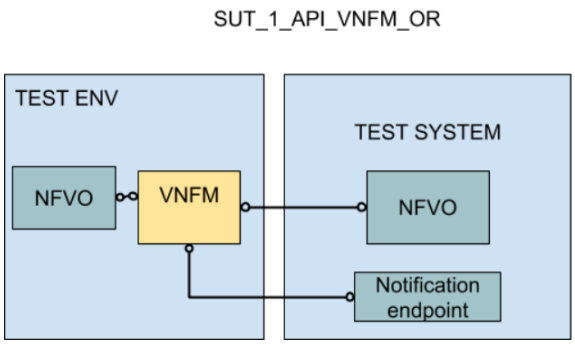

Figure 20: Test configuration SUT_1_API_VNFM_OR

9.3.2 Test Cases

9.3.2.1 VNF Lifecycle Management API (Or-Vnfm)

TD ID	Operation	Resource	Method
TD_API_1	Create VNF Identifier	/vnf_instances	POST
TD_API_2	Query all VNF Identifiers	/vnf_instances	GET
TD_API_3	Query VNF	/vnf_instances/{Id}	GET
TD_API_4	Terminate VNF	/vnf_instances/{Id}/terminate	POST
TD_API_5	Patch VNF Identifier	/vnf_instances/{Id}	PATCH
TD_API_6	Delete VNF Identifier	/vnf_instances/{Id}	DELETE
TD_API_7	Instantiate VNF	/vnf_instances/{Id}/instantiate	GET
TD_API_8	Get Operation Status	/vnf_lcm_op_occs/{Id}	POST
TD_API_9	Subscribe	/subscriptions	POST
TD_API_10	Query all subscriptions	/subscriptions	GET
TD_API_11	Query single subscription	/subscriptions/{Id}	GET
TD_API_12	Terminate Subscription	/subscriptions/{Id}	DELETE

TD_API_30	Scale	/vnf_instances/{Id}/scale	POST
TD_API_31	Scale to level	/vnf_instances/{Id}/scale_to_level	POST

Table 22. VNF Lifecycle Management TCs

9.4 SOL003 - NFVO

9.4.1 Test Configuration

The test configuration as described below was defined to test the interfaces exposed by NFVOs towards the VNFM such as Virtualised Resources Quota Available Notification API, VNF Lifecycle Operation Granting API and VNF Package Management. The test system is acting as the VNFM.

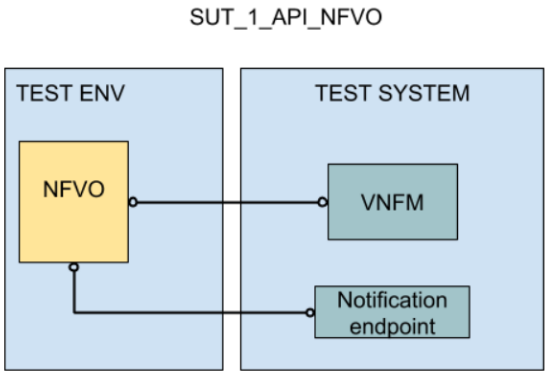

Figure 21: Test configuration SUT_1_API_NFVO

9.4.2 Test Cases

9.4.2.1 VNF Package Management API (Or-Vnfm)

TD ID	Operation	Resource	Method
TD_API_14	Query On-boarded VNF Package Information, including obtaining the VNFD	/vnf_packages	GET
TD_API_15	Fetch On-boarded VNF Package	/vnf_packages/{Id}/package_content	GET
TD_API_16	Fetch On-boarded VNF Package Artifacts	/vnf_packages/{Id}/artifacts/{Path}	GET
TD_API_17	Querying/reading on-boarded VNF package information (individual)	/vnf_packages/{Id}	GET
TD_API_18	Reading the VNFD of an on-boarded VNF package	/vnf_packages/{Id}/vnfd	GET

Table 23. VNF Package Management API TCs

9.4.2.2 VNF Lifecycle Operation Granting API (Or-Vnfm)

TD ID	Operation	Resource	Method
TD_API_19	Grant request with synchronous response	/grants	POST

Table 24. VNF Lifecycle Operation Granting API TCs

9.5 SOL005 - NFVO

9.5.1 Test Configuration

The test configuration as described below was defined to test the interfaces exposed by NFVOs towards the OSS such as the NSD Management API. The test system is acting as the OSS.

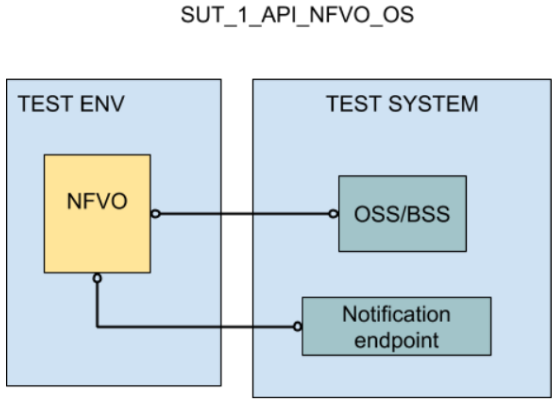

Figure 22: Test configuration SUT_1_API_NFVO

9.5.2 Test Cases

9.5.2.1 NSD Management API (Os-ma-nfvo)

TD ID	Operation	Resource	Method
TD_API_32	Query NSD descriptors	/ns_descriptors	GET
TD_API_33	Create NSD descriptor	/ns_descriptors	POST
TD_API_34	Read NSD descriptor	/ns_descriptors/{Id}	GET
TD_API_35	Update NSD descriptor	/ns_descriptors/{Id}	POST
TD_API_36	Upload NSD content	/ns_descriptors/{{Id}}/nsd_content	PUT
TD_API_37	Get NSD content	/ns_descriptors/{{Id}}/nsd_content	GET
TD_API_38	Delete NSD	/ns_descriptors/{Id}	DELETE

Table 25. NSD Management API TCs

10 Results

10.1 Overall Results

During the Plugtests, a total of 106 Test Sessions were run: 97 different combinations of the Functions Under Test (FUTs) in scope: VNFs, MANOs and VIM&NFVI were tested for interoperability and 9 FUTs participated to the API track on a number of different configurations addressing different features and APIs.

The following sections provide an overview of the reported results: overall, per test group, pert tests case. To facilitate the analysis, results are presented as follows:

Result	Meaning
OK	Test Case run. Interoperability (or API test) successfully achieved.
NO	Test Case run. Interoperability (or API test) not achieved.
NA	Not Applicable: Feature not supported by one or more Functions Under Test
Run	Total number of Test Cases Run = OK + NO
Total	Total number of Test Cases = OK + NO + NA = Run + Not Run

Table 26: 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 taken into account in the Total Results

The two tables below provides the overall results (aggregated data) for all the test cases run during the interoperability and API Track Test Sessions, from all participating companies:

		Interoperability (TCs Run)		TCs Not Run	TCs Totals	
Overall Results	Number of Test Sessions	OK	NO	NA	Run	Total
	97	736	95	636	831	1.467

Table 27a: IOP Overall Results

		API Validation (TCs Run)		TCs Not Run	TCs Totals	
Overall Results	Number of Test Sessions	OK	NO	NA	Run	Total
	9	58	25	28	83	111

Table 27b: API Track Overall Results

During each Test Session, depending on the targeted configuration and features to be tested, a different number of test cases were offered to the involved participants.

Overall, the test plans included 226 test cases, organised in different groups as described in chapter 8 (for interoperability) and chapter 9 (for API Track). Through the 106 Test Sessions run, a total of 1578 Test Results were reported. This figure includes both the executed and non-executed test cases. Overall, a total of 914 individual test cases were run and results reported for them.

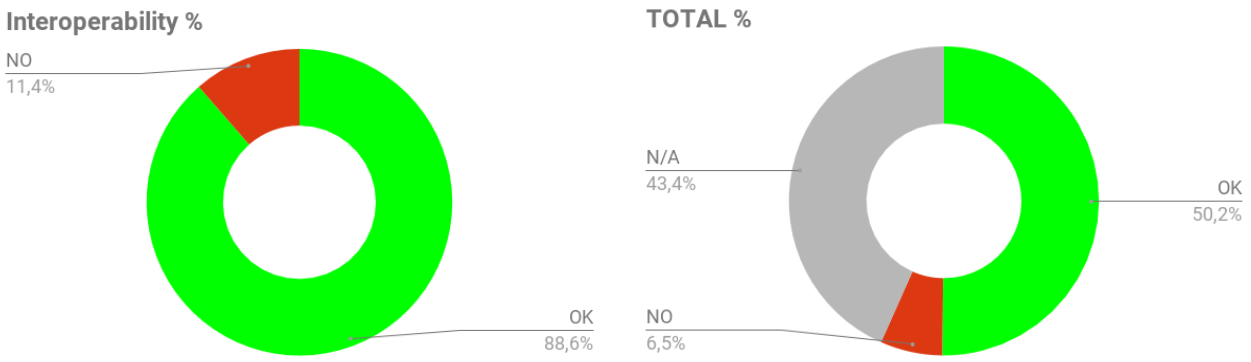

Figure 23. IOP Overall results 3rd NFV Plugtests (%)

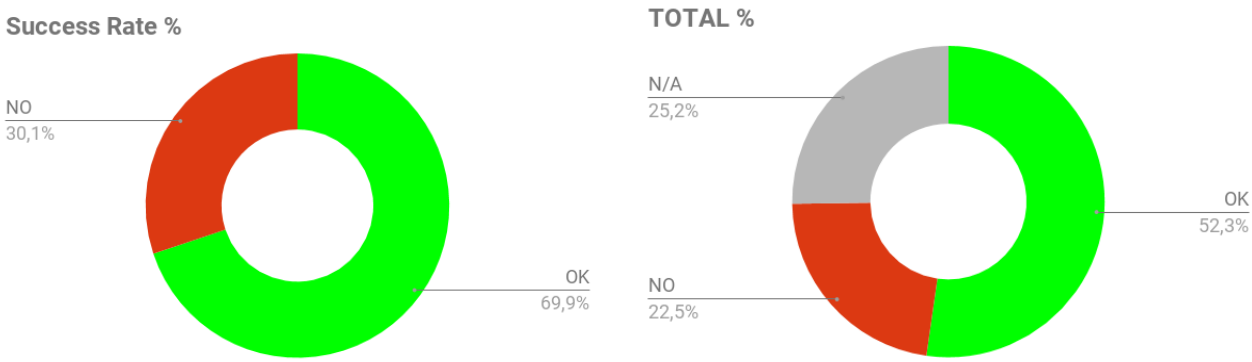

Figure 24. API Track Overall results 3rd NFV Plugtests (%)

The next clauses present more detailed results per test group and test cases and will allow to identify the areas and features with higher execution and interoperability rates.

10.2 Results per Group

10.2.1 Mandatory Sessions

10.2.1.1 Multi-Vendor NS

The table and figure below provide an overview of the results per Multi-Vendor NS group. Overall, 66 Multi-Vendor NS test sessions were run.

	Interoperability		Not Executed	Totals		Totals			
	OK	NO	NA	Run	Results	% Run	% OK	% NO	% NA
MV_ONBOARD	113	0	10	113	123	91,87%	100,00%	0,00%	8,13%
MV_INSTANTIATE	62	1	0	63	63	100,00%	98,41%	1,59%	0,00%
MV_SCALE_NS_MANUAL	26	0	84	26	110	23,64%	100,00%	0,00%	76,36%
MV_SCALE_NS_VNF_IND	8	3	37	11	48	22,92%	72,73%	27,27%	77,08%
MV_SCALE_NS_VIM_KPI	0	2	4	2	6	33,33%	0,00%	100,00%	66,67%

MV_SCALE_NS_VNF_REQ	2	0	14	2	16	12,50%	100,00%	0,00%	87,50%
MV_SCALE_VNF_MANUAL	18	1	41	19	60	31,67%	94,74%	5,26%	68,33%
MV_SCALE_VNF_VNF_IND	10	4	34	14	48	29,17%	71,43%	28,57%	70,83%
MV_SCALE_VNF_VIM_KPI	0	2	14	2	16	12,50%	0,00%	100,00%	87,50%
MV_SCALE_VNF_VNF_REQ	2	0	12	2	14	14,29%	100,00%	0,00%	85,71%
MV_UPDATE_VNF	54	5	55	59	114	51,75%	91,53%	8,47%	48,25%
MV_PM_VR	45	21	54	66	120	55,00%	68,18%	31,82%	45,00%
MV_PM_VNF_KPI	21	8	43	29	72	40,28%	72,41%	27,59%	59,72%
MV_FM_VR	8	6	46	14	60	23,33%	57,14%	42,86%	76,67%
MV_FM_VNF	16	6	49	22	71	30,99%	72,73%	27,27%	69,01%
MV_TERMINATE	58	1	3	59	62	95,16%	98,31%	1,69%	4,84%
MV_DELETE	108	0	16	108	124	87,10%	100,00%	0,00%	12,90%
TOTAL	551	60	516	611	1127	54,21%	90,18%	9,82%	45,79%

Table 28. Results per Multi-Vendor NS Sub-Group

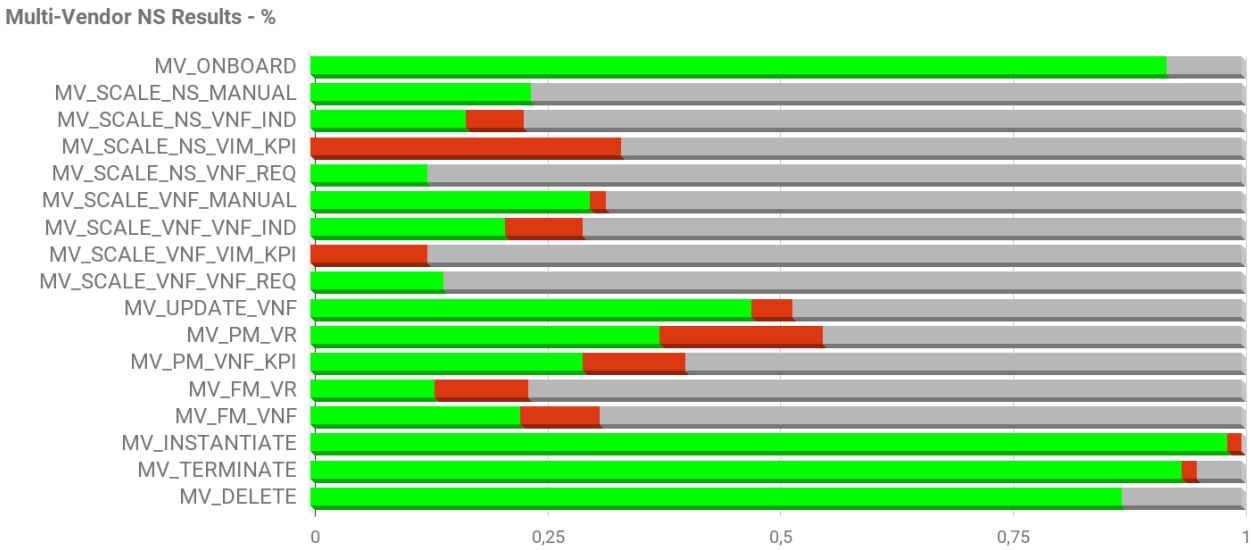

Figure 25. Results per Multi-Vendor NS Sub-Group - %

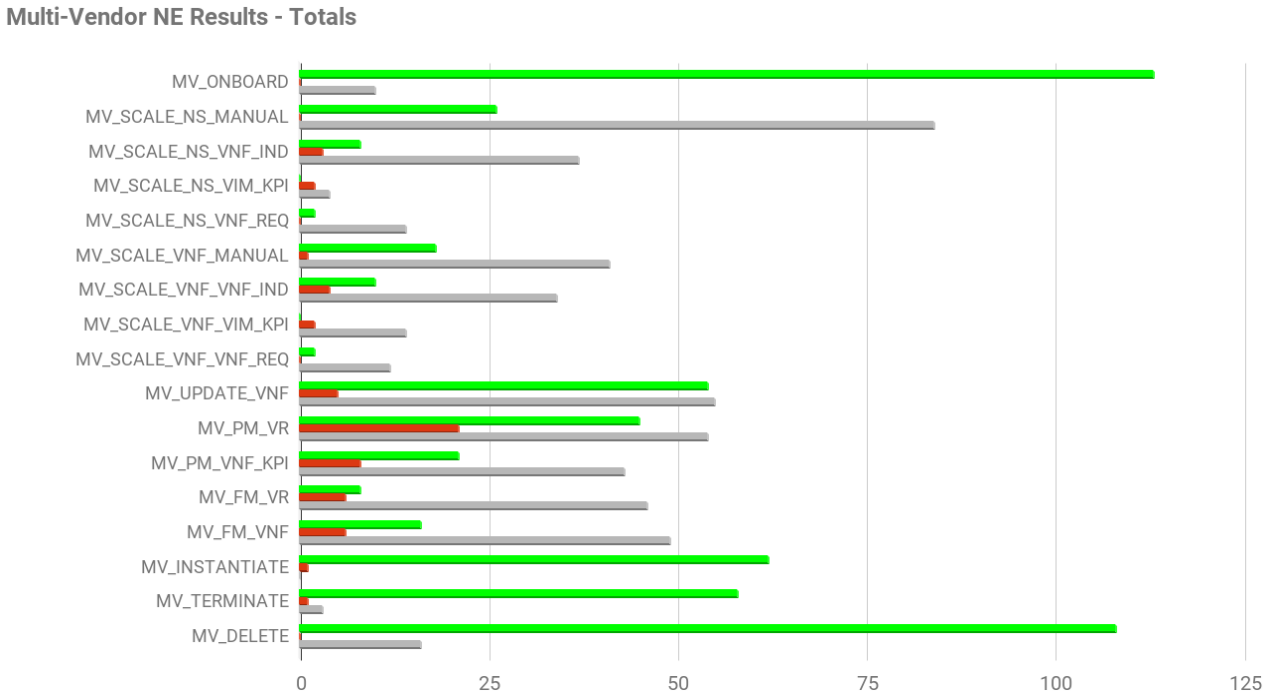

Figure 26. Results per Multi-Vendor NS Sub-Group - Totals

As shown in the figure below, the interop rate for this group was slightly higher than the one obtained during the 2nd NFV Plugtests (held only a few months before in January 2018) for Multi-VNF test sessions. While the number of test sessions was lower, the overall number of test cases run saw an increase of 11%, quite possibly due to the fact that the number of test cases for these sessions was higher, and therefore, the sessions longer. It is worth noting, that during the 1st NFV Plugtests, held in January 2017, all the testing focused on Single-VNF Network Services, so no result was collected for this group.

The small decrease in the number of this type of sessions was highly compensated by the significant increase of sessions and results reported for parallel activities like automated testing and API Validation sessions, as described in the next chapters.

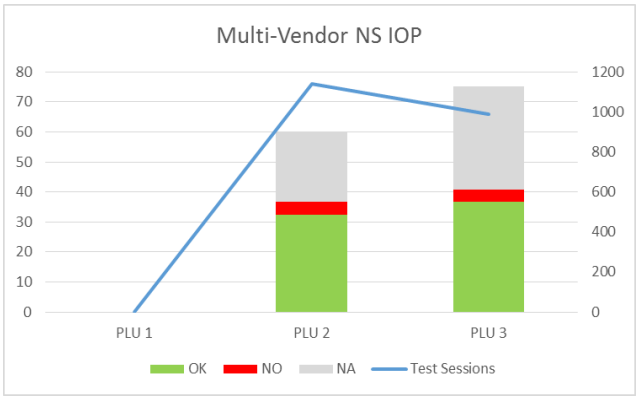

Figure 27. NFV Plugtests overview: Multi-Vendor NS

10.2.2 Optional Sessions

10.2.2.1 Overview

The tables and figures below provide an overview of the results for the test groups in the optional Test Sessions, including both interoperability and API Track test groups. Overall the number of optional test sessions and test cases run in these groups was lower than for the mandatory ones. The following clauses present the results on each group and subgroups.

Results per Group (optional sessions)	Number of Test Sessions	Success Rate (TCs Run)		TCs Not Run	TCs Totals	
		OK	NO	NA	Run	Total
Auto LC Validation	22	135	33	82	168	250
Multi-Site	1	10	0	4	10	14
Multi-vendor NS SFC	1	4	2	0	6	6
s-VNFM Direct	0	0	0	0	0	0
s-VNFM Indirect	0	0	0	0	0	0
Single-vendor NS	4	18	0	34	18	52
Multi-vendor NS EPA	3	18	0	0	18	18
Single-vendor NS Scale to level	0	0	0	0	0	0
API Track	9	58	25	28	83	111

Table 29. Results per Group (optional sessions)

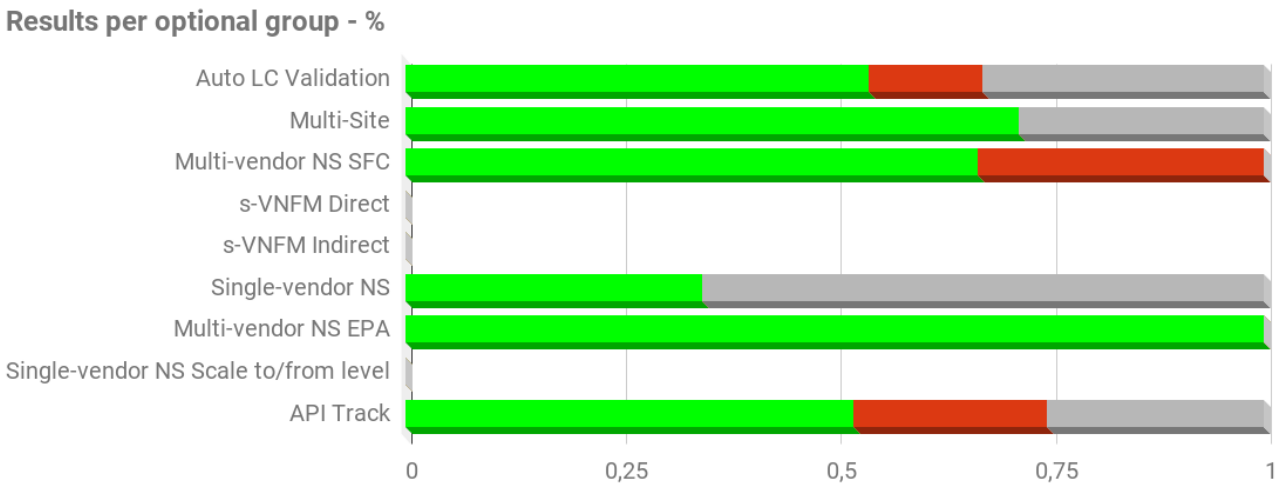

Figure 28. Results per Group (optional sessions) - %

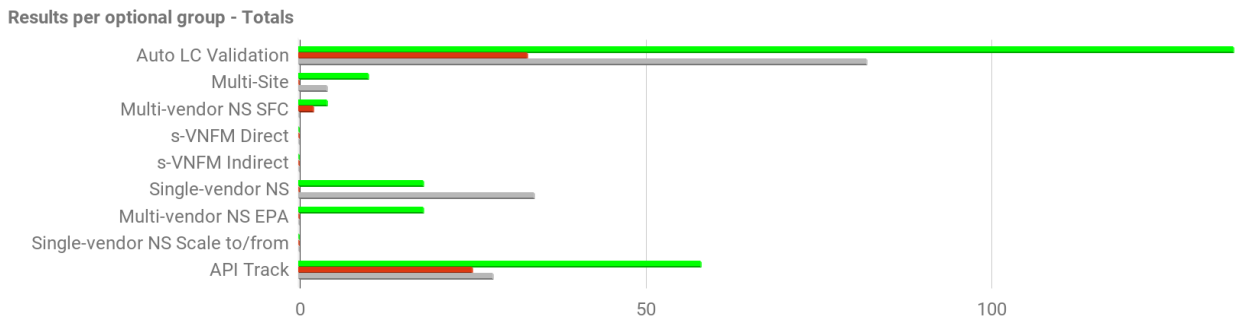

Figure 29. Results per Group (optional sessions) - Totals

10.2.2.2 Automatic LCM Validation

Automated testing saw an important progress with regards to previous Plugtests, as shown in the table below:

	Interoperability		Not Executed	Totals		Totals			
	OK	NO	NA	Run	Results	% Run	% OK	% NO	% NA
AUTOLCM_ONBOARD	22	0	0	22	22	100,00%	100,00%	0,00%	0,00%
AUTOLCM_INSTANTIATE	17	5	0	22	22	100,00%	77,27%	22,73%	0,00%
AUTOLCM_SCALE_NS_MANUAL	8	6	30	14	44	31,82%	57,14%	42,86%	68,18%
AUTOLCM_SCALE_VNF_MANUAL	17	6	7	23	30	76,67%	73,91%	26,09%	23,33%
AUTOLCM_UPDATE_VNF	17	7	20	24	44	54,55%	70,83%	29,17%	45,45%
AUTOLCM_FM_VNF	16	6	22	22	44	50,00%	72,73%	27,27%	50,00%
AUTOLCM_TERMINATE	16	3	3	19	22	86,36%	84,21%	15,79%	13,64%
AUTOLCM_DELETE	22	0	0	22	22	100,00%	100,00%	0,00%	0,00%
TOTAL	135	33	82	168	250	67,20%	80,36%	19,64%	32,80%

Table 30. Results per Automatic LCM Validation Sub-Group

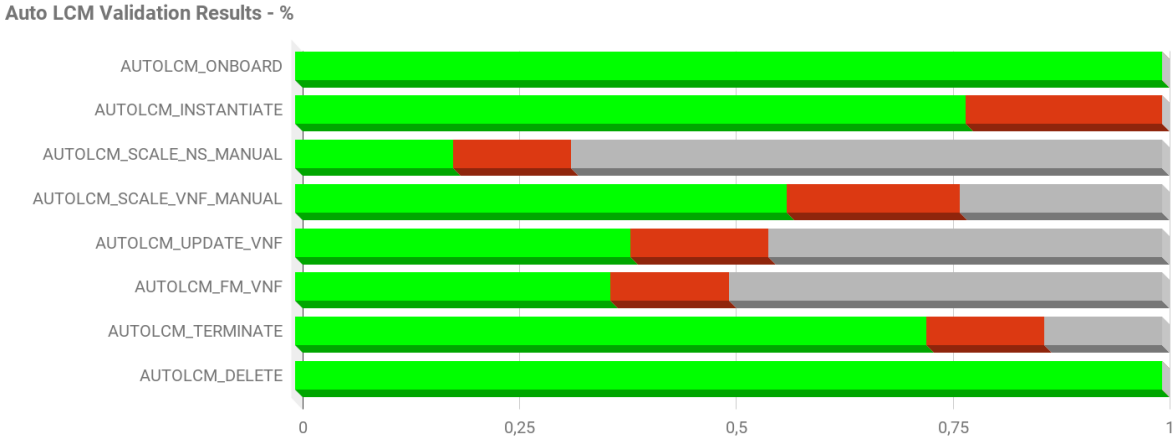

Figure 30. Results per Automatic LCM Validation Sub-Group - %

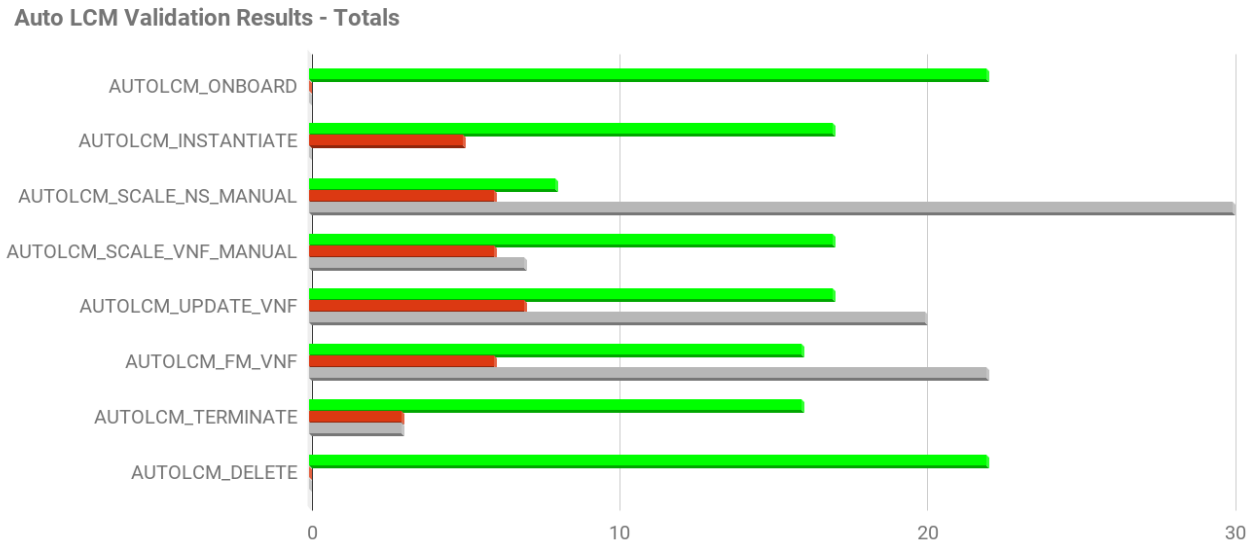

Figure 31. Results per Automatic LCM Validation Sub-Group – Total

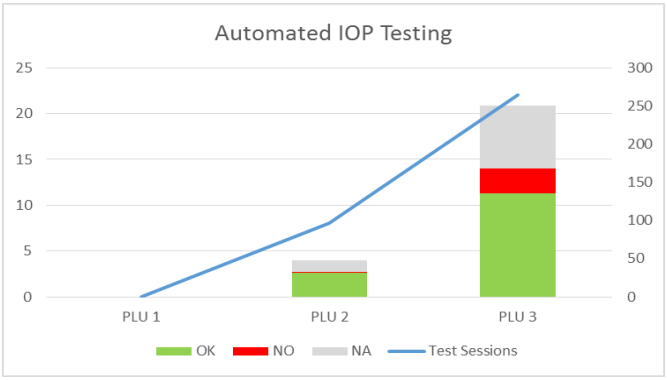

Figure 32. NFV Plugtests overview: Automated Interop Testing

If we compare with previous Plugtests, the figures show a significant increase in the number of automated test sessions (+175%) and the overall number of individual test cases that could be run automatically by a test system (+425%). This was due both to the increase in the number of test cases that were automated and by a growing interest in automation among Plugtests participants. Please note that automated testing started at the 2nd NFV Plugtests, only a few months before this event.

Automation brings many benefits over manual test execution, including increased test volume, repeatability and reliability of the testing process. Running tests repeatedly, in a continuous manner, facilitates the reveal of certain patterns, such as instabilities in the NFV system and inconsistencies of the operation outcomes. At the same time, valuable conclusions can be drawn from the statistical analysis of the large data sets collected during the testing process.

10.2.2.3 Single-Vendor NS

This configuration was offered for pre-testing purposes, in order to test Single VNF Networks Services before focusing on more complex multi-vendor NS. As seen in the figures, very little results were reported, as participants were able to build on achievements from the previous Plugtests only a few months before and could very quickly start testing multi-vendor NS configurations.

	Interoperability		Not Executed	Totals		Totals			
	OK	NO	NA	Run	Results	% Run	% OK	% NO	% NA

SV_ONBOARD	8	0	0	8	8	100,00%	100,00%	0,00%	0,00%
SV_INSTANTIATE	4	0	0	4	4	100,00%	100,00%	0,00%	0,00%
SV_SCALE_NS_MANUAL	0	0	2	0	2	0,00%	0,00%	0,00%	100,00%
SV_SCALE_NS_VNF_IND	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
SV_SCALE_NS_VIM_KPI	0	0	2	0	2	0,00%	0,00%	0,00%	100,00%
SV_SCALE_NS_VNF_REQ	0	0	2	0	2	0,00%	0,00%	0,00%	100,00%
SV_SCALE_VNF_MANUAL	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
SV_SCALE_VNF_VNF_IND	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
SV_SCALE_VNF_VIM_KPI	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
SV_SCALE_VNF_VNF_REQ	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
SV_UPDATE_VNF	0	0	8	0	8	0,00%	0,00%	0,00%	100,00%
SV_PM_VR	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
SV_PM_VNF_KPI	0	0	8	0	8	0,00%	0,00%	0,00%	100,00%
SV_FM_VR	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
SV_FM_VNF	0	0	6	0	6	0,00%	0,00%	0,00%	100,00%
SV_TERMINATE	2	0	2	2	4	50,00%	100,00%	0,00%	50,00%
SV_DELETE	4	0	4	4	8	50,00%	100,00%	0,00%	50,00%
TOTAL	18	0	34	18	52	34,62%	100,00%	0,00%	65,38%

Table 31. Results per Single-Vendor NS Sub-Group

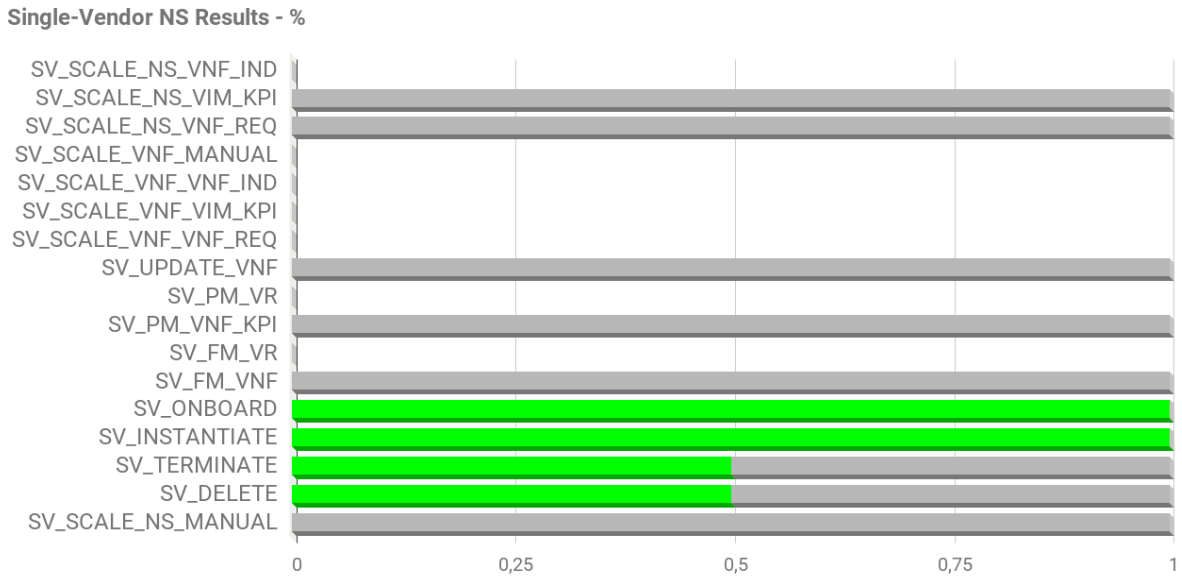

Figure 2. Results per Single-Vendor NS Sub-Group - %

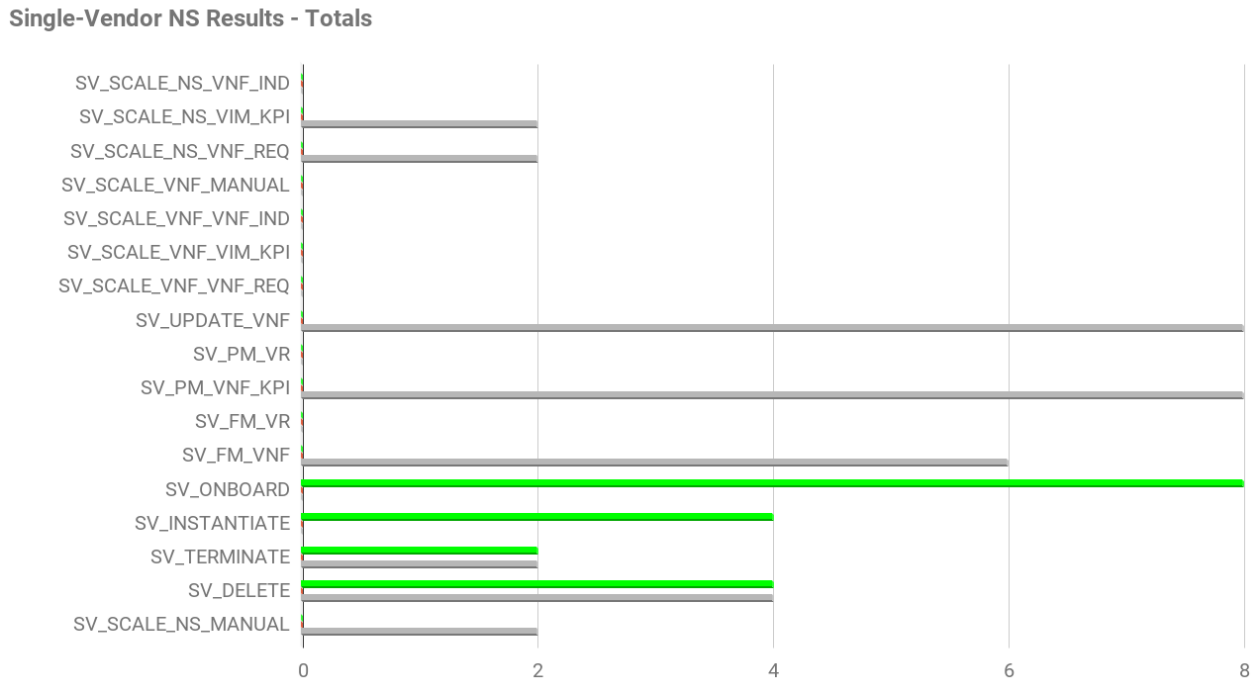

Figure 3. Results per Single-Vendor NS Sub-Group - Total

10.2.2.4 Single-Vendor NS Scale To Level

Single-Vendor NS Scale To Level test groups were optional during the 3rd NFV Plugtests. No results were reported for them, possibly because the test cases were contributed quite late in the Plugtests preparation process. See the list of related tests cases in clause 8.2.

10.2.2.5 Multi-Vendor NS with EPA

	Interoperability		Not Executed	Totals		Totals			
	OK	NO	NA	Run	Results	% Run	% OK	% NO	% NA
MV_EPA_ONBOARD	6	0	0	6	6	100,00%	100,00%	0,00%	0,00%
MV_EPA_INSTANTIATE	3	0	0	3	3	100,00%	100,00%	0,00%	0,00%
MV_EPA_SCALE_NS_MANUAL	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
MV_EPA_SCALE_VNF_MANUAL	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
MV_EPA_TERMINATE	3	0	0	3	3	100,00%	100,00%	0,00%	0,00%
MV_EPA_DELETE	6	0	0	6	6	100,00%	100,00%	0,00%	0,00%
TOTAL	18	0	0	18	18	100,00%	100,00%	0,00%	0,00%

Table32. Results per Multi-Vendor NS with EPA Sub-Group

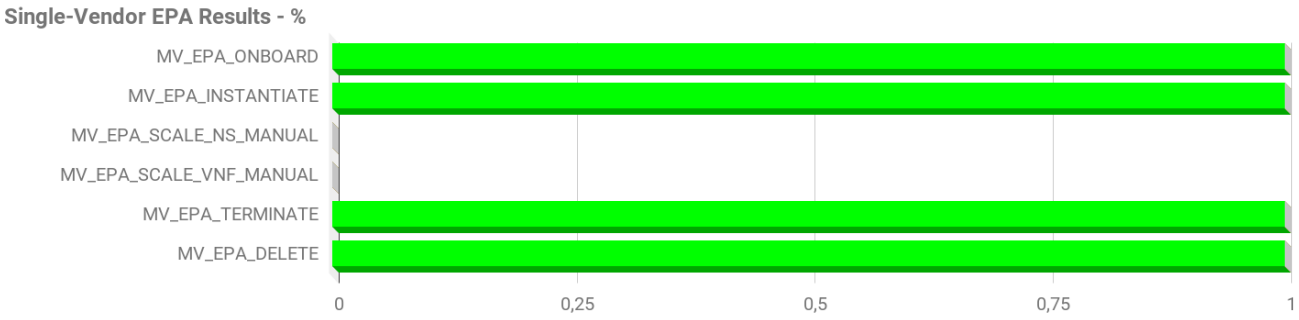

Figure 4. Results per Multi-Vendor NS with EPA Sub-Group - %

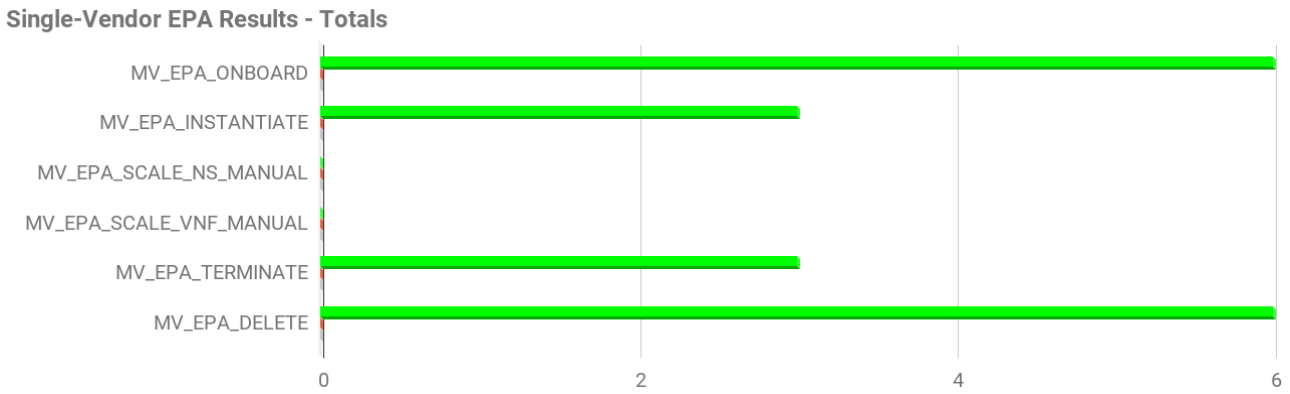

Figure 5. Results per Multi-Vendor NS with EPA Sub-Groups - Total

10.2.2.6 Multi-Vendor NS with SFC

	Interoperability		Not Executed	Totals		Totals			
	OK	NO	NA	Run	Results	% Run	% OK	% NO	% NA
MV_SFC_ONBOARD	1	1	0	2	2	100,00%	50,00%	50,00%	0,00%
MV_SFC_INSTANTIATE	0	1	0	1	1	100,00%	0,00%	100,00%	0,00%
MV_SFC_TERMINATE	1	0	0	1	1	100,00%	100,00%	0,00%	0,00%
MV_SFC_DELETE	2	0	0	2	2	100,00%	100,00%	0,00%	0,00%
TOTAL	4	2	0	6	6	100,00%	66,67%	33,33%	0,00%

Table 33. Results per Multi-Vendor NS with SFC Sub-Group

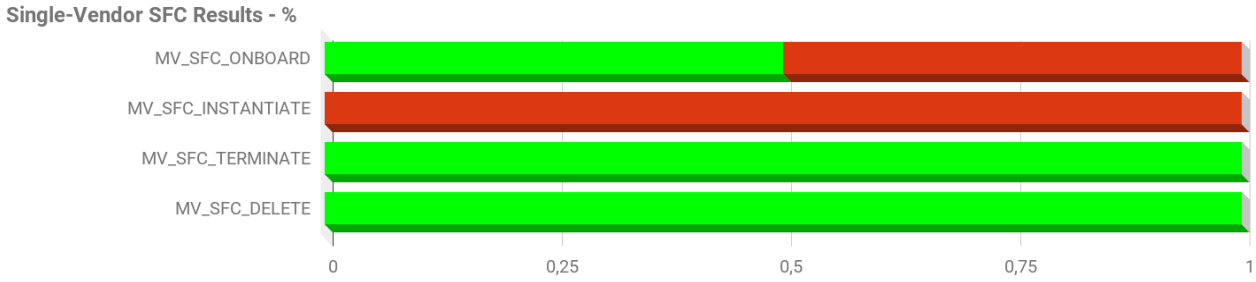

Figure 6. Results per Multi-Vendor NS with SFC Sub-Group - %

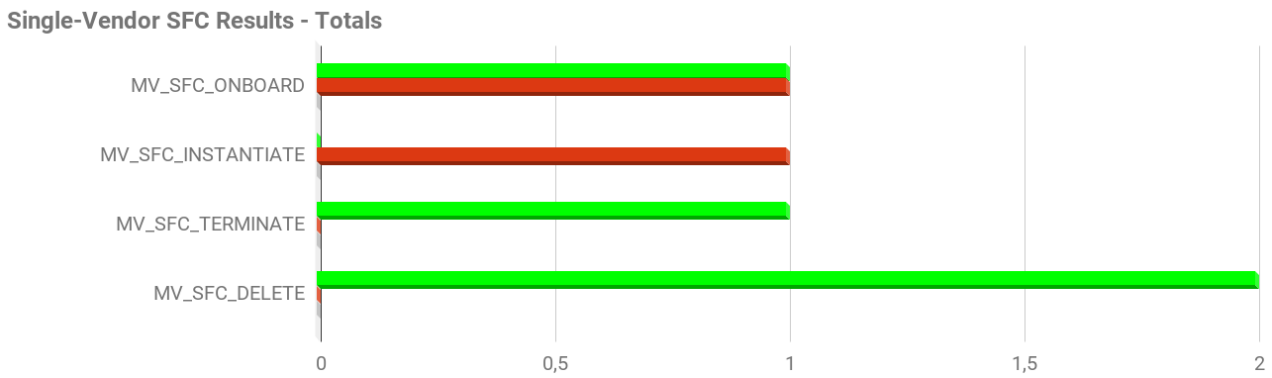

Figure 7. Results per Multi-Vendor NS with SFC Sub-Groups - Total

10.2.2.7 Multi-Site

	Interoperability		Not Executed	Totals		Totals			
	OK	NO	NA	Run	Results	% Run	% OK	% NO	% NA
MS_MV_ONBOARD	2	0	0	2	2	100,00%	100,00%	0,00%	0,00%
MS_MV_INSTANTIATE	1	0	0	1	1	100,00%	100,00%	0,00%	0,00%
MS_MV_SCALE_NS_MANUAL	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
MS_MV_SCALE_VNF_MANUAL	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
MS_MV_PM_VR	4	0	0	4	4	100,00%	100,00%	0,00%	0,00%
MS_MV_PM_VNF_KPI	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
MS_MV_FM_VR	0	0	2	0	2	0,00%	0,00%	0,00%	100,00 %
MS_MV_FM_VNF	0	0	2	0	2	0,00%	0,00%	0,00%	100,00 %
MS_MV_TERMINATE	1	0	0	1	1	100,00%	100,00%	0,00%	0,00%
MS_MV_DELETE	2	0	0	2	2	100,00%	100,00%	0,00%	0,00%
TOTAL	10	0	4	10	14	71,43%	100,00%	0,00%	28,57%

Table 34. Results per Multi-Site Sub-Group

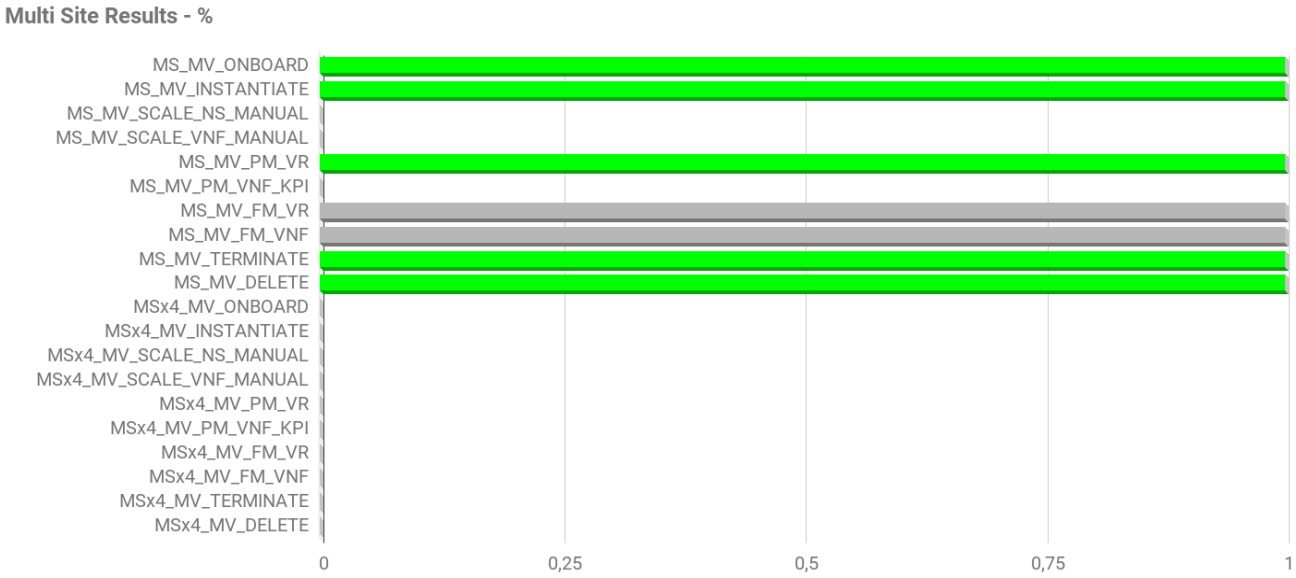

Figure 8. Results per Multi-Site Sub-Group - %

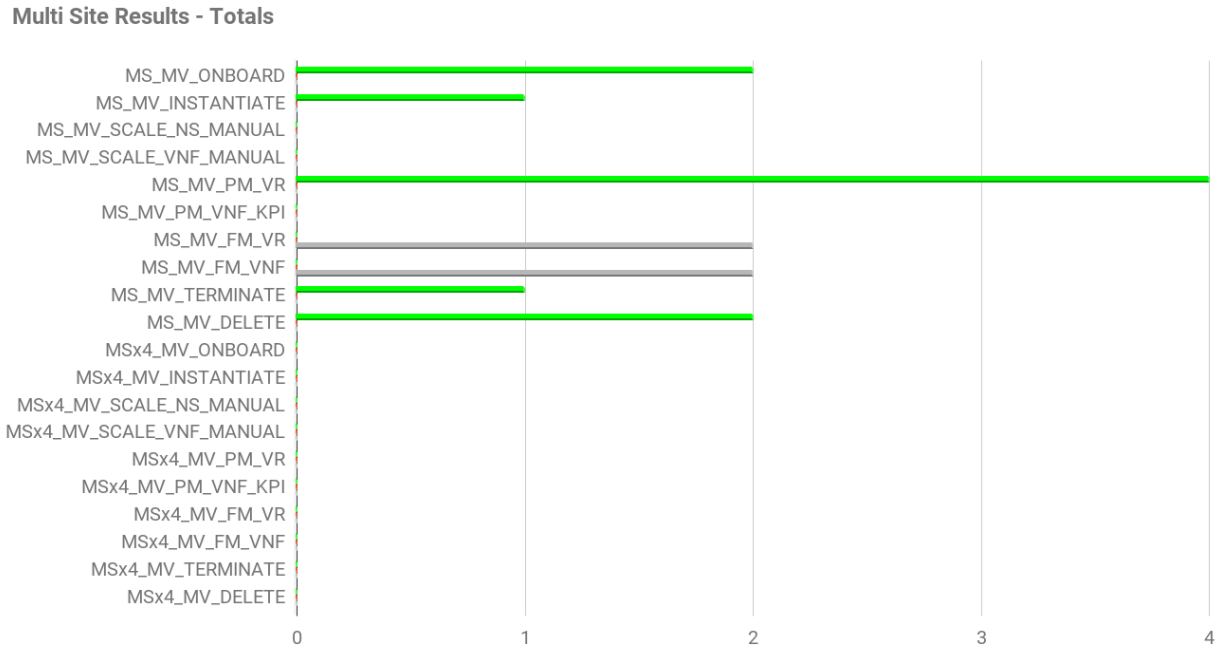

Figure 9. Results per Multi-Site SFC Sub-Groups - Total

10.2.2.8 Specific VNFM

Specific VNFM test groups were optional during the 3rd NFV Plugtests. No results were reported for them on either direct or indirect mode: S-VNFM-D and S-VNFM-I. See the list of tests cases for S-VNFM-D and S-VNFM-I test groups in 8.5.

10.2.2.9 API Track

The NFV API validation track run by ETSI in parallel with the interoperability test sessions allowed participants to evaluate the alignment of their implementations with NFV SOL OpenAPIs. The scope of this API track, which was experimented during the 2nd NFV Plugtests for the first time with a subset of NFV-SOL002 and NFV-SOL003 APIs, was extended to include NFV-SOL005, the Northbound Interface of NFV MANO.

	Interoperability		Not Executed	Totals		Totals			
	OK	NO	NA	Run	Results	% Run	% OK	% NO	% NA
API_NFVO_SOL5_NSDM	3	3	1	6	7	85,71%	50,00%	50,00%	14,29%
API_VNF_SOL2_CONF	3	2	1	5	6	83,33%	60,00%	40,00%	16,67%
API_VNF_SOL2_IND	8	5	1	13	14	92,86%	61,54%	38,46%	7,14%
API_VNFM_SOL3_LCM	44	15	25	59	84	70,24%	74,58%	25,42%	29,76%
API_NFVO_SOL3_PKGM	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
API_NFVO_SOL3_GRANT	0	0	0	0	0	0,00%	0,00%	0,00%	0,00%
TOTAL	58	25	28	83	111	74,77%	69,88%	30,12%	25,23%

Table 35. Results per API Track Sub-Group

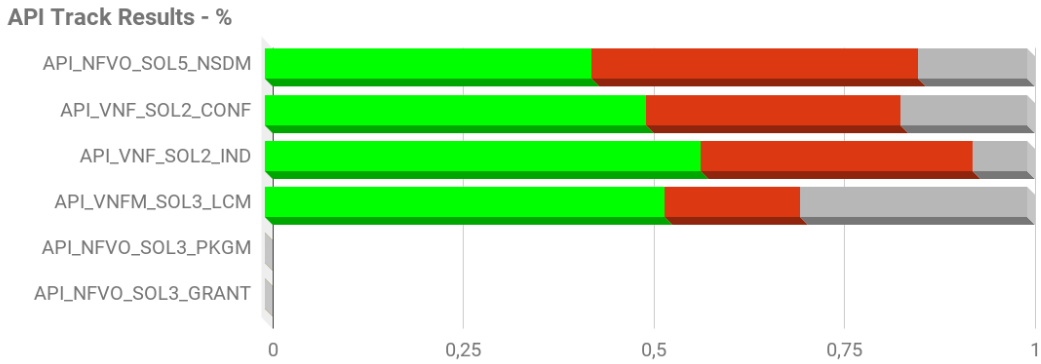

Figure 41. Results per API Track Sub-Group - %

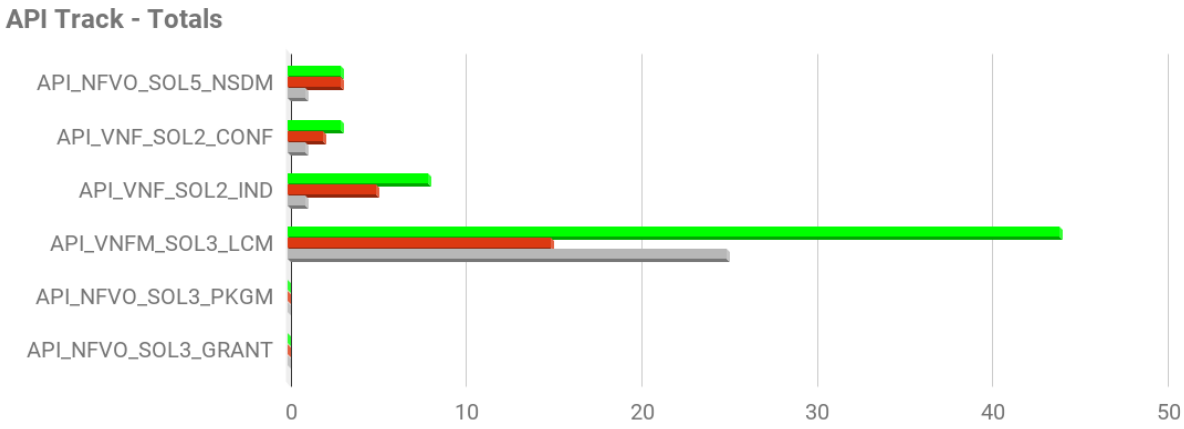

Figure 42. Results per API Track Sub-Group - Total

The API track saw also a growing interest among participants, with a significant increase in the number of test sessions (+125%) and overall number of test cases run (+100%) with regards to the previous event

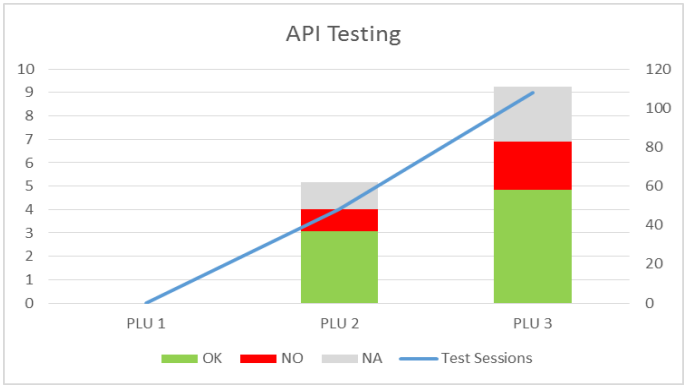

Figure 43. NFV Plugtests overview (API Testing)

10.3 Results per Test Case

The full list of results grouped per Test Case, including both interoperability and API Track test sessions is provided in Annex A.

11 Interoperability Demonstrations

During the second week of Plugtests, participants were offered the opportunity to run some multi-vendor demonstrations and share the learnings with the community.

11.1 Demo 1: Multi-VNF NS Orchestration

Participants: A10 Networks, Fortinet, Spirent, Cisco, Wind River, Red Hat, Lenovo

The demo addressed an advanced customer service chain use-case, using multi-vendor technologies, and provided a complete demonstration of the orchestrated instantiation, customer traffic generation, scaling out/in and moving of various service VNFs. Specifically, it used the NFVO/VNFM MANO solution to instantiate a complete Network Service (NS) which included Firewall VNF(s) (vFW) front-ended by a Load Balancer VNF (vLB). Client and server traffic generator VNFs were included in the Network Service and simulated customer bearer HTTP traffic.

Once deployed and traffic simulated, the VNFM monitored the vFW SNMP session rate MIB and triggered a scale-out notification to the NFVO when a pre-configured 750 sessions/sec threshold was hit. The NFVO/VNFM then instructed the VIM to add a second vFW VNF into the service chain topology (as defined in the NS). When the new vFW VNF was deployed, the VNFM instructed the NFVO to update the existing vLB running configuration (using native vendor commands) to incorporate the new vFW device into its load balancing selection algorithm. The vLB then implemented HTTP session-based load balancing across the multiple vFW instances. This demonstrated an automated VNF "dayN" configuration change to an existing NS/VNF, triggered by network conditions, as the NS could not depend on “day0” VNF configuration files alone. The demo also showcased the NFVO/VNFM moving the complete service chain across different VIM platforms.

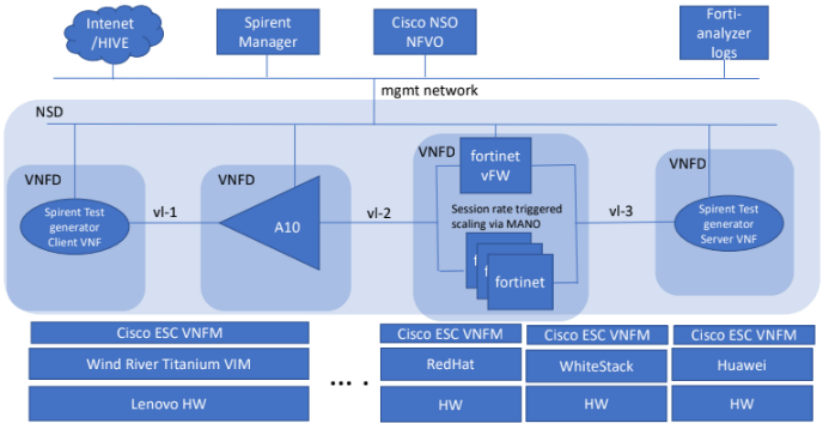

Figure 44. Demo 1: Multi-VNF NS Orchestration

11.2 Demo 2: Orchestrated Service Assurance in NFV

Participants: Anritsu, ng4T, Cisco, Wind River, Lenovo

Telecom operators are now trying to productize their initial NFV deployments and need to have the right tools to obtain a clear understanding of telecom network status (vEPC, vIMS, etc.).

The concept of Service Assurance in NFV is important for Network Operator operations teams as they need to know how telecom network is behaving once it is virtualized.

This demo showed how vEPC troubleshooting and real-time monitoring can be achieved using a passive probing solution deployed in a multi-vendor NFV environment, composed of:

- vEPC emulator from ng4T: End-users, vRAN and vEPC Core are emulated by ng4T VNFs using a pre-determined traffic pattern.

- MANO from CISCO: handles instantiation and termination of the VNFs and network service. Scale-out/in were not showcased.
- NFVI consisting of Lenovo servers running Wind River's Titanium VIM: provides a high reliability hosts for VNFs.
- vProbe solution from Anritsu: captures traffic in real-time, decodes and stores call data records, displays real-time (15 seconds granularity) user-defined KPIs on customizable dashboard

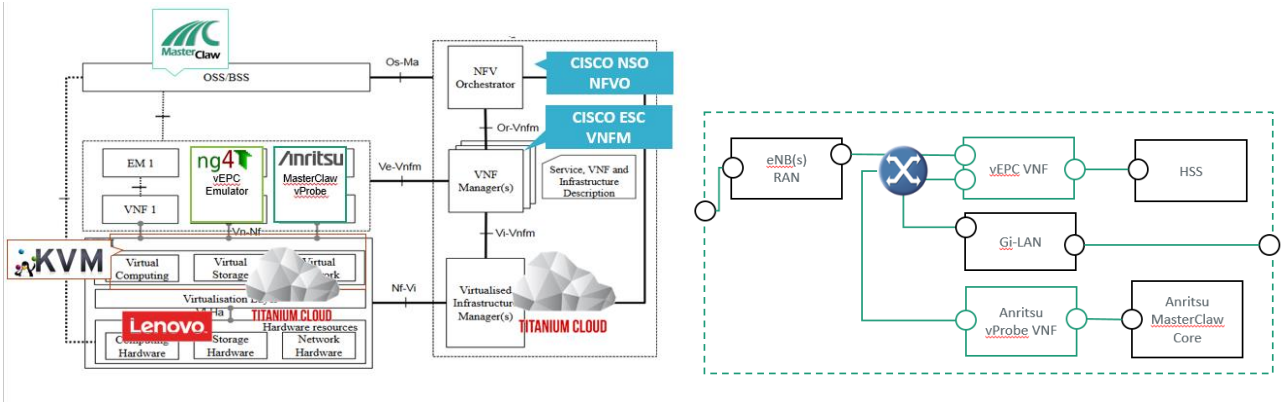

Figure 45. Demo 2: Orchestrated Service Assurance in NFV

11.3 Demo 3: 4G Mobile Network Orchestration

Participants: ng4T, OperAirInterface SA, Mobileum, Whitestack (OSM), Wind River, Huawei (OPNFV), Lenovo

This demo leveraged the collaboration among different vendors and open source communities to showcase a multi-vendor network service implementing a virtualised 4G mobile network, from the end users to the applications. The demo showed the different phases of an end-to-end service deployment, including:

- Phase 1 – Design: It started with the design of the NS using MANO modelling capabilities to adapt and take full advantage of the different environments, including VIM networking and security parameters, fully customizable at the descriptor level.
- Phase 2 – Onboarding: Then it showed how on-boarding can be handled through the MANO's northbound interface by using its graphical interface.
- Phase 3 – Instantiation: The demo tackled instantiation over the available VIMs, orchestrated from the catalogue at the graphical user interface.
- Phase 4 – Operation: Finally, it showcased NS operation including application end-to-end functionality, integration between components, initialization and configuration through MANO's VNF Manager. It also showed interaction with external tools for fault and performance monitoring and management.

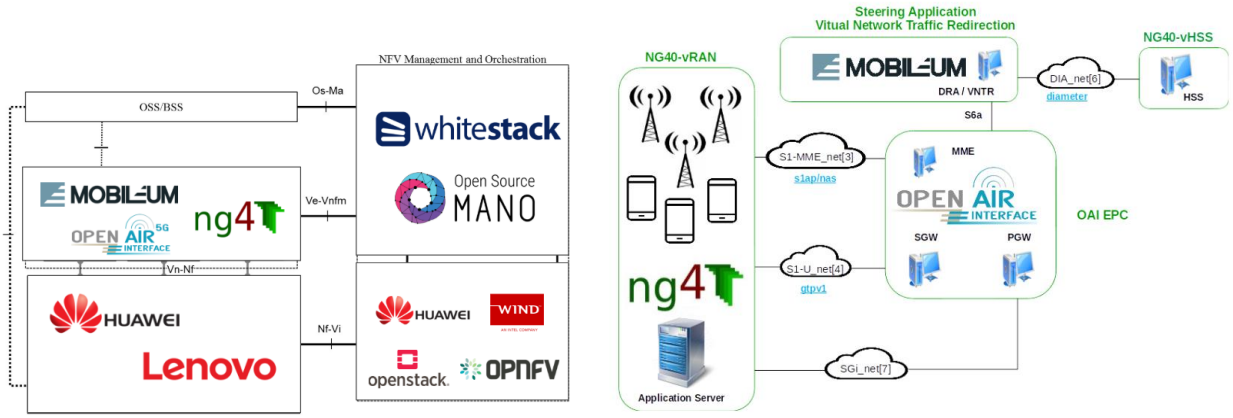

Figure 46. Demo 3: 4G Mobile Network Orchestration

11.4 Demo 4: Automated Life Cycle Validation

Participants: Fortinet, Spirent, Cisco, RIFT.io, Wind River

This demo showcased automated NS/VNF Life Cycle Validation in a multi-vendor NFV system based on one of the SUT Configurations in scope of this Plugtests. Automated testing allowed to verify:

- Quality of VNF service after repeated instantiation/termination cycles;
- Stability of the NFV virtual environment (trends of the measured lifecycle statistics);
- Robustness of lifecycle management operation, e.g. scale in/out operation.

The automated test cases run during the demo were based on 3rd ETSI Plugtests Test Plan V0.0.4, Chapter 6.8 AUTO LCM VALIDATION.

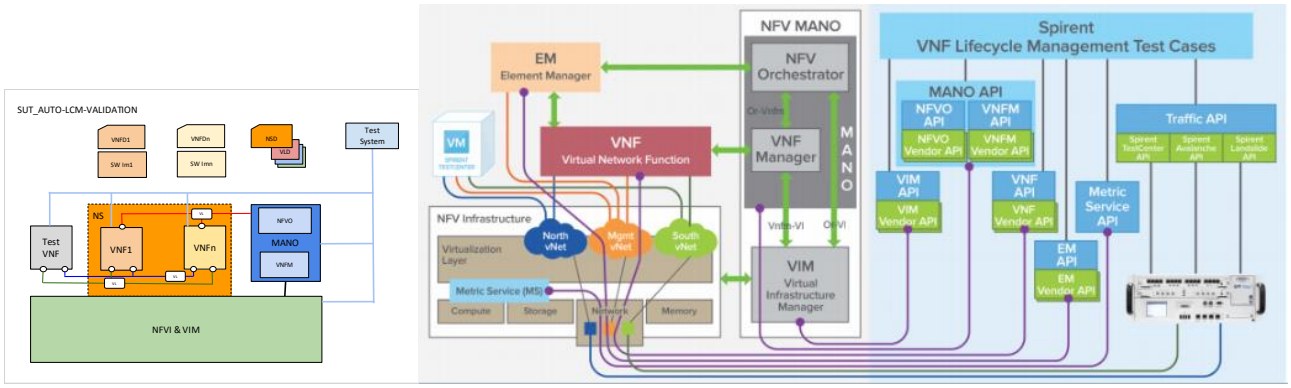

Figure 47. Demo 4: Test System for automated Life Cycle validation

11.5 Demo 5: NFV SOL Based VNFD Validation

Participants: Enterprise Web, Spirent

This demo showcased a tool allowing for SOL001 (Tosca) and SOL006 (YANG) VNF Descriptors validation.

During the demo, Tosca-based VNFDs from Spirent were tested for compliance with the ETSI specifications using the tool provided by Enterprise Web. The VNFDs were uploaded to a portal, validated instantly, and the tool identified non-

compliance ranging from invalid syntax (Tosca format errors), to invalid structure (missing or invalid attributes), and invalid semantics (inconsistent and mismatched identifiers). The tool was then used to rapidly debug the VNFDs bringing them back into compliance.

Finally, the demo concluded with a look at the graph-based VNFD model generated by EnterpriseWeb directly from the NFV-SOL specifications.

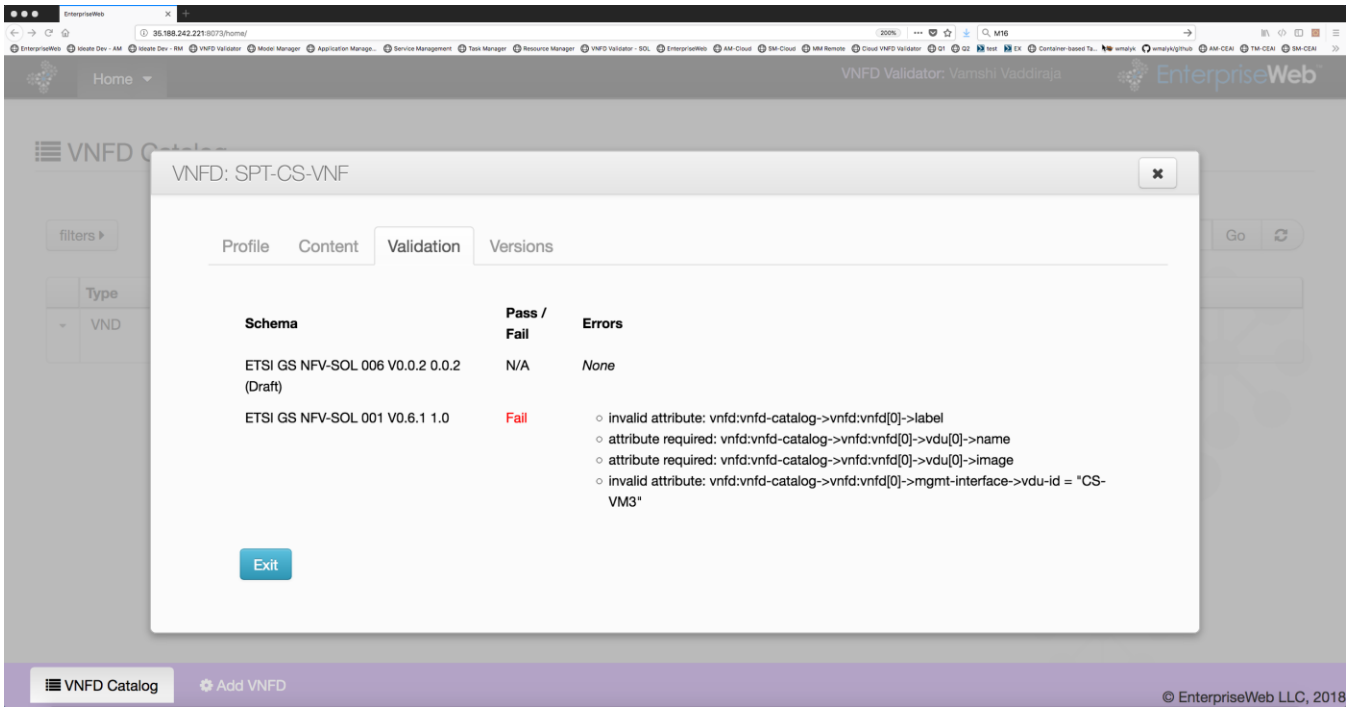

Figure 48. Demo 5: Detailed VNFD validation report

11.6 Demo 6: Multi-VNF NS Orchestration

Participants: Spirent, A10 Networks, SONATA (5Gtango), Red Hat, Lenovo.

This demo showcased a multi-VNF Network Service deployment, instantiation, agent-based monitoring and scaling. The Network Service was composed by a Spirent VNF, a traffic generator; an A10 VNF, a load balancer; and multiple NGINX VNFs, http servers.

The purpose of the demo was to highlight the ability of the SONATA MANO to scale the Network Service, by increasing the number of backend NGINX VNFs. The trigger for this scaling is provided by NGINX monitoring, considering the size of the HTTP queue. That gives a good hint about the stress of those VNFs.

The traffic generation was configured to gradually increase the amount of traffic (requests/s). The scaling actions included both the instantiation of a new NGINX VNF, as well as the provisioning of the A10 load balancer with a new back-end server. The SONATA SSM (Service Specific Function) allows the programming of the behaviour of a particular LCM operation (e.g. scaling), providing high levels of flexibility.

Figure 49. Demo 6: Multi-VNF NS Orchestration (NS Scaling)

12 Plugtests Outcome

During the 3rd NFV Plugtests a number of 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 specifications, NFV OpenAPIs, the test plans and the test automation.

12.1 Feedback on NFV Specifications

12.1.1 General

12.1.1.1 API and Specs Versioning

SOL APIs (SOL002, SOL003, SOL005) specifications v.2.4.1 were released without updating the version id of the APIs. Participants reported that this makes it impossible to indicate support of version 2.3.1.

This issue has been acknowledged by ETSI NFV SOL Working Group who is currently working on some proposals to solve the versioning issues.

12.1.1.2 JSON Schema validation, extra fields

Participants requested clarification on the semantics of JSON schema and their validation: in particular they wanted to know if when extra fields are added to a JSON object, the JSON schema validation outcome changes.

This approach could be useful to inject requirements or clarifications in the specifications as well as to define the testing strategies for APIs.

12.1.1.3 Id instead of location

Regarding the HyperMedia approach of the API, clarifications where asked on the reasons to return an entire URL in the Location response header, instead of the mere ID or the subject resource.

12.1.1.4 API readability

Some comments and suggestions where raised regarding API readability, in an effort to provide useful feedback for future changes in the API design. They touch many design decisions that have been taken in the beginning, but are key issue when API adoption is at stake, thus, still important for discussion.

1. **Naming:** when we write `{apiRoot}/nsd/v1/ns_descriptors`: we are referring to 'network service descriptors' twice (nsd & ns_descriptors) which might be confusing. It could be advisable to follow a non-redundant approach, like;
    - `{apiRoot}/nss/v1/descriptors`, or
    - `{apiRoot}/network-services/v1/descriptors`
2. **REST-ify APIs:** suggestions:
    - To address **resources** with a plural name, like in (e.g.) `/network-services` (i.e., no verbs -- the only verbs would be the HTTP ones, GET, POST, ...);
    - To transform actions into resources that are created, meaning the verb/action is executed (a good example is `/subscriptions`)
    - To consider having either:

- a generic endpoint to treat all operations (e.g., `.../ns-lcm-requests`), with the body of POST requests having the NS id (instantiations) or the NS Instance ID (updates and terminations), plus all the remaining needed data; or
- specific per operation endpoints (e.g., POST `.../nss/:id/instances` would instantiate a NS, POST `.../nss/:id/instances/:instance_id/terminations` to terminate, etc.)

3. **Versioning:** a special case of naming... for example, consider versioning the whole set of APIs, like in (e.g.) `.../vX/nsd`
4. **/info** and **/ns_descriptors:**
   - `.../info` and `.../ns_descriptors` have a high degree of cohesion: we cannot accept one and not the other, for a given NS;
   - Why separating the `.../info` from the `.../ns_descriptors`, when it could be part of the NSD's meta-data?
   - What should happen when just (e.g.) the `info` is successfully created (but not the `ns_descriptors`)? Should we consider 'automatic cleaning' operations in the Service Platform?
   - **Suggestion:** to merge both endpoints and treat /info-related data as a meta-data section of the object being stored, with the /ns_descriptors-related data being the data section of the same object;
5. **PNFs at the NS level:** some questions were raised on why PNFs are considered at the NS level instead of having their own entity like VNFs have. After discussion it was concluded that the reason could be that the PNFs are not "lifecycle-managed" by the NFV-MANO, and are only relevant and used when they are part of an NS, and the NFV-MANO is involved or supports establishing the connectivity to the PNF.

12.1.2 SOL002

12.1.2.1 VNF instances ids inconsistency

According SOL003 v2.4.1 "The VNFM creates a new VNF instance resource in NOT_INSTANTIATED state, and the associated VNF instance identifier" but according to SOL 002 v2.4.1 VNF indicator interface provides indicators to individual VNF instances through two endpoints:

1. GET /indicators/{vnfInstanceId}
2. GET /indicators/{vnfInstanceId}/{indicatorId}

It was raised a question on how the server endpoint - which implements VNF indicator interface - knows the mapping vnfInstanceIds to internal VNF services (which provide themselves indicators) provided that VNFM generates vnfInstanceId in the LCM interface and if this is done independently from the VNF.

Clarifications on when and how vnfInstanceId is communicated between VNFM and VNF were requested. In particular, whether it is expected to be a Pull (VNF initiated) or Push (VNFM initiated) communication.

During the discussion it was mentioned that for passing the vnfInstanceId, a possibility was to reuse the VNF Configuration interface specified in clause 9 of SOL002 v2.4.1 (VNFM push method). The vnfInstanceId can be passed as a KeyValuePair of the vnfSpecificData attribute in the VnfConfigurationData data type (see clause 9.5.3.3 of SOL002 v2.4.1). In the specific context of the VNF indicators interface, it would also be relevant to discuss whether the vnfInstanceId is needed in the VNF Indicator interface produced by a VNF instance.

12.1.3 SOL003

12.1.3.1 VNFM Assignment of floating IPs

According to IFA011, the VNFD contains an information, that some VduCpd/VnfExtCps shall have floating IPs associated with them, but InstantiateVnfRequest and GrantRespons do not provide information about in which network these addresses shall be allocated.

During the discussion it was clarified that the ExtVirtualLinkData type (see clause 4.4.1.11 of SOL003 v2.4.1) and its child data types provide several placeholders for providing address information, e.g., VnfExtCpData (clause 4.4.1.10), CpProtocolData (clause 4.4.1.10a) and IpOverEthernetAddressData (clause 4.4.1.10c). The ExtVirtualLinkData is used in both the InstantiateVnfRequest and the GrantResponse.

12.1.3.2 Connection points order in VNFD

Some participants raised that an IFA011 compatible VNFD does not provide information about associating Cps with VM network cards, so it is no clear which physical interface (eth0, eth1, etc) shall be used to implement which Cp. However, sometimes this is very important to specify, e.g. because only eth0 may have a DHCP client enabled.

12.1.3.3 Relation between vnfdId and vnfPkgId

The correlation between the 2 identifiers vnfId and vnfPkgId is maintained by the NFVO. Upon request of vnf Instance Id creation the VNFM needs to query the NFVO to retrieve the vnfPgkId.

It was suggested to allow/recommend the NFVO to add this information in the createVnfInstanceReq.

12.1.3.4 VIM Connection info

It was suggested to standardise (at least as a recommendation) the data needed in the vimConnectionInfo object? (e.g. IP/PORT and AUTH endpoint)

It was clarified that this is currently under discussion in the ETSI NFV SOL WG. It is related to the fact that initially it was envisioned that (part of the) values to be used in the vimConnectionInfo object would be known based on a registry. Further guidance or specification is expected to be provided in the future.

12.1.3.5 extCps vs extLinkPorts

A question was raised on the main difference between extCps vs extLinkPorts. It was discussed that the difference comes mainly from the information modelling. The ETSI GR NFV-IFA 015 differentiates between the linkPorts and CPs. The linkPorts are the attachment point from the network point of view (aka a vPort), whereas the CPs are the attachment to the network from the VNF/VNFC point of view (aka a vNIC). In practice, some NFVI-VIM solutions do not make such a big distinction because they realize the vNICs as network ports.

12.1.4 SOL005

12.1.4.1 Typos

The following typos were reported:

- In "**Table 5.4.2.3.1-2: Details of the POST request/response on this resource**" the reference to **CreateNsdInfoRequest** data type is wrong. The clause reported is **5.5.2.4** but shall be **5.5.2.3**.
- In "**Table 5.4.5.3.2-2: Details of the GET request/response on this resource**" the reference to **PnfdInfo** data type is wrong. The clause reported is **5.5.2.2** but shall be **5.5.2.5**.

12.1.4.2 Pkg management content return type

It was reported that allowing only ZIP as the response content type in /package_content (both PUT and GET) decreases readability. It was suggested to allow other Content types, e.g. text/plain.

12.1.4.3 State of NSs

Some participants asked the meaning of Flavour at the NS level. It was concluded that the meaning is similar to flavour at VNF level. The NS flavour concept provides flexibility to the network service designer to express different NS setups using a single NSD. For instance, one may define an NS flavour that makes use of a certain VNF flavour, with some specific Virtual Links with certain QoS requirements, etc.

12.2 Feedback on NFV OpenAPIs

12.2.1 Bugs in OpenAPIs

The following bugs were reported in the NFV SOL OpenAPIs.

- The schema of 200 response of GET **/subscriptions** in SOL003 LCM expects one object of type Subscription instead of an array with Subscriptions as items.
- The schema of 200 response of GET **/vnf_lcm_op_occs** in SOL003 LCM expects one object of type LcmOperation instead of an array with LcmOperation as items.
- The spec of SOL003 defines no payload on response of **/operate** task in LCM API, while the OpenAPIs has a schema of an expected payload.
- The spec of SOL003 defines no payload on response of **/scale task** in LCM API, while the OpenAPIs has a schema of an expected payload.
- The spec of SOL003 defines no payload on response of **/scale_to_level** task in LCM API, while the OpenAPIs has a schema of an expected payload.
- The spec of SOL003 defines no payload on response of **/change_flavour** task in LCM API, while the OpenAPIs has a schema of an expected payload.
- The spec of SOL003 defines no payload on response of **/terminate task** in LCM API, while the OpenAPIs has a schema of an expected payload.
- The spec of SOL003 defines no payload on response of **/heal task** in LCM API, while the OpenAPIs has a schema of an expected payload.
- The spec of SOL003 defines no payload on response of **/change_ext_conn** task in LCM API, while the OpenAPIs has a schema of an expected payload.
- The spec of SOL003 defines notificationTypes as ENUM instead of array of ENUMS in create notification subscription.
- SOL 005 GET /ns_descriptors should return an array.
- SOL 005 GET /ns_descriptors identifiers should be strings not objects.
- SOL 002 /configuration, vnfConfigurationData.extCps should be an array instead of an object.
- SOL 002 /configuration vnfcConfigurationData.extCps should be vnfcConfigurationData.intCps and should be an array instead of an object.
- SOL 003, LCM, resourceHandle has vimConnectionId optional in clause 4.4.1.7 but is required in the OpenAPIs.

12.2.2 Recommendations

12.2.2.1 "Null" value in optional Fields

In the OpenAPIs representation of NFV APIs, currently the "null" value is not allowed for optional fields. This means that the current schema requires either to remove the optional field or to provide some value in it.

During the Plugtests it was observed that several REST libraries will produce optional empty fields with a "null" value for a non-present optional field, which semantically is equivalent to stripping the field off.

Given this, it could be advisable to allow the use of "null" value in those optional fields where objects and strings are allowed. E.g. to replace:

```
"type": "string"
```

By:

```
"type": ["string", "null"]
```

12.3 Feedback on the Test Plans

12.3.1 Interoperability Test Plan – TST007

The Interoperability Test Plan for the 3rd NFV Plugtests was developed by leveraging on the test plans of previous NFV Plugtests [1NFVPLU-TP] and [2NFVPLU-TP] aiming at improving them while maximizing the execution rates of the different test cases.

In particular, the 3rd NFV Plugtests Test Plan has been developed in the Plugtests preparation phase, from April to May 2018, also involving participants to collect their feedbacks and ideas for potential improvements with respect to existing test cases and test descriptions. The process of Test Plan improvement has been implemented following two main directions:

1. review and consolidate the existing test cases and test descriptions,
2. add new interoperability features and test cases.

On one hand, the 2nd NFV Plugtests Test Plan [2NFVPLU-TP] went through a consolidation process to simplify and generalize some specific test cases (e.g. Performance Monitoring and Fault Management), with the ultimate goal of trying to increase execution rate. In this direction, the 2nd NFV Plugtests Fault Management and Performance Monitoring test cases and test descriptions for Single and Multi-Vendor NS groups (see Clause 8.2 and 8.3) were generalized in terms of execution steps and descriptions to accommodate different implementations of monitoring and fault management features across FUTs, also explicitly describing the MANO as composed by an NFVO and a (generic) VNFM.

Similarly, the specific VNFM test cases (see Clause 8.5) were updated to specifically include (in the Test Configuration and Test Descriptions) a specific / external VNFM (either in indirect or direct resource allocation mode) and thus allow to adapt the Test Descriptions for each configuration when needed.

On the other hand, some new test cases have been introduced in this 3rd NFV Plugtests Test Plan, to cover additional interoperability features:

- SFC implemented through NSH protocol,
- NS and VNF scale to level.
- Management of LCM related faults coming from the VNF

Concerning the former, a full new dedicated set of test cases (see Clause 8.3) has been introduced in the 3rd NFV Plugtests Test Plan in support of NSs with SFC implemented through NSH.

For the second, new scaling test descriptions were contributed by participants to cover NS and VNF scale to level operations taking into account scaling levels and aspects, as described in latest NFV specs.

Finally, in line with the 2nd NFV Plugtests Report outcomes [2NFVPLU-R], and to the subsequent discussions within NFV IFA WG, new test cases have been added to cover the management of VNF faults not directly related to virtualised resources, and / or coming from lifecycle aspects (e.g. VNF not alive and not reachable from VNFM). It is worth to mention that these new FM test cases do not cover purely application related VNF faults.

In summary, the following table summarizes those 3rd NFV Plugtests new or updated test descriptions with respect to the 2nd NFV Plugtests Test Plan [2NFVPLU-TP].

Test Description ID	Feature	Modification	3rd NFV Plugtests Test Plan Clause
TD_NFV_MULTIVENDOR_FM_VR_ALARM_001 TD_NFV_MULTIVENDOR_FM_VR_CLEAR_001	FM	Updated	6.1.8.1
TD_NFV_MULTIVENDOR_PM_VR_CREATE_MONITOR_001 TD_NFV_MULTIVENDOR_PM_VR_CREATE_THRESHOLD_001 TD_NFV_MULTIVENDOR_PM_VR_DELETE_MONITOR_001 TD_NFV_MULTIVENDOR_PM_VR_DELETE_THRESHOLD_001	PM	Updated	6.1.9.1
TD_NFV_MULTIVENDOR_PM_VNF_KPI_CREATE_MONITOR_001 TD_NFV_MULTIVENDOR_PM_VNF_KPI_DELETE_MONITOR_001 TD_NFV_MULTIVENDOR_PM_VNF_KPI_CREATE_THRESHOLD_001 TD_NFV_MULTIVENDOR_PM_VNF_KPI_DELETE_THRESHOLD_001	PM	New	6.1.9.2
TD_NFV_S-VNFM-D_FM_VNF_VR_ALARM_001 TD_NFV_S-VNFM-D_FM_VNF_VR_CLEAR_001	FM	Updated	6.5.5.2
TD_NFV_S-VNFM-D_PM_VNF_VR_CREATE_MONITOR_001 TD_NFV_S-VNFM-D_PM_VNF_VR_CREATE_THRESHOLD_001 TD_NFV_S-VNFM-D_PM_VNF_VR_DELETE_MONITOR_001 TD_NFV_S-VNFM-D_PM_VNF_VR_DELETE_THRESHOLD_001	PM	Updated	6.5.4.1
TD_NFV_S-VNFM-D_PM_VNF_KPI_CREATE_MONITOR_001 TD_NFV_S-VNFM-D_PM_VNF_KPI_DELETE_MONITOR_001	PM	Updated	6.5.4.2
TD_NFV_S-VNFM-D_PM_VNF_KPI_CREATE_THRESHOLD_001 TD_NFV_S-VNFM-D_PM_VNF_KPI_DELETE_THRESHOLD_001	PM	New	6.5.4.2
TD_NFV_SFC_NS_LCM_INSTANTIATE_001	NS LCM	New	6.3.1.1
TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_001 TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_002 TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_003 TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_VNF_001 TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_VNF_002 TD_NFV_SCALE-LEVEL_NS_LCM_SCALE_TO_LEVEL_VNF_003	NS LCM	New	6.1.5 6.1.6
TD_NFV_MULTIVENDOR_FM_VNF_ALARM_001 TD_NFV_MULTIVENDOR_FM_VNF_CLEAR_001	FM	New	6.1.8.2
TD_NFV_S-VNFM-D_FM_VNF_ALARM_001 TD_NFV_S-VNFM-D_FM_VNF_CLEAR_001	FM	New	6.5.5.2

Table 37. Input to TST007

As in previous NFV Plugtests, the test plan will be contributed to ETSI NFV TST WG as input for the revision of TST007 Guidelines for Interoperability Testing.

12.3.2 API Validation Test Plan – TST010

The definition of the Test Plan for the API Track in the 3rd NFV Plugtests was based on the learnings and the test plan developed for the 2nd NFV Plugtests.

The main additional features of the new test plan were:

- Extending the APIs (i.e. NSD Management) and operations (e.g. VNF instance instantiation and termination) in scope,
- Automated check of message payloads via schema validation and
- More rigorous checks on notifications triggered by the operation under test.

The API validation track already highlighted several parts of the specifications that are prone to misinterpretation and implementation errors. This indicates that – even rather simple - syntactic checks on the API operations are already very valuable to provide insights on some potentially critical issues and raise awareness within the organizations participating to the API testing.

The feedback collected from participants and the growing interest and adoption of NFV APIs encourage the development of further API testing activities in the context of future NFV Plugtests.

The learnings of this track will be contributed to ETSI NFV as input to TST010 NFV Conformance Testing, which in the future will also help to improve, evolve and / or complement the API validation track with NFV conformance testing.

12.4 Feedback on IOP Test Automation

During this Plugtests, one of the key areas of improvement was the automation and repeated execution of a subset of the interoperability Test Plan, with over 1000 test runs and an overall duration of 157 hours. A detailed analysis of the results allowed to draw some conclusions:

- During repeated runs, sporadic test failures can be observed. These failures are often due to instabilities in the test environment, especially when a high number of simultaneous test sessions are being run by multiple participants, which is often the case during a Plugtests event.
- During the interop test session, basic functional validation is achieved by sending end-to-end traffic through the NS. Here, most commonly observed problems were related to traffic not flowing or packet loss, which could be caused by several reasons, including but not only, the mentioned instability.

It was observed that test environment instability could lead to an increase of the duration of some lifecycle operations (e.g. instantiation could take 3 times longer than usual, reaching over 10 minutes in some critical cases).

To illustrate the above considerations, the following table lists the minimum, maximum and delta for the instantiation time, measured in different test session runs. Each row corresponds to a Test Session featuring a unique combination of MANO, VIM and multivendor NS:

Test Session	Min [sec]	Max [sec]	Delta [sec]	# of runs
1	63	190	127	103
2	680	1197	517	53
3	505	888	383	51
4	98	598	500	43

5	412	540	128	37
6	490	1670	1180	36
7	508	734	226	33
8	64	153	89	32
9	694	902	208	30
10	549	702	153	30
11	75	158	83	28
12	402	702	300	22
13	72	238	166	17
14	64	195	131	15
15	530	622	92	14
16	211	230	19	8
17	114	171	57	8
18	89	132	43	6
19	120	120	-	1
20	879	879	-	1
21	164	164	-	1
22	1010	1010	-	1

Table 36. Measures times in automated test sessions

Some specific factors were identified as contributing to test environment instability:

- HIVE network congestion, often caused by massive VNF packages upload/download. To avoid network congestion, participants were instructed to avoid such operations during the test sessions and prepare the testing environment outside of the testing hours.
- Overload of NFV Platforms. While thanks to the learnings of previous Plugtests events, the platforms were adequately dimensioned to support many parallel test sessions, it is extremely important to terminate the test sessions gracefully and free the allocated resources to obtain consistent results in the next session. Once again, participants were instructed to clean up and bring back the systems under test to their initial status at the end of the test sessions.

Finally, to overcome these factors, automated testing was also scheduled overnight.

Overall, it was outlined that repeated automated testing helps to reveal behaviour inconsistencies or anomalies, which can go unnoticed in manual testing.

Annex A – Results per Test Case

	Interoperability		Not Executed	Totals	
	OK	NO	NA	Run	Results
TD_NFV_SINGLEVENDOR_ONBOARD_VNF_PKG_001	4 (100.0%)	0 (0.0%)	0 (0.0%)	4 (100.0%)	4
TD_NFV_SINGLEVENDOR_ONBOARD_NSD_001	4 (100.0%)	0 (0.0%)	0 (0.0%)	4 (100.0%)	4
TD_NFV_SINGLEVENDOR_NS_LCM_INSTANTIATE_001	4 (100.0%)	0 (0.0%)	0 (0.0%)	4 (100.0%)	4
TD_NFV_MULTIVENDOR_ONBOARD_VNF_PKG_001	51 (100.0%)	0 (0.0%)	10 (16.4%)	51 (83.6%)	61
TD_NFV_MULTIVENDOR_ONBOARD_NSD_001	62 (100.0%)	0 (0.0%)	0 (0.0%)	62 (100.0%)	62
TD_NFV_MULTIVENDOR_NS_LCM_INSTANTIATE_001	62 (98.4%)	1 (1.6%)	0 (0.0%)	63 (100.0%)	63
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_001	13 (100.0%)	0 (0.0%)	42 (76.4%)	13 (23.6%)	55
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_001	13 (100.0%)	0 (0.0%)	42 (76.4%)	13 (23.6%)	55
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_002a	0 (0.0%)	1 (100.0%)	12 (92.3%)	1 (7.7%)	13
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_002a	0 (0.0%)	1 (100.0%)	11 (91.7%)	1 (8.3%)	12
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_002b	4 (80.0%)	1 (20.0%)	7 (58.3%)	5 (41.7%)	12
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_002b	4 (100.0%)	0 (0.0%)	7 (63.6%)	4 (36.4%)	11
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_003	0 (0.0%)	1 (100.0%)	2 (66.7%)	1 (33.3%)	3
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_003	0 (0.0%)	1 (100.0%)	2 (66.7%)	1 (33.3%)	3
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_004	1 (100.0%)	0 (0.0%)	7 (87.5%)	1 (12.5%)	8
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_004	1 (100.0%)	0 (0.0%)	7 (87.5%)	1 (12.5%)	8
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_VNF_001	9 (90.0%)	1 (10.0%)	20 (66.7%)	10 (33.3%)	30
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_VNF_001	9 (100.0%)	0 (0.0%)	21 (70.0%)	9 (30.0%)	30
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_VNF_002a	1 (50.0%)	1 (50.0%)	10 (83.3%)	2 (16.7%)	12
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_VNF_002a	1 (50.0%)	1 (50.0%)	10 (83.3%)	2 (16.7%)	12
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_VNF_002b	4 (80.0%)	1 (20.0%)	7 (58.3%)	5 (41.7%)	12
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_VNF_002b	4 (80.0%)	1 (20.0%)	7 (58.3%)	5 (41.7%)	12
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_VNF_003	0 (0.0%)	1 (100.0%)	7 (87.5%)	1 (12.5%)	8
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_VNF_003	0 (0.0%)	1 (100.0%)	7 (87.5%)	1 (12.5%)	8
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_OUT_VNF_004	1 (100.0%)	0 (0.0%)	6 (85.7%)	1 (14.3%)	7
TD_NFV_MULTIVENDOR_NS_LCM_SCALE_IN_VNF_004	1 (100.0%)	0 (0.0%)	6 (85.7%)	1 (14.3%)	7

TD_NFV_MULTIVENDOR_NS_LCM_UPDATE_STOP_VNF_001	27 (93.1%)	2 (6.9%)	28 (49.1%)	29 (50.9%)	57
TD_NFV_MULTIVENDOR_NS_LCM_UPDATE_START_VNF_001	27 (90.0%)	3 (10.0%)	27 (47.4%)	30 (52.6%)	57
TD_NFV_MULTIVENDOR_PM_VR_CREATE_MONITOR_001	11 (64.7%)	6 (35.3%)	13 (43.3%)	17 (56.7%)	30
TD_NFV_MULTIVENDOR_PM_VR_CREATE_THRESHOLD_001	11 (68.8%)	5 (31.3%)	14 (46.7%)	16 (53.3%)	30
TD_NFV_MULTIVENDOR_PM_VR_DELETE_MONITOR_001	12 (70.6%)	5 (29.4%)	13 (43.3%)	17 (56.7%)	30
TD_NFV_MULTIVENDOR_PM_VR_DELETE_THRESHOLD_001	11 (68.8%)	5 (31.3%)	14 (46.7%)	16 (53.3%)	30
TD_NFV_MULTIVENDOR_PM_VNF_KPI_CREATE_MONITOR_001	5 (71.4%)	2 (28.6%)	11 (61.1%)	7 (38.9%)	18
TD_NFV_MULTIVENDOR_PM_VNF_KPI_DELETE_MONITOR_001	5 (71.4%)	2 (28.6%)	11 (61.1%)	7 (38.9%)	18
TD_NFV_MULTIVENDOR_PM_VNF_KPI_CREATE_THRESHOLD_001	6 (75.0%)	2 (25.0%)	10 (55.6%)	8 (44.4%)	18
TD_NFV_MULTIVENDOR_PM_VNF_KPI_DELETE_THRESHOLD_001	5 (71.4%)	2 (28.6%)	11 (61.1%)	7 (38.9%)	18
TD_NFV_MULTIVENDOR_FM_VR_ALARM_001	5 (62.5%)	3 (37.5%)	22 (73.3%)	8 (26.7%)	30
TD_NFV_MULTIVENDOR_FM_VR_CLEAR_001	3 (50.0%)	3 (50.0%)	24 (80.0%)	6 (20.0%)	30
TD_NFV_MULTIVENDOR_FM_VNF_ALARM_001	8 (72.7%)	3 (27.3%)	25 (69.4%)	11 (30.6%)	36
TD_NFV_MULTIVENDOR_FM_VNF_CLEAR_001	8 (72.7%)	3 (27.3%)	24 (68.6%)	11 (31.4%)	35
TD_NFV_MULTIVENDOR_NS_LCM_TERMINATE_001	60 (98.4%)	1 (1.6%)	5 (7.6%)	61 (92.4%)	66
TD_NFV_MULTIVENDOR_TEARDOWN_DELETE_NSD_001	61 (100.0%)	0 (0.0%)	5 (7.6%)	61 (92.4%)	66
TD_NFV_MULTIVENDOR_TEARDOWN_DELETE_VNF_PKG_001	51 (100.0%)	0 (0.0%)	15 (22.7%)	51 (77.3%)	66
TD_NFV_MULTISITE_ONBOARD_VNF_PKG_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_MULTISITE_ONBOARD_NSD_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_MULTISITE_NS_LCM_INSTANTIATE_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_MULTISITE_PM_VR_CREATE_MONITOR_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_MULTISITE_PM_VR_CREATE_THRESHOLD_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_MULTISITE_PM_VR_DELETE_MONITOR_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_MULTISITE_PM_VR_DELETE_THRESHOLD_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_MULTISITE_NS_LCM_TERMINATE_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_MULTISITE_TEARDOWN_DELETE_NSD_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_MULTISITE_TEARDOWN_DELETE_VNF_PKG_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_EPA_ONBOARD_VNF_PKG_001	3 (100.0%)	0 (0.0%)	0 (0.0%)	3 (100.0%)	3
TD_NFV_EPA_ONBOARD_NSD_001	3 (100.0%)	0 (0.0%)	0 (0.0%)	3 (100.0%)	3
TD_NFV_EPA_NS_LCM_INSTANTIATE_001	3 (100.0%)	0 (0.0%)	0 (0.0%)	3 (100.0%)	3

TD_NFV_EPA_NS_LCM_TERMINATE_001	3 (100.0%)	0 (0.0%)	0 (0.0%)	3 (100.0%)	3
TD_NFV_EPA_TEARDOWN_DELETE_NSD_001	3 (100.0%)	0 (0.0%)	0 (0.0%)	3 (100.0%)	3
TD_NFV_EPA_TEARDOWN_DELETE_VNF_PKG_001	3 (100.0%)	0 (0.0%)	0 (0.0%)	3 (100.0%)	3
TD_NFV_AUTOLCMV_ONBOARD_NSD_001	22 (100.0%)	0 (0.0%)	0 (0.0%)	22 (100.0%)	22
TD_NFV_AUTOLCMV_NS_LCM_INSTANTIATE_001	17 (77.3%)	5 (22.7%)	0 (0.0%)	22 (100.0%)	22
TD_NFV_AUTOLCMV_NS_LCM_SCALE_OUT_001	3 (42.9%)	4 (57.1%)	15 (68.2%)	7 (31.8%)	22
TD_NFV_AUTOLCMV_NS_LCM_SCALE_IN_001	5 (71.4%)	2 (28.6%)	15 (68.2%)	7 (31.8%)	22
TD_NFV_AUTOLCMV_NS_LCM_SCALE_OUT_VNF_001	10 (83.3%)	2 (16.7%)	3 (20.0%)	12 (80.0%)	15
TD_NFV_AUTOLCMV_NS_LCM_SCALE_IN_VNF_001	7 (63.6%)	4 (36.4%)	4 (26.7%)	11 (73.3%)	15
TD_NFV_AUTOLCMV_NS_LCM_UPDATE_STOP_VNF_001	9 (75.0%)	3 (25.0%)	10 (45.5%)	12 (54.5%)	22
TD_NFV_AUTOLCMV_NS_LCM_UPDATE_START_VNF_001	8 (66.7%)	4 (33.3%)	10 (45.5%)	12 (54.5%)	22
TD_NFV_AUTOLCMV_FM_VR_ALARM_001	8 (72.7%)	3 (27.3%)	11 (50.0%)	11 (50.0%)	22
TD_NFV_AUTOLCMV_FM_VR_CLEAR_001	8 (72.7%)	3 (27.3%)	11 (50.0%)	11 (50.0%)	22
TD_NFV_AUTOLCMV_NS_LCM_TERMINATE_001	16 (84.2%)	3 (15.8%)	3 (13.6%)	19 (86.4%)	22
TD_NFV_AUTOLCMV_TEARDOWN_DELETE_NSD_001	22 (100.0%)	0 (0.0%)	0 (0.0%)	22 (100.0%)	22
TD_NFV_SFC_ONBOARD_VNF_PKG_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_SFC_ONBOARD_NSD_001	0 (0.0%)	1 (100.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_SFC_NS_LCM_INSTANTIATE_001	0 (0.0%)	1 (100.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_SFC_NS_LCM_TERMINATE_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_SFC_TEARDOWN_DELETE_NSD_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_SFC_TEARDOWN_DELETE_VNF_PKG_001	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_NFV_API_1_VNFM_LCM	2 (33.3%)	4 (66.7%)	0 (0.0%)	6 (100.0%)	6
TD_NFV_API_2_VNFM_LCM	2 (33.3%)	4 (66.7%)	0 (0.0%)	6 (100.0%)	6
TD_NFV_API_3_VNFM_LCM	2 (33.3%)	4 (66.7%)	0 (0.0%)	6 (100.0%)	6
TD_NFV_API_4_VNFM_LCM	4 (100.0%)	0 (0.0%)	2 (33.3%)	4 (66.7%)	6
TD_NFV_API_5_VNFM_LCM	1 (100.0%)	0 (0.0%)	5 (83.3%)	1 (16.7%)	6
TD_NFV_API_6_VNFM_LCM	3 (100.0%)	0 (0.0%)	3 (50.0%)	3 (50.0%)	6
TD_NFV_API_7_VNFM_LCM	4 (66.7%)	2 (33.3%)	0 (0.0%)	6 (100.0%)	6
TD_NFV_API_8_VNFM_LCM	3 (100.0%)	0 (0.0%)	3 (50.0%)	3 (50.0%)	6

TD_NFV_API_9_VNFM_LCM	5 (100.0%)	0 (0.0%)	1 (16.7%)	5 (83.3%)	6
TD_NFV_API_10_VNFM_LCM	4 (80.0%)	1 (20.0%)	1 (16.7%)	5 (83.3%)	6
TD_NFV_API_11_VNFM_LCM	5 (100.0%)	0 (0.0%)	1 (16.7%)	5 (83.3%)	6
TD_NFV_API_12_VNFM_LCM	5 (100.0%)	0 (0.0%)	1 (16.7%)	5 (83.3%)	6
TD_NFV_API_20_VNF_CONF	2 (100.0%)	0 (0.0%)	0 (0.0%)	2 (100.0%)	2
TD_NFV_API_21_VNF_CONF	1 (50.0%)	1 (50.0%)	0 (0.0%)	2 (100.0%)	2
TD_NFV_API_22_VNF_CONF	0 (0.0%)	1 (100.0%)	1 (50.0%)	1 (50.0%)	2
TD_NFV_API_23_VNF_IND	1 (50.0%)	1 (50.0%)	0 (0.0%)	2 (100.0%)	2
TD_NFV_API_24_VNF_IND	1 (50.0%)	1 (50.0%)	0 (0.0%)	2 (100.0%)	2
TD_NFV_API_25_VNF_IND	2 (100.0%)	0 (0.0%)	0 (0.0%)	2 (100.0%)	2
TD_NFV_API_26_VNF_IND	1 (50.0%)	1 (50.0%)	0 (0.0%)	2 (100.0%)	2
TD_NFV_API_27_VNF_IND	1 (50.0%)	1 (50.0%)	0 (0.0%)	2 (100.0%)	2
TD_NFV_API_28_VNF_IND	1 (50.0%)	1 (50.0%)	0 (0.0%)	2 (100.0%)	2
TD_NFV_API_29_VNF_IND	1 (100.0%)	0 (0.0%)	1 (50.0%)	1 (50.0%)	2
TD_NFV_API_30_VNFM_LCM	3 (100.0%)	0 (0.0%)	3 (50.0%)	3 (50.0%)	6
TD_NFV_API_31_VNFM_LCM	1 (100.0%)	0 (0.0%)	5 (83.3%)	1 (16.7%)	6
TD_API_32_NFVO_NSDM	0 (0.0%)	1 (100.0%)	0 (0.0%)	1 (100.0%)	1
TD_API_33_NFVO_NSDM	0 (0.0%)	1 (100.0%)	0 (0.0%)	1 (100.0%)	1
TD_API_34_NFVO_NSDM	0 (0.0%)	1 (100.0%)	0 (0.0%)	1 (100.0%)	1
TD_API_35_NFVO_NSDM	0 (0.0%)	0 (0.0%)	1 (100.0%)	0 (0.0%)	1
TD_API_36_NFVO_NSDM	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_API_37_NFVO_NSDM	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1
TD_API_38_NFVO_NSDM	1 (100.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)	1

History

Document history		
V1.0.0	30/08/2018	Publication