ITUPublications
Recommendations

International Telecommunication Union
Standardization Sector

Recommendation

# ITU-T Q.4045 (12/2023)

SERIES Q: Switching and signalling, and associated measurements and tests

Testing specifications – Testing specifications for Cloud computing

# Framework of network function virtualization automated testing

ITU-T Q-SERIES RECOMMENDATIONS

**Switching and signalling, and associated measurements and tests**

| | |
|---|---|
| SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE | Q.1-Q.3 |
| INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING | Q.4-Q.59 |
| FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN | Q.60-Q.99 |
| CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS | Q.100-Q.119 |
| SPECIFICATIONS OF SIGNALLING SYSTEMS NO. 4, 5, 6, R1 AND R2 | Q.120-Q.499 |
| DIGITAL EXCHANGES | Q.500-Q.599 |
| INTERWORKING OF SIGNALLING SYSTEMS | Q.600-Q.699 |
| SPECIFICATIONS OF SIGNALLING SYSTEM NO. 7 | Q.700-Q.799 |
| Q3 INTERFACE | Q.800-Q.849 |
| DIGITAL SUBSCRIBER SIGNALLING SYSTEM NO. 1 | Q.850-Q.999 |
| PUBLIC LAND MOBILE NETWORK | Q.1000-Q.1099 |
| INTERWORKING WITH SATELLITE MOBILE SYSTEMS | Q.1100-Q.1199 |
| INTELLIGENT NETWORK | Q.1200-Q.1699 |
| SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000 | Q.1700-Q.1799 |
| SPECIFICATIONS OF SIGNALLING RELATED TO BEARER INDEPENDENT CALL CONTROL (BICC) | Q.1900-Q.1999 |
| BROADBAND ISDN | Q.2000-Q.2999 |
| SIGNALLING REQUIREMENTS AND PROTOCOLS FOR THE NGN | Q.3000-Q.3709 |
| SIGNALLING REQUIREMENTS AND PROTOCOLS FOR SDN | Q.3710-Q.3899 |
| TESTING SPECIFICATIONS | Q.3900-Q.4099 |
| Testing specifications for next generation networks | Q.3900-Q.3999 |
| Testing specifications for SIP-IMS | Q.4000-Q.4039 |
| **Testing specifications for Cloud computing** | **Q.4040-Q.4059** |
| Testing specifications for IMT-2020 and IoT | Q.4060-Q.4099 |
| PROTOCOLS AND SIGNALLING FOR PEER-TO-PEER COMMUNICATIONS | Q.4100-Q.4139 |
| PROTOCOLS AND SIGNALLING FOR COMPUTING POWER NETWORKS | Q.4140-Q.4159 |
| PROTOCOLS AND SIGNALLING FOR QUANTUM KEY DISTRIBUTION NETWORKS | Q.4160-Q.4179 |
| SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2020 | Q.5000-Q.5049 |
| COMBATING COUNTERFEITING AND STOLEN ICT DEVICES | Q.5050-Q.5069 |

*For further details, please refer to the list of ITU-T Recommendations.*

# Recommendation ITU-T Q.4045

## Framework of network function virtualization automated testing


### Summary

Recommendation ITU-T Q.4045 provides an overview and framework requirements of network function virtualization (NFV) automated testing. This Recommendation introduces the NFV automated testing framework, and provides an overview of NFV automated testing and design considerations. In addition, the framework and requirements for NFV automated testing are derived based on the use cases presented Appendix I.



### History*

| Edition | Recommendation | Approval | Study Group | Unique ID |
|---|---|---|---|---|
| 1.0 | ITU-T Q.4045 | 2023-12-14 | 11 | 11.1002/1000/15721 |

### Keywords

Automated testing, framework, NFV.


* To access the Recommendation, type the URL https://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID.

## FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

## NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

## INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

## Table of Contents

# Recommendation ITU-T Q.4045

## Framework of network function virtualization automated testing

## 1 Scope

This Recommendation provides the framework of network function virtualization (NFV) automated testing. It addresses the following subjects:

– Overview of NFV automated testing;

– Design considerations of NFV automated testing;

– NFV automated testing framework;

– Framework requirements of NFV automated testing;

– Typical use cases to derive framework requirements.

## 2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T H.644.2] Recommendation ITU-T H.644.2 (2019), *Virtual content delivery network – Network virtualization*.

[ITU-T Q.3719] Recommendation ITU-T Q.3719 (2019), *Signalling requirements for the separation of control plane and user plane in a virtualized broadband network gateway (vBNG)*.

[ITU-T Q.4043] Recommendation ITU-T Q.4043 (2019), *Interoperability testing requirements of a virtual switch*.

[ITU-T Y.2320] Recommendation ITU-T Y.2320 (2015), *Requirements for virtualization of control network entities in next generation network evolution*.

[ITU-T Y.3150] Recommendation ITU-T Y.3150 (2020), *High-level technical characteristics of network softwarization for IMT-2020*.

## 3 Definitions

### 3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

**3.1.1 network functions virtualization (NFV)** [b-ETSI GS NFV 003]: Principle of separating network functions from the hardware they run on by using virtual hardware abstraction.

**3.1.2 test case** [b-ITU-T Y.4500.15]: Specification of the actions required to achieve a specific test purpose, starting in a stable testing state, ending in a stable testing state and defined in either natural language for manual operation or in a machine-readable language (such as testing and test control notation version 3 (TTCN-3)) for automatic execution.

**3.1.3 test object** [b-ITU-T X.745]: A managed object that exists only for a controlled test invocation and which has attributes, operations and notifications that pertain to that instance of test.

**3.1.4 test script** [b-ITU-T M.3710]: A service-specific software unit that can perform the test operations of a particular service automatically.

## 3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

**3.2.1 test case template**: A structured description of a test case that organizes the unstructured data of the test case into a standardized data structure to simplify the conversion from test case to test script.

**3.2.2 test entity**: An object in the test script that represents the software or hardware related to a test.

**3.2.3 test orchestration**: A process that sorts the execution order of a set of test cases and interactions with the test tools for carrying out certain test cases in an automated manner.

**3.2.4 test scheme template**: A combination of test cases and relevant test parameters for designing the testing activities required to accomplish specific testing objectives from a technical perspective.

**3.2.5 test task**: The instantiation of a test scheme template, created by the tester, which is used to perform specific test activities.

# 4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

| | |
|---|---|
| 5GC | 5G Core |
| API | Application Programming Interface |
| BRAS | Broadband Remote Access Server |
| CD | Continues Delivery |
| CI | Continues Integration |
| CSC | Cloud Service Customer |
| CSP | Cloud Service Provider |
| CLI | Command Line Interface |
| E-CORD | Enterprise- Central Office Re-Architected as a Datacentre |
| ID | Identification |
| IPv4 | Internet Protocol version 4 |
| IPv6 | Internet Protocol version 6 |
| IT | Information Technology |
| LCM | Lifecycle Management |
| MANO | Management and Orchestration |
| MTU | Maximum Transmission Unit |
| NFV | Network Function Virtualization |
| NFVAT | NFV Automated Testing |
| NF | Network Function |
| NFVI | Network Function Virtualization Infrastructure |

| | |
|---|---|
| NFVO | Network Function Virtualization Orchestration |
| NGNe | Next Generation Network Evolution |
| NS | Network Service |
| NSD | Network Services Descriptor |
| ONAP | Open Network Automation Platform |
| OSM | Open Source MANO |
| SD-WAN | Software-Defined Wide Area Network |
| vCDN | Virtual Content Delivery Network |
| vCPE | Virtualized Customer Premises Equipment |
| vEPC | Virtual Evolved Packet Core |
| VIM | Virtual Infrastructure Management |
| VNF | Virtualization Network Function |
| VNFD | Virtualized Network Function Descriptor |
| VNFM | Virtualization Network Function Management |
| VOLTE | Voice Over LTE |
| XOS | X-Operating System |

## 5 Conventions

In this Recommendation:

The keywords "is required" indicate a requirement which must be strictly followed and from which no deviation is permitted, if conformance to this Recommendation is to be claimed.

The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, this requirement need not be present to claim conformance.

## 6 Overview of NFV automated testing

### 6.1 NFV and NFV architecture

Network function virtualization (NFV) is a technology which uses software to install, control and operate the network functions running on general hardware. The goal of NFV is to replace the private, specialized and closed network functions in the telecommunication network, and to realize a unified and universal software open architecture.

Future networks are evolving with NFV for better flexibility and scalability. A lot of this work has been done in the International Telecommunication Union (ITU):

– [ITU-T Y.2320] provides the requirements for virtualization of control network entities in next generation network evolution (NGNe), including the requirements of virtual infrastructures, virtualized network entities, and management systems.

– [ITU-T Y.3150] introduces NFV as a key component for network softwarization in IMT-2020.

– [ITU-T H.644.2] specifies the functional architecture, related functions and functional blocks, and high-level reference points which implement content delivery network virtualization by utilizing networking virtualization technologies.

– [ITU-T Q.3719] describes a virtualized broadband network gateway with separated control plane and user plane, which is used to either augment or replace the existing traditional broadband remote access server (BRAS).

– [ITU-T Q.4043] introduces the virtual switch and its interoperability requirements.

The underlying technology logic of NFV depends on virtualization technology. Virtualization technology can decompose general computing, storage, network and other hardware devices into a variety of virtual resources. This facilitates hardware decoupling and allows for the dynamic and flexible deployment of network functions in accordance with specific needs.

Figure 6-1 shows the NFV high-level architecture, including network function virtualization infrastructure (NFVI), management and orchestration (MANO) and virtualization network function (VNF).

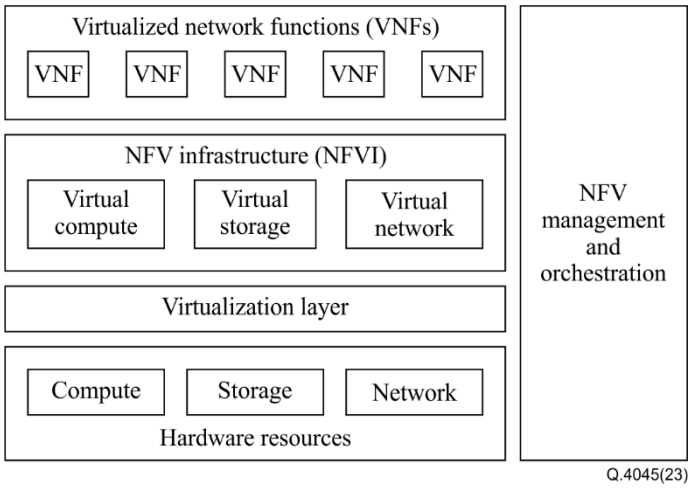



**Figure 6-1 – High-level NFV architecture**

Sourced from [b-ETSI GS NFV 002]

NFVI includes the virtualization layer and hardware resources, which can be deployed across several physical locations. All virtual resources should be in a unified shared resource pool. VNFs run on top of NFVI, which provides some kinds of network services deployed in virtual machines, containers or bare-metal servers by using the infrastructure provided by NFVI. MANO provides the overall management and orchestration of NFV, which is composed of network function virtualization orchestration (NFVO), virtualization network function management (VNFM) and virtual infrastructure management (VIM). They manage, respectively the network service (NS) life-cycle, the VNF life-cycle and the NFVI.

## 6.2 NFV automated testing

### 6.2.1 Challenges in NFV testing

The three main working layers of NFV are identified as the VNF layer, the NFVI layer, and the NFV MANO layer. Each layer is composed of multi-components provided by different vendors. Each component depends on and calls on each other. Because many implementations of components are private commercial solutions instead of open sourced, the compatibility and interoperability between them is poor, and the system integration complexity is high. In order to promote the application and implementation of NFV architecture, a lot of repetitive testing work needs to be carried out. Compared with traditional IT testing, NFV testing has the following challenges:

– With more components involved, the testing environment is more complicated. Upgrades in each component may lead to changes in the testing environment, requiring re-testing. At the

same time, the iteration cycle of NFV network elements is shorter, which then needs a higher testing frequency.

– Adoption of NFV has cloud-based elastic resource management issues, orchestration issues, operation and maintenance issues, as well as interoperability issues caused by layer decoupling. Consequently, the test content and test scope are larger.

Testing carried out manually is inefficient and unreliable. There is a need to develop NFV automated testing tools to improve the efficiency and reliability of NFV testing. Test cases in automated testing are usually developed before test execution in the format of scripts which can be executed repeatedly to reduce testing costs. It is also easy to fix the script flaws after they are discovered and modify the script based on different test scenarios. Additionally, automated testing can take advantage of a lot of existing testing tools and executes test tasks in parallel.

### 6.2.2 Test dimensions in NFV automated testing

NFV automated testing should support the following test dimensions for baseline, security and assurance so that basic goals of solution validation, interoperability, verification of service layer agreement (SLA)/assurance, and deployment readiness can be validated.

Test dimensions for VNF:

– Basic validation – verification of core services, lifecycle management (LCM) operations, and onboarding.

– Functionality – verification of basic functionality, after reboot, behaviour with virtual infrastructure management (VIM) network fault, behaviour when VIM controller is restarted, VIM network restarted, and support for IPv6-Only (test by turning off IPv4 on the interfaces).

– Performance – verification based on different resource configurations, max throughput, instance creation/deletion time, forwarding performance benchmarking using different traffic types, packet/frame size, and load tests.

– Scale – verification of multiple instances of VNF, max number of VNF, and scale-out/in.

– VNFM interactions – verification of integrity of VNFD/NSD, instantiation, placement, destruction, resilience/recovery, monitoring, backup/restore, updates/roll-back, LCM after rebooting the VNF, and VNF config injection/reconfig time.

– NFVO interactions – verification of VNF deployment using available network services descriptors (NSDs), live migration, and fault management and logging.

– Service chain.

Test dimensions for NFVI:

– NFVI stack installation.

– Fault management and log collection.

– MTU changes.

– Platform awareness.

– NFVI under loaded conditions.

– Support for IPv6-only (test by turning off IPv4 on the interfaces).

– Support for service chain.

Test dimensions for MANO:

– VNF types.

– Lifecycle management – verification of VNFD/VNF image onboarding, deployment using supported flavour, and instantiation/destruction/resilience/recovery/monitoring/updates/rollbacks.

– Various VIM infrastructure.
– VNF placement/scaling.
– NFVO fault management and log collection.
– Different types of NFVO (component testing of VNFM).
– End-to-end service orchestration.
– Different types of VNFM (component testing of NFVO).
– Different types of VIM (component testing of NFVO).
– Network service instantiation.

Broad test dimensions to cover interoperability, compatibility, assurance, service orchestration, and deployment readiness:

– VNF onboarding.
– VNF config upgrades.
– Data plane performance.
– Fault and healing.
– Functionality.
– Service instantiation.
– NFV service configuration.
– Control plane scaling.
– Placement constraints.
– Use-case specific test cases.

# 7 Design consideration of NFV automated testing

## 7.1 Role consideration for NFV automated testing

There are three roles involved in NFV automated testing, test designer, script developer and tester:

– NFV test designer: the person who is responsible for the design of NFV test plan. Based on the test purpose, the role involves identifying what test cases need to be conducted with which test tools during the test, determining the execution order of test cases, and determining what parameters need to be configured. The NFV test designer in NFV automated testing is usually someone with professional NFV background and rich test experience.
– Script developer: the person who is responsible for developing of test scripts based on the test cases designed by test designers. The script developer is usually a programmer with a background in computer or software development. There is no need for the script developer to have NFV related experience.
– NFV tester: the person who is responsible for conducting of NFV test, which involves choosing an appropriate test plan, setting up the test environment, executing test cases automatically and analysing testing results. The NFV tester is someone who is familiar with the NFV automated testing tool.

## 7.2 Support of single component test and scenario test

Three main working layers are identified in the NFV architecture: VNF, NFVI and NFV MANO. Each layer is composed of multiple components. It is necessary for a single component test to make sure it can function correctly. NFV automated testing needs to support single component tests.

Applications and implementations of NFV architecture usually include multiple components provided by various vendors from the three working layers. These components collaborate to enable different scenarios. Examples of NFV scenarios are: 5GC, VIMs, voice Over LTE (VOLTE), enterprise software-defined wide area network (SD-WAN), enterprise-central office re-architected as a datacentre (E-CORD), access vCPE, service provider virtual evolved packet core (vEPC), virtual content delivery network (vCDN), etc. It is essential for a specific scenario test to confirm whether these components are compatible with each other and whether they provide corresponding functions together. NFV automated testing needs to support scenario tests.

### 7.3 Support of reusability and flexibility

In order to promote the applications and implementations of NFV architecture in different locations, especially for telecommunication providers and manufacturers, many repetitive testing needs to be carried out. NFV automated testing needs to support test reusability to minimize redundant work, which can be achieved by using a template. Templating includes the templating of test schemes and the templating of test cases, which can be used directly or be customized based on different requirements. They are both concepts of the test design stage.

In addition, applications and implementations of NFV architecture may vary across different locations due to special local conditions. NFV automated testing should offer flexible scalability, enabling NFV testers to adjust test schemes, test cases and relevant test parameters in order to adapt to different conditions.

### 7.4 Support of test result reliability

Automated testing minimizes manual involvement, but ensuring the reliability of test results is crucial. However, the NFV system is complex, a test task will contain multiple complete test cases, and each test case contains multiple steps, each of which performs multiple operations on multiple test objects. Retaining the intermediate operating state becomes challenging, making it difficult to reproduce the exact state after the test, and compromising the reliability of the test results. To address this challenge a real-time test log during test execution, a test task stoppage to check the status, and a detailed test report after the test is completed, are needed.

## 8 NFV automated testing framework

### 8.1 Resource data model

The core elements of resource data models in NFV automated testing include the test scheme template, test case template, test task, test case, test tool, test parameter, test object and test report. The relationship among these elements is shown in Figure 8-1.

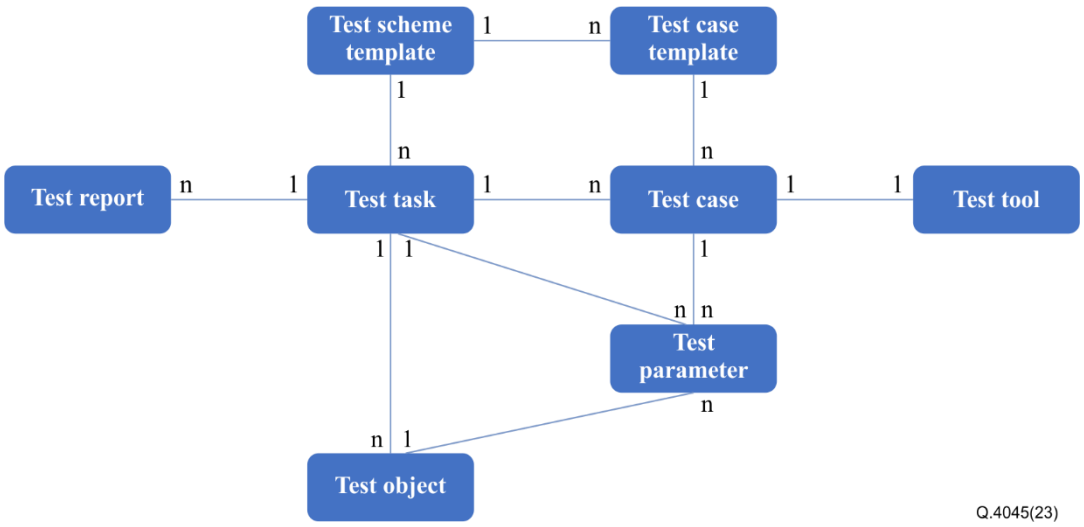



**Figure 8-1 – Resource data model in NFV automated testing**

A test scheme template refers to a template for designing the testing activities required to accomplish specific testing objectives from a technical perspective. A test scheme template combines test cases templates and relevant test parameters, which can be used directly or be customized based on different test objectives. A test case template serves as a structured description of a test case, organizing the unstructured data of the test case into a standardized data structure to simplify the conversion from test case to test script. A test task is the instantiation of a test scheme template while a test case is the instantiation of a test case template, both by filling in the parameters. A test object represents a managed object which has attributes, operations and notifications to the test task. A test tool is the hardware or software used to carry out or assist in carrying out the test case required. A test report is a document produced at the end of the test giving an overall summary of the test carried out.

## 8.2 NFV automated testing stages

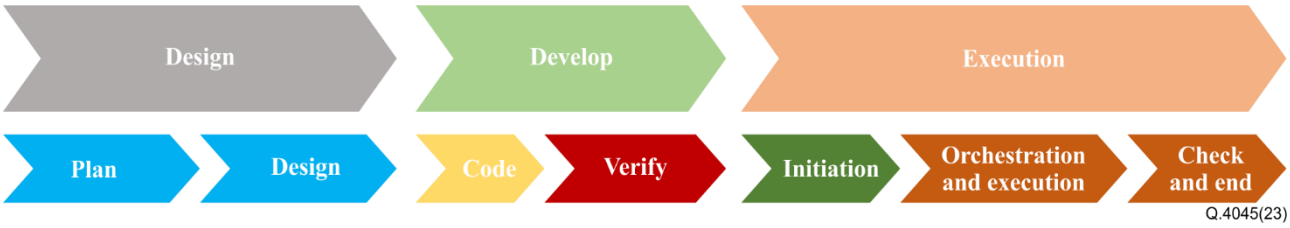



**Figure 8-2 – NFV automated testing stages**

The lifecycle of NFV automated testing includes 7 stages, where each stage is iterative. Each stage defined in Figure 8-2 is described as follows:

### 1) Plan

At this stage, the NFV test designer defines a test plan based on the test requirements provided by NFV tester. The test plan can be presented by test task description documents and test case description documents.

### 2) Design

At this stage, the NFV test designer designs what test cases need to be executed to complete the test scheme, what input should be set for each test case template, and the execution order of the test

cases based on the test plan. Templates are generated after the design is completed, and these templates follow the design considerations presented in clause 7.

**3) Code**

At this stage, the test developer develops the test script according to the test case template designed by the NFV test designer and the test parameters required by the test case.

**4) Verify**

At this stage, the test developer tests the usability of the test script and resolves any bugs found.

**5) Initiation**

At this stage, the NFV tester selects the required test scheme template and instantiates the test task to perform automated testing on the test objects.

**6) Orchestration and execution**

At this stage, the execution order of test cases is generated for the tester to adjust according to specific needs. Then the test tools will carry out those test cases in an automated manner.

**7) Check and end**

At this stage, the tester will get a test report produced by the system. The tester measures and analyses the report and then conducts acceptance of the test objects. If the acceptance conclusion is failed, new demands will be provided to the product development team.

# 9 Framework requirements of NFV automated testing

## 9.1 Requirements of NFVAT for test template support

It is required that NFV automated testing (NFVAT) provides test scheme templates and test case templates for NFV testers.

NOTE 1 – Test scheme template and test case template are the concepts of the test design state. Test scheme templates can be divided into component test scheme templates and scenario test scheme templates.

NOTE 2 – A component test scheme template of NFV aims at the testing of a single component which only belongs to one working layer in NFV architecture.

NOTE 3 – A scenario test scheme template of NFV aims at the testing of a specific scenario to confirm whether the functional components are compatible with each other and whether they have the ability to provide corresponding capabilities together in a specific scenario. An NFV scenario consists of multiple components from different NFV architecture layers.

It is required that NFVAT provides a test scheme template repository and a test case template repository to upload, store, update, and delete templates for NFV testers.

It is required that NFVAT generates test case templates according to the test information input by NFV testers.

NOTE 4 – Test case templates include test case description, inputs, preconditions, test steps, and expected results.

NOTE 5 – Test information input by NFV testers is the description of test steps and expected results.

NOTE 6 – Test case description includes test case ID, test case type, test object, and test case provider.

NOTE 7 – Inputs include the information required for script execution and the serialized byte sequence of the test entity.

NOTE 8 – Preconditions include test cases that this test case depends on, conditions that the test environment needs to meet, and scripts required to verify the conditions.

NOTE 9 – A test step includes step description, preconditions, step identifiers, and step details. The preconditions are used to identify whether the step can be executed. The step details are used to define the specific actions of each step.

NOTE 10 – Step details define the interaction type, entities involved in the test, interaction content, and output information. The type is command line interface (CLI) or application programming interface (API). The output information includes the log and the entity generated by the test step, and the result of the test step.

NOTE 11 – Expected results describe the final results that the test should obtain.

It is recommended that NFVAT generates test scripts based on the test case template.

## 9.2 Requirements of NFVAT for test task instantiation

It is required that NFVAT manages the life cycle of test tasks, including the creation, start, update, pause and deletion of test tasks.

It is required that NFVAT manages test parameters, which contains the parameters of test cases, and the parameters of test component/scenario.

NOTE 1 – Examples of test case parameters are the type of test case to indicate whether it is a functional test case or a performance test case, test tool which will carry out the test case, and test category parameter closed/semi-closed to identify whether this test case will pass data to other test cases.

NOTE 2 – Examples of test component/scenario parameters are the test object/objects of the test component/scenario, the execution order and number of all test cases, input and output data of all test cases.

It is required that NFVAT provides a user interface for NFV testers to create test tasks, search test task status and view test task results.

## 9.3 Requirements of NFVAT for test task orchestration and execution

It is required that NFVAT provides an orchestration engine to analysis test cases and form a process file recording the execution order of each test case in the test template.

It is required that NFVAT carries out test cases in the test scheme template according to the execution order in the process file.

It is recommended that NFVAT provides test detection points and supports troubleshooting during test task execution.

NOTE 1 – The test case script developer sets test detection points and develops test cases according to test requirements during the development of test case scripts.

NOTE 2 – Before testing, the NFV tester selects several expected pause positions from the test detection points.

NOTE 3 – During the execution of the test task, when it is executed to the detection point of the test case script, check whether the position of the detection point matches the expected pause position; If it matches, generate a test pause instruction and pause the test task. The tester shall inspect the intermediate state of the test. If it does not match, generate the test continue instruction and continue to execute the test task.

It is recommended that NFVAT supports the storing and viewing of the intermediate status of the test task execution.

It is required that NFVAT generates independent log files for the main script log, VIM log, NFVO log, VNFM log, VNF log for each test case, as applicable.

It is required that NFVAT monitors the test task execution metrics to reflect test status.

NOTE 4 – The metrics of NFV automated testing task are used by both NFV tester and operator to know whether a test task works properly. It usually includes resource-level metrics and test-level metrics.

NOTE 5 – Resource-level metrics are metrics of resources used during NFVAT, including compute/network/storage hardware metrics, virtualization layer metrics, virtual compute/network/storage

resource metrics, VNF metrics, etc. Each of the resource-level metrics can be divided into resource utilization rate, resource operation status, resource quantity, etc.

NOTE 6 – Test-level metrics are defined by the NFVAT designer, and vary from test task to test task. Taking the 5GC network function testing task as an example, metrics include session number, traffic throughput, etc.

It is required that NFVAT stores the history of test task failures during task execution.

It is recommended that NFVAT adaptively integrates with different testing tools and facilitates the transfer of test parameters across these tools.

It is recommended that NFVAT converts the test case data into test input data that conforms to the data specification of a specific testing tool.

It is recommended that NFVAT determines the specific testing tool that matches the test case based on the characteristics of the test case from multiple testing tools and invoke the specific testing tool to execute the test case using the testing tool's data specifications for the test input data.

NOTE 7 – The characteristics of the test case is extracted by analysing test cases.

NOTE 8 – The test cases and their associated data are retrieved from the orchestration engine.

It is required that NFVAT restores the initial state of the test environment until the NFV tester deletes the test task.

## 9.4 Requirements of NFVAT for test report generation

It is required that NFVAT generates a test report of each test task automatically for the NFV tester to download.

It is required that NFVAT provides information about the test task in the report.

NOTE 1 – Information includes stakeholders involved in the test, information about the test environment, all parameters set during the task, records of the execution process, problems found and, etc.

It is required NFVAT provides test result to indicate whether the test is passed, partially passed or failed.

It is recommended NFVAT scores the test based on test results.

It is recommended that NFVAT provides a setting on the storage time length of the test report, beyond which test reports would be automatically deleted.

It is recommended that NFVAT provides the ability to analyse the execution results of multiple test tasks from various dimensions and generates reports with statistical information.

NOTE 2 – The dimensions include a specific test object, a specific test case, a specific time period, etc.

## 9.5 Requirements of NFVAT for test tool management

It is recommended that NFVAT supports registry of the test tool to the test tool catalogue.

It is required that NFVAT provides an instantiation ability of the test tool according to corresponding test cases and calls of test tools to execute test cases.

It is recommended that NFVAT provides version control for test tools.

It is recommended that NFVAT supports pushing test tools from a repository to a remote NFV test bed.

It is recommended that NFVAT maintains the dependencies of the testing tool's libraries.

It is recommended that NFVAT verifies the network connectivity of test tools.

## 9.6 Requirements of NFVAT for test object management

It is required that NFVAT supports registry of test object to the test object catalogue.

It is recommended that NFVAT verifies the network connectivity of test objects.

It is recommended that NFVAT provides version control for test objects.

### 9.7 Requirements of NFVAT from system level

It is required that NFVAT could be customizable to accommodate different types of MANO platforms and include at least one open-source MANO platform.

NOTE 1 – Examples of open-source MANO platforms are OSM, ONAP, Tacker, XOS, etc.

It is recommended NFVAT includes pre-built dynamically configurable NFVI templates.

It is required that NFVAT could be deployed in containers or VMs based on open-source infrastructure platforms.

NOTE 2 – Examples of open-source infrastructure platforms are OpenStack and Kubernetes.

It is required that NFVAT provides extensible interfaces for external hardware traffic generators and network devices.

It is required that NFVAT supports being customizable for product vendor and end user needs.

# Appendix I

## Use cases of automated testing of NFV

(This appendix does not form an integral part of this Recommendation.)

### I.1 Use case template

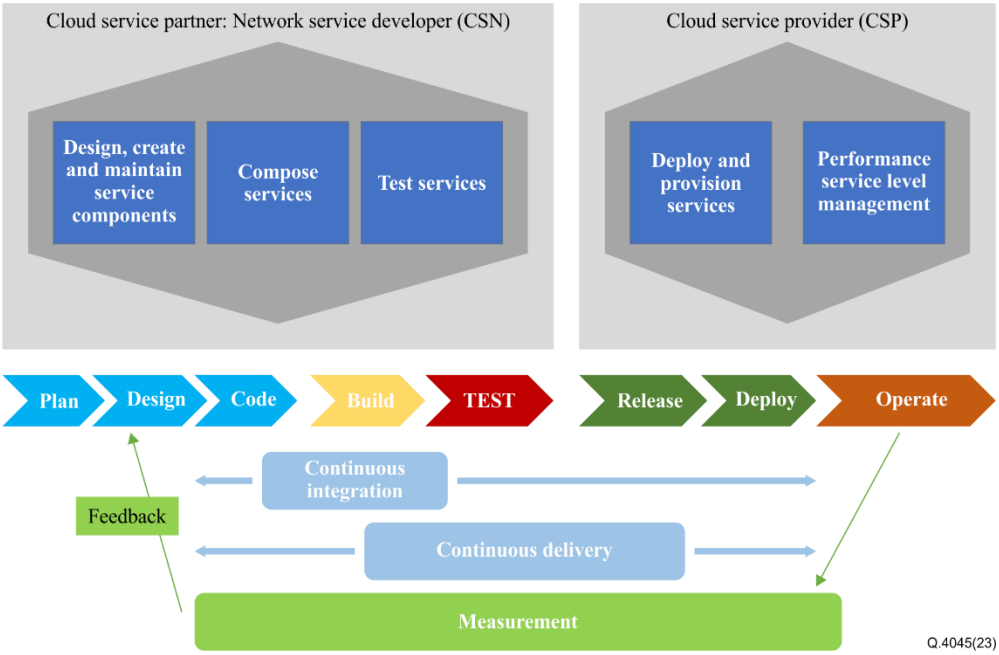
The use cases developed in this appendix should adopt the following unified format for consistent readability and convenient material organization.

**Table I.1 – Use case template**

| | |
|---|---|
| Title | Title of the use case |
| Description | Scenario description of the use case |
| Roles | Roles involved in the use case |
| Figure (optional) | Figure to explain the use case, but is not mandatory |
| Pre-conditions (optional) | The necessary pre-conditions that should be achieved before starting the use case. |
| Post-conditions (optional) | The post-condition that will be carried out after the termination of current use case. |
| Derived requirements | Requirements derived from the use cases, whose detailed descriptions are presented in the dedicated clauses. |

### I.2 Use case of NFV automated testing in cloud service continues integration/continues delivery

| | |
|---|---|
| Title | Use case of NFV automated testing in cloud service continues integration / continues delivery |
| Description | A VNF can be provided based on cloud computing, and the continuous integration and delivery of cloud services can be used to accelerate the online process of network functions. The continuous integration and delivery process of a cloud computing-based VNF can be summarized as follows:<br>– Cloud service partner (CSN): network service developers analyse the cloud service customer (CSC): network service customer's network requirements and define specific tasks.<br>– Based on this, the CSN: network service developers provide a development plan. The plan and corresponding tasks are created on the development platform.<br>– Then, a series of CI based development practices are used to ensure that VNF code or functional entities can be integrated, tested and deployed.<br>– Strict automated tests are conducted to verify whether the requirements of VNF are met. Only the code or functional entity that has passed the automated test can be deployed to the real environment through the deployment pipeline defined in the CD.<br>– After the code is delivered to the production environment, cloud service provider (CSP): network service provider automatically monitors its status to receive quality assurance feedback. According to the service level agreement (SLA) between CSP: network service provider and CSC: network service customer, CSP: network service provider is responsible for the continuity of network functions in the cloud.<br>– Finally, CSP: network service provider provides VNF services to CSC: network service customer.<br>– VNF automated testing is an important part of continuous integration and |

| | |
|---|---|
| | continuous delivery, which determines whether the service can meet user needs before deployment. |
| Roles | – CSN: network service developers<br>– CSC: network service provider<br>– CSP: network service customer |
| Figure (optional) |  <br>**Figure I.1 – NFV automated testing in cloud service continues integration/continues delivery** |
| Pre-conditions (optional) | |
| Post-conditions (optional) | |
| Derived requirements | – 9.1 Requirements of NFVAT for test template support<br>– 9.2 Requirements of NFVAT for test task instantiation<br>– 9.3 Requirements of NFVAT for test task orchestration and execution<br>– 9.4 Requirements of NFVAT for test report generation<br>– 9.5 Requirements of NFVAT for test tool management<br>– 9.6 Requirements of NFVAT for test object management<br>– 9.7 Requirements of NFVAT from system level |

## I.3 Use case of test case template

| | |
|---|---|
| Title | Use case of test case template |
| Description | In the process of NFV testing, the NFV tester often needs to create a virtual machine for test tools to call. As a basic operation, a test case template can be created for it, which is convenient for repeated use in the future. A test case template is designed by the NFV test designer.<br>This test case template should contain the following information:<br>– test case ID;<br>– test case type: this test case is a functional test case;<br>– precondition: the jump host where the test tool is located needs to communicate with the network where the virtual machine will be created. The network |

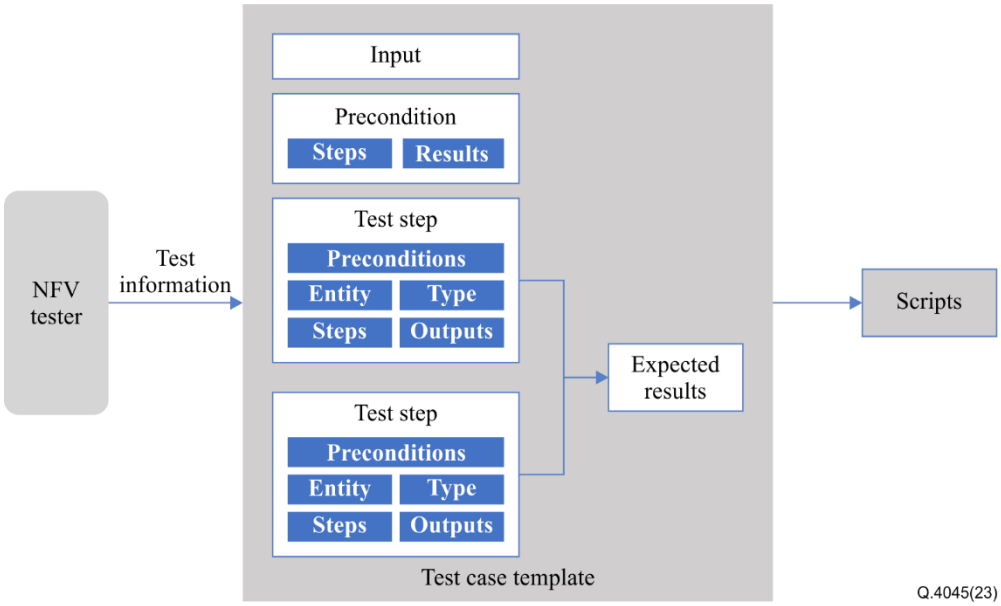
connection can be verified by the ping command;

– input information: the name of the virtual machine, the flavour of the virtual machine, the network of the virtual machine, etc.;
– test steps: call the API of the virtual infrastructure management platform, request to create a virtual machine;
– expected results: virtual machine1 is successfully created, the operating system of virtual machine1 is working well, and virtual machine1 is in flavour1.

The test case template can be structured as below.

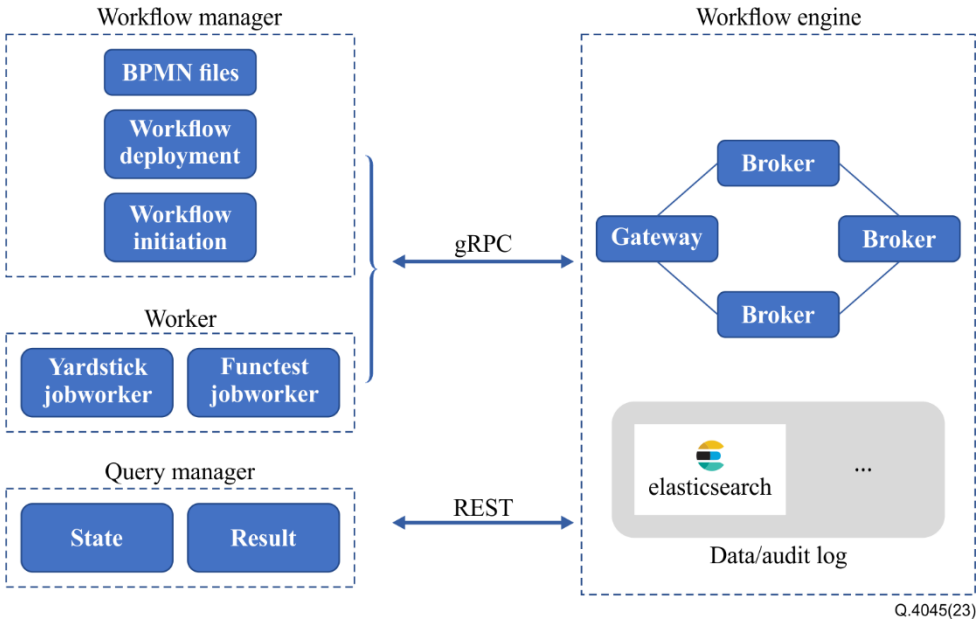
---

```
ID: 002
Test_case_type: functional
Object: OpenStack
Inputs:
 - name: flavor_1
   type: openstack_flavor
 - name: network_1
   type: openstack_network
pre_conditions:
 dependency: 001 #None
 condition: jump host connects to network_1
 pre_steps:
   - step_id: server_ping_server
     description: jump host connects to network_1
     step_detail:
       entity: $jump host$network_1
       type: CLI
       content: ping $network_1.gateway
       // the gateway of network_1
       output:
         log: $log1
         result: $status1
steps:
 - step_id: create_server
   description: call openstack create virtual machine API
   pre_condition: $status1
   step_detail:
     entity: $openstack
     type: API
     content: #API request body
     output:
       log: $log2
       result: $status2
```

| | |
|---|---|
| | entity: $virtual machine1<br>Expected results:<br>- virtual machine1 is successfully created<br>- the operating system of virtual machine1 is working well<br>- virtual machine1 is in flavour1<br>---<br>Based on the test case template, the test case can be easily transformed into test scripts by test script developer. |
| Roles | – NFV test designer<br>– Script developer<br>– NFV tester |
| Figure (optional) |  <br>**Figure I.2 – Usage of test template** |
| Pre-conditions (optional) | |
| Post-conditions (optional) | |
| Derived requirements | – 9.1 Requirements of NFVAT for test template support |

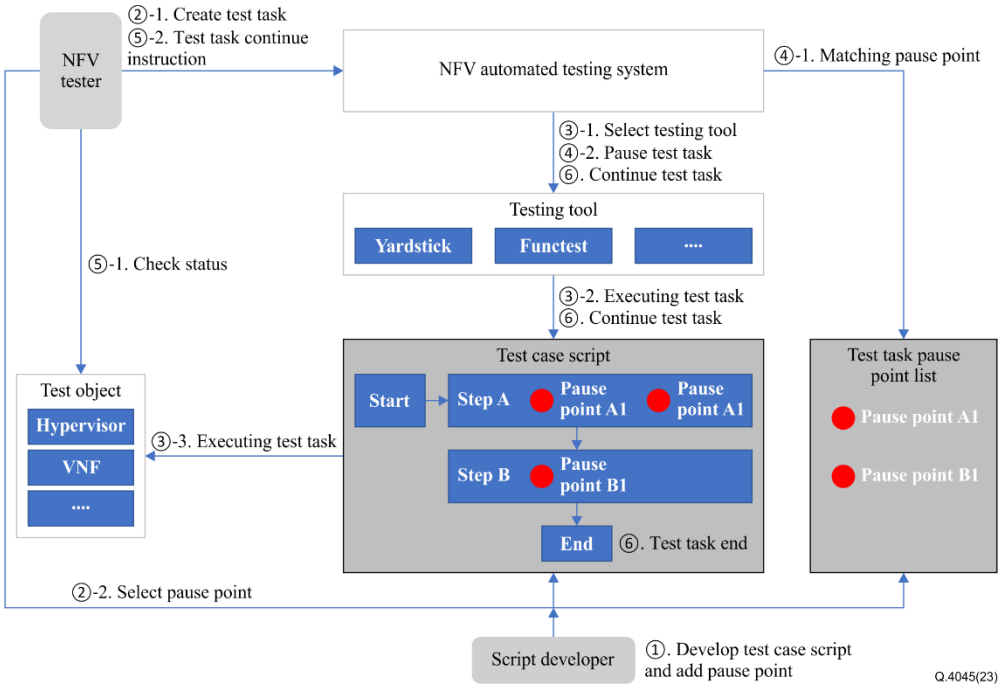
## I.4 Use case of NFV automated testing orchestration

| | |
|---|---|
| Title | Use case of NFV automated testing orchestration |
| Description | The purpose of NFV test orchestration is to create multiple automated test jobs and execute them one by one as planned. It is the most critical step in NFV automated testing. Figure I.3 shows an implementation example of test orchestration. It usually includes the following modules:<br>– Workflow manager: receives requests from the website, generates workflow description files, deploys specific workflows to the workflow engine, instantiates workflows, etc.<br>– Workflow engine: receives the workflow issued by the workflow manager, and generates the job to be processed according to the parameters input by the NFV tester. |

| | |
|---|---|
| | – Worker: provides the job workers of various types of jobs to process jobs. The workflow engine can call the job workers to execute jobs or put the jobs in the queue for the job workers to obtain.<br>– Query manager: NFV tester queries the execution status and results of workflow instances by query manager. |
| Roles | – NFV tester |
| Figure (optional) |  <br>**Figure I.3 – NFV automated testing orchestration process** |
| Pre-conditions (optional) | |
| Post-conditions (optional) | |
| Derived requirements | – 9.3 Requirements of NFVAT for test task orchestration and execution<br>– 9.5 Requirements of NFVAT for test tool management<br>– 9.6 Requirements of NFVAT for test object management |

## I.5 Use case of test reliability

| | |
|---|---|
| Title | Use case of test reliability |
| Description | Automated testing requires less manual participation. The state of the system usually returns to the initial state after the test is completed. If the automatic testing is executed in a black box, it is difficult to reproduce and check the intermediate state of the test after the test is completed. This reduces the credibility of the final test results. At the same time, if the test is terminated due to problems encountered during the test, these problems may be a problem of VNF service, a problem of network configuration, a problem of environment setting, a problem of resource quota, etc. It is difficult to locate faults.<br>If a test pause point is provided during the test process for testers to pause the execution of the test during NFV automatic testing and check the intermediate state of the test, such as checking the network connection of virtual machine and verifying the configuration of VNF, the reliability of test results can be improved. At the same time, it can provide a guarantee for the troubleshooting of environmental problems, the recurrence of faults and the confirmation of the state of test objects.<br>The interaction is as follows: |

| | |
|---|---|
| | – NFV test designer designs pause points for each test case.<br>– Script developers develop test case scripts, set check points at key positions of test cases, and write relevant codes.<br>– NFV testers create test task, select test cases in NFV automatic testing platform according to test requirements. For each test case, NFV tester select the pause points that meet the test requirements from the check points in the test case script. A list of test task pause points is formed.<br>– The NFV automatic testing platform selects the corresponding test tool, and gradually performs the test on the test object according to the step information of the test case.<br>– When the test case script executes to the check point, query whether the current check point is in the test task pause point list. If the check point is in the test task pause point list, the automatic test platform sends the test pause instruction to the test tool to pause the current test task.<br>– After the test task is suspended, the tester logs in the management module or GUI of the test object and the test environment to check the intermediate state, such as the network connection of the virtual machine and the port group setting of the virtualization platform, and analyses whether the current state is consistent with expectation.<br>– If the situation is inconsistent with the expectation, troubleshoot the problem. If the current state is consistent with the expectation, requests to continue the test.<br>– The automated test platform receives the request to continue the test and continues to execute the test case until the test task is completed. |
| Roles | – NFV test designer<br>– Script developer<br>– NFV tester |
| Figure (optional) |  <br>**Figure I.4 – Interaction between NFV testers and NFV automatic testing system for test reliability** |
| Pre-conditions (optional) | |
| Post-conditions (optional) | |

| | |
|---|---|
| Derived requirements | 9.1 Requirements of NFVAT for test template support<br>9.2 Requirements of NFVAT for test task instantiation<br>9.3 Requirements of NFVAT for test task orchestration and execution |

## I.6 Use case of NFV automated testing report

| | |
|---|---|
| Title | Use case of NFV automated testing report |
| Description | The test report of NFV automated testing comprises the documents which can present the process and results of the test, present the problems and defects found during the test, provide a basis for correcting the problems in the NFV system, and lay a foundation for NFV acceptance and delivery.<br>Test report should contain the following information:<br>– Information of the NFV tester.<br>– Test object information, including product name, version and product provider.<br>– Test environment information, including deployment location, surrounding systems and equipment, and equipment connection, etc.<br>– Test task information, including task name, test template used.<br>– Test case information, including the total amount of test cases, the detailed information of each test case, the start time and end time of each test case, the test conclusion of each test case (including pass, fail, partial pass), etc.<br>– The execution process of the test, including the resources used, the parameters configured, the intermediate execution status, the execution results obtained, the verification of the results, etc.<br>– Test score and test result.<br>An example of the test report is shown as below.<br>----------<br>Tester: Davide<br>Test object: hypervisor<br>Product type: computing virtualization<br>Product version: v0.05<br>Provider: Hua Wei<br>Environment name: Washington Data Center<br>Environment description: None<br>Test case amount: 3<br>Amount of test cases that do not pass: 0<br>Amount of test cases that pass: 3<br>Test score: 100<br>Test result: PASS<br>---------- |
| Roles | – NFV tester |
| Figure (optional) | |
| Pre-conditions (optional) | |
| Post-conditions (optional) | |
| Derived requirements | – 9.4 Requirements of NFVAT for test report generation |

# Bibliography


[b-ITU-T M.3710] Recommendation ITU-T M.3710 (2013), *Overview of an automated service test to support cost-efficient telecom service assurance*.

[b-ITU-T X.745] Recommendation ITU-T X.745 (1993), *Information technology – Open Systems Interconnection – Systems Management: Test management function*.

[b-ITU-T Y.4500.15] Recommendation ITU-T Y.4500.15/Q.3955 (2018), *oneM2M – Testing framework*.

[b-ETSI GS NFV 002] ETSI GS NFV 002 V1.2.1 (2014), *Network functions virtualisation (NFV); Architectural Framework*.

[b-ETSI GS NFV 003] ETSI GS NFV 003 V1.5.1 (2020), *Network functions virtualisation (NFV); Terminology for main concepts in NFV*.

# SERIES OF ITU-T RECOMMENDATIONS

| | |
|---|---|
| Series A | Organization of the work of ITU-T |
| Series D | Tariff and accounting principles and international telecommunication/ICT economic and policy issues |
| Series E | Overall network operation, telephone service, service operation and human factors |
| Series F | Non-telephone telecommunication services |
| Series G | Transmission systems and media, digital systems and networks |
| Series H | Audiovisual and multimedia systems |
| Series I | Integrated services digital network |
| Series J | Cable networks and transmission of television, sound programme and other multimedia signals |
| Series K | Protection against interference |
| Series L | Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant |
| Series M | Telecommunication management, including TMN and network maintenance |
| Series N | Maintenance: international sound programme and television transmission circuits |
| Series O | Specifications of measuring equipment |
| Series P | Telephone transmission quality, telephone installations, local line networks |
| **Series Q** | **Switching and signalling, and associated measurements and tests** |
| Series R | Telegraph transmission |
| Series S | Telegraph services terminal equipment |
| Series T | Terminals for telematic services |
| Series U | Telegraph switching |
| Series V | Data communication over the telephone network |
| Series X | Data networks, open system communications and security |
| Series Y | Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities |
| Series Z | Languages and general software aspects for telecommunication systems |

Published in Switzerland
Geneva, 2024