ETSI GR NFV-TST 011 V1.1.1 (2019-03)

Network Functions Virtualisation (NFV);
Testing;
Test Domain and Description Language Recommendations

Disclaimer

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

Reference
DGR/NFV-TST011

Keywords
language, NFV, testing

ETSI

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

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

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

Important notice

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

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

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

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

Contents

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

Foreword

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

Modal verbs terminology

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

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

Executive summary

The present document proposes a model of the NFV test domain and recommends requirements for a test Domain Specific Language (DSL) to manipulate it. In the context of NFV, a network service is supplied by multiple vendors and each vendor has its own test technology, interfaces into the system under test, and test languages (usually GPLs like Java®, Ruby®, Python®, etc.) In order to create a common test language, the test cases follow a standardized test case model that the language can manipulate, and that can be implemented within individual test technologies. The model includes shareable and reusable artefacts tied to the test domain: execution flow, data, abstract resources, environment, etc.

Integration of multiple test technologies is only possible by a system that can accept contributions of test resources from multiple parties. These contributions may include lab resources, test APIs, test data, high-level function libraries, test execution platforms, etc. The test environment is then constructed dynamically from these contributions. To allow the dynamic nature of the test environment, it is necessary for the test case to be decoupled from specific resource contributions and express the test process in terms of resource abstractions. Mapping these abstractions to concrete resources is the job of a Dynamic Resource Management (DRM) system. This is done by creating an environment resource meta-model available to the test case developers at design time. The meta-model is then used for creation of specific environment instance models at runtime. Each environment instance includes dynamic resource contributions to which resource abstractions are mapped.

Introduction

With the advent of NFV, the industry is experiencing the following transformative challenges:

- Multiple contributions to a network service
- Open collaboration
- Shift away from dedicated resources (sharing of resources)
- Shift of integration responsibility from vendors to service providers (or their agents)
- Test cases/plans can be repeated multiple times, making reuse critical

To address these challenges and to encourage collaboration, the present document provides recommendations for NFV test domain modelling and a Test DSL that does not force vendors/participants to change their test technology/language and enables efficient utilization of resources. To enable reuse, the model also decouples test data and test environment from the test case and uses dynamic allocation of test resources.

1 Scope

The present document proposes a model of the NFV test domain and recommends requirements for a test Domain Specific Language (DSL) to manipulate it. The description includes an NFV test automation ecosystem that facilitates interaction among NFV suppliers and operators, based on the DevOps principles.

The NFV test domain contains:

- System Under Test (SUT): Network Functions (NF), Network Functions Virtualisation Infrastructure (NFVI) and network services.
- Test Resources: tools or instrumented NF's and NFVI elements that test cases can interface to manipulate the SUT.
- Test Execution Flow: controlled and uncontrolled state transitions.
- Test case configuration data and parameters: test-resource-specific and non-test-resource-specific.

The present document explores the following attributes to enable efficient multi-supplier NFV interaction:

- Reusability of test plans, test cases and test resources.
- Abstraction of test data.
- Decoupling of test case from the test environment.
- Use of test resource abstractions in place of concrete resources.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

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

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

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

[i.1] ETSI GS NFV-TST 001 (V1.1.1): "Network Functions Virtualisation (NFV); Pre-deployment Testing; Report on Validation of NFV Environments and Services".

[i.2] ETSI GS NFV-MAN 001 (V1.1.1): "Network Functions Virtualisation (NFV); Management and Orchestration".

[i.3] ETSI GS NFV-SOL 003 (V2.3.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point".

[i.4] ETSI GS NFV-IFA 006: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification".

[i.5] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS NFV 003 [i.5] and the following apply:

execution engine: means by which a program can be executed on the target platform using the two approaches of interpretation and compilation

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS NFV 003 [i.5] and the following apply:

DSL	Domain Specific Language
GPL	General Purpose Language
HLF	High-Level Function
TEP	Test Execution Platform

4 Test Domain

4.1 Overview

The NFV test domain is a set of artefacts and systems for testing NFV-based solutions. NFV introduced the concept of "dynamically configurable and fully automated cloud environments" (https://www.etsi.org/technologies/nfv) for network functions. The present document models the NFV test domain as a set of abstractions so that the same level of flexibility is available in testing those network functions. Figure 1 is included to illustrate relationships among artefacts discussed in clause 4; the NFV Test Domain model is described in more detail in clause 6. In addition, the present document proposes requirements for a Domain-specific language (DSL) to manipulate that test domain. Using these models and recommended requirements, suppliers and service providers are able to leverage different test technologies, dynamically allocate test resources, and reuse test plans, test cases, and Test Execution Platforms (TEPs). In the present document, a test case always refers to a computer program that can be executed by a test automation system.

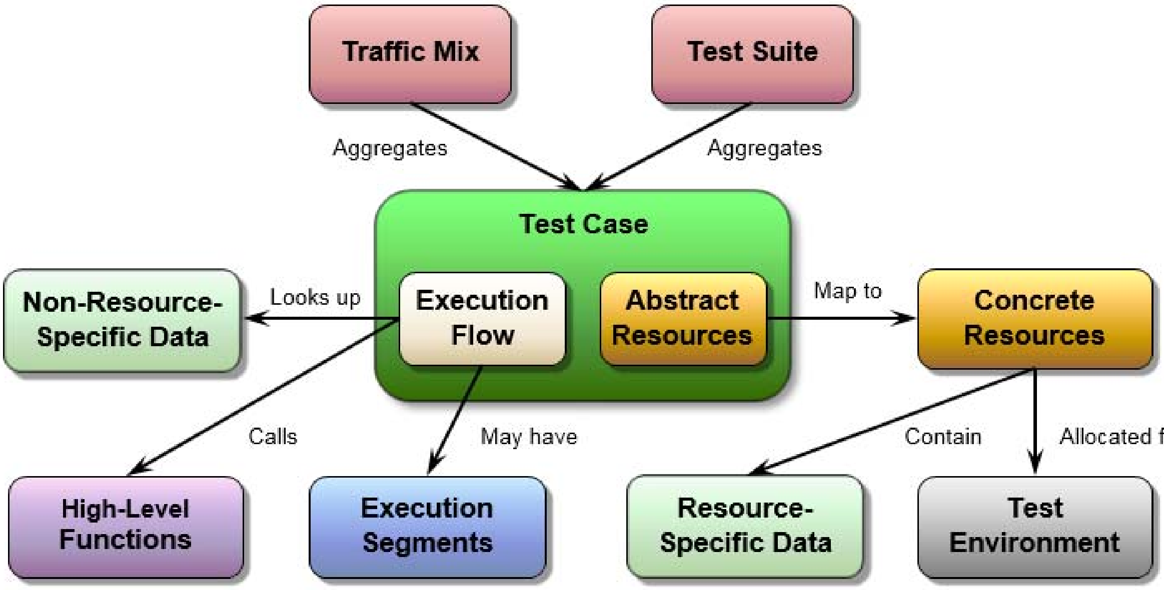

Figure 1: NFV Test Domain Artefact Relationships

The NFV test domain is comprised of:

- System Under Test (SUT) [i.1]: Network Functions (NF), Network Functions Virtualisation Infrastructure (NFVI), and network services
- Test Case Resources: tools or instrumented NF's and NFVI elements that test cases can interface to manipulate the SUT
- Execution Flow: controlled and uncontrolled state transitions
- High-Level Functions
- Test case configuration data and parameters: test-resource-specific and non-test-resource-specific
- Execution Segments
- Test Environment
- Test Suites & Traffic Mixes

4.2 Test Case Resources

In order to be tested, the SUT exposes a set of interfaces over which test interactions happen. These interfaces vary in the degree of complexity and may include entire protocol stacks. The test drivers communicate with the SUT by sending and receiving encoded messages over one or more of these interfaces. This means that the implementation of the interface is also present on the test side.

It is therefore necessary for the executing test case to create or otherwise acquire one or more objects that implement the SUT interfaces and use them to send and receive messages to and from the SUT. These objects in turn expose their own interfaces to the test case to allow the test to manipulate the message flow more easily. These objects will be referred to as abstract resources. In essence an abstract resource is an instance of an SUT interface that provides its own API to the test.

Abstract resources are test case-facing abstractions that utilize units of lab equipment, software, and/or data to do real work. These units will be referred to as concrete resources. In the context of a shared lab, concrete resources are shared among multiple users and are allocated to specific test cases for the duration of the test. This guarantees that the test case gets exclusive access to concrete resources required for its execution. Some examples of concrete resources include instances of instrumented or stubbed out MANO components, VNFs, VNFCs, etc. Availability of concrete resources is generally limited and concurrently executing test cases contend to gain access to them.

The relationship between abstract and concrete resources is typically many-to-many, and the mapping between them is described in clause 6.3. The test DSL is expected to enforce proper resource declaration, preclude any access to concrete resources outside the resource management system, and provide the user with an intuitive way to declare and manipulate abstract resources. Once an abstract resource is mapped to an allocated concrete resource they form a single entity used by the test case to interact with the SUT. This entity will be called test case resource as illustrated in Figure 2.

4.3 Test Execution Flow

As the test case executes, every resource goes through a sequence of states, reflecting the SUT functionality being tested. These sequences range in complexity from trivial to very complex. The test case may be interested in some but not all of these states. For example, if a resource is running a protocol stack, unless protocol conformance is being tested, most low-level messaging is of no interest to the test case. It may only be interested in the successful or unsuccessful outcome of such messaging. The degree to which the test case has visibility of the resource state can vary for the same type of resource depending on the test scenario. Consequently, it is necessary for a mechanism for different levels of control over the resource state to be present. The test case should have the ability to "take over" the resource state transition when necessary and let the resource run its own state machine at other times. The resource states controlled by the test case are referred to as controlled states.

Controlled states of all test case resources at any given time form the state of the test case. Test case initiates state transitions by sending or receiving and verifying messages to and from the SUT on any of the test case resources. For example, if the SUT is a VNFM implementation and an Instantiate VNF request is sent from the NFVO resource, the test case can verify receiving a Grant VNF Lifecycle Operation request by the NFVO resource and after acknowledging it with a Grant VNF Lifecycle Operation response, verify that the Allocate Resource request is received by the VIM resource. Running a single state machine per test case for controlled states while letting individual resources run their own state machines for uncontrolled states provides an intuitive and flexible framework for SUT interface interworking.

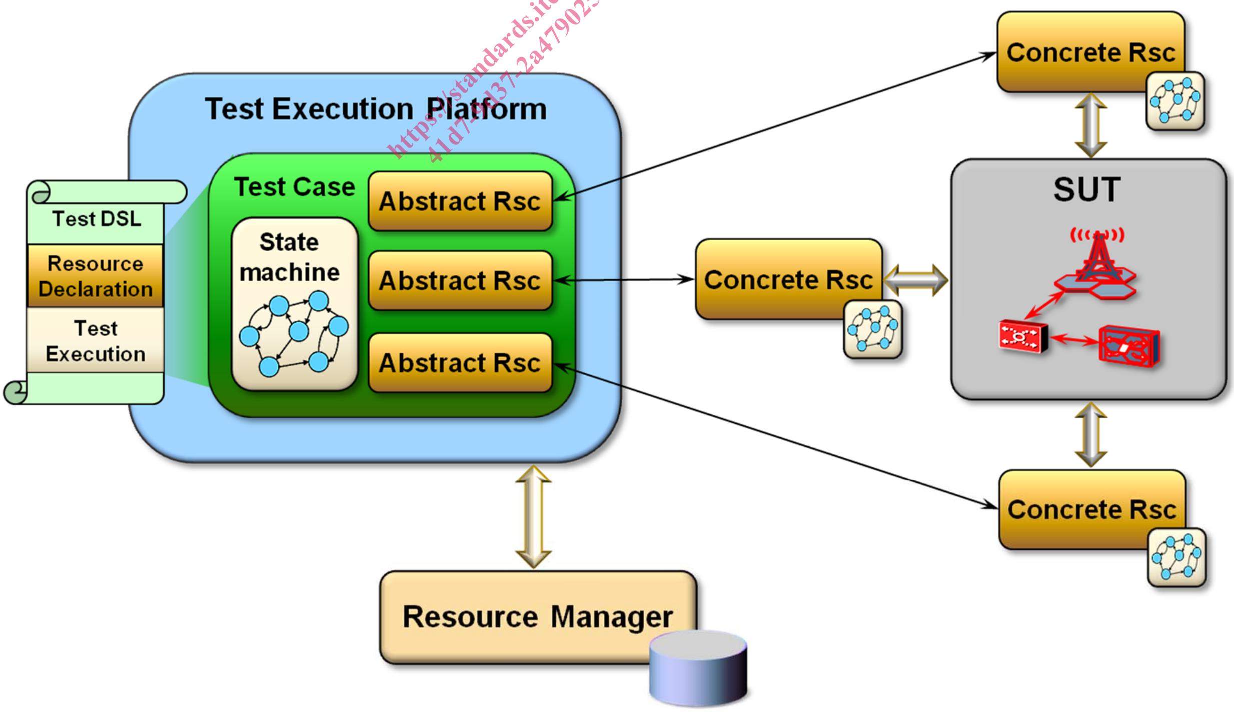

Figure 2: Test Case Resources and State Machine