H2020 5GASP Project

Grant No. 101016448

D5.3 Development of testing tools and framework for the Automotive and PPDR Verticals

Abstract

The development of the flexible and automatic Test, Validation, and Certification Services (TVCS) in 5GASP is one of the main goals of this project. These services will help SMEs and Telecommunication Operators significantly shorten the cycles between the development, testing, and production of their NetApps. This document details all the efforts to develop 5GASP TVCS, the goals it aims to achieve and presents the type of flexible testing, the automation architecture, and the implementation of the automation testing. Furthermore, the document also presents the test scope and lists the test cases in the current initial development phase of 5GASP TVCS. Finally, the automation workflows are presented as a guideline to show the detailed steps from the onboarding of the NetApp triplet to the validation process results collection.

Document properties

Document number	D5.3
Document title	Development of testing tools and framework for the Automotive and PPDR Verticals
Document responsible	Ben Shaw (EANTC)
Document editor	Ben Shaw (EANTC)
Editorial team	EANTC, OdinS, ITAv, VMWare, UoP, UNIVBRIS
Target dissemination level	PU
Status of the document	Submission version
Version	V1.0

Document history

Revision	Date	Issued by	Description
0.1	11/02/2022	EANTC	Initial draft
0.2	15/03/2022	EANTC	Updated draft
0.3	21/03/2022	EANTC	Merge all the contributions into one draft
0.4	25/03/2022	EANTC	Updated based on the comments from the reviewers of each section
1.0	01/04/2022	EANTC	Final version – updated based on the comments from the reviewers of the full document

List of Authors

Company	Contributor	Contribution
EANTC	Ben Shaw	Abstract, Objectives of this document, Document structure, Testing by the tools, Automation Testing Architecture (Tools), CI/CD Manager, 5GASP Test Scope, Certification initial criteria, Automation Testing Workflows
VMware	Vesselin Arnaudov	Deliverable Reviewers, Automation Testing Architecture reviewer
BLB	Roman Odarchenko, Mihail Myagkiy	Automation Testing Architecture (Tools) reviewer, Vertical Service testing, Automotive service test cases, Cross-vertical service test cases, Vertical Service KPIs and the expected values
DriveU	Eli Shapira	Cross-vertical service test cases, Vertical Service KPIs and the expected values
UNIVBRIS	Navdeep Uniyal, Xenofon Vasilakos	Cross-vertical service test cases, document review, testing by the tools, test scope
Lamda Networks	Leonidas Lymberopoulos	Cross-vertical service test cases

Orange Romania	Catalin Brezeanu Andreea Bonea Marius Iordache	Local test repository, Deliverable reviewers
ININ	Luka Koršič	Value added testing - Certification expanded criteria, PPDR Service test cases
ITAV	Diogo Gomes Rafael Direito Daniel Gomes Daniel Corujo Rui L. Aguiar Pedro Bastos	Reviewers of the sections: “Objectives of this Document”, “Approach and Methodology”, “Document Structure”, “Value Added Testing - Certification expanded Criteria”. Main contributors of the following sections: “Testing by the Scripts”, “Automation Testing Architecture” (section’s introduction), “Automation Testing Architecture (Scripts)”, “CI/CD Manager”, “CI/CD Agent”, “Test Results Visualization Dashboard”, and “Key Performance Indicators” (section’s introduction)
OdinS	Jorge Gallego Madrid Ana Hermosilla García Antonio Skarmeta	Approach and methodology, Automation Testing Architecture, Automation Testing Implementation, Testing Tools, Vertical Service Testing, Automotive Service Test Cases and E2E Network KPIs and the expected values

Disclaimer

This document has been produced in the context of the 5GASP Project. The research leading to these results has received funding from the European Community's H2020 Program under grant agreement number 101016448.
All information in this document is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose. The reader thereof uses the information at its sole risk and liability.
For the avoidance of all doubts, the European Commission has no liability in respect of this document, which is merely representing the author’s view.

Contents

List of Figures

List of Tables

List of Acronyms

3GPP	3rd Generation Partnership Project
5G	5th Generation
5GASP	5G Application & Services experimentation and certification Platform
5GASP-C	5GASP Certification
5G-PPP	5G Infrastructure Public Private Partnership

5G SA	5G Stand Along
API	Application Programming Interface
CD	Continuous Deployment
CI	Continuous Integration
CN	Core Network
CNF	Cloud-Native Network Function
CSS	Cascading Style Sheets
EMHO	Efficient MEC Handover
ETSI	European Telecommunications Standards Institute
FTP	File Transfer Protocol
FTPS	File Transfer Protocol Secure
GUI	Graphical User Interface
gNB	Next Generation Node B
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HTML	HyperText Markup Language
IFA	International Franchise Association
IOPS	Isolated Operation for Public Safety
K8s	Kubernetes
KPI	Key Performance Indicator
LTR	Local Test Repository
MANO	Management and Orchestration
ML	Machine Learning
NEST	Network Slice Template
NFV	Network Function Virtualization
NFVO	Network Function Virtualization Orchestrator
NODS	NetApp Onboarding and Deployment Services
MOS	Mean Opinion Score
NS	Network Service
NSD	Network Service Descriptor
NSaaS	Network Slice as a Service
NWDAF	Network Data Analytics Function
ONAP	Open Networking Automation Platform
OSM	Open-Source MANO
PPDR	Public Protection and Disaster Relief
QCOW	QEMU (Quick Emulator) Copy On Write
RAN	Radio Access Network
REST	Representational State Transfer
RUS	Road-side-Unit
SME	Small and Medium-sized Enterprise
TAF	Test Automation Framework
TEE	Test Execute Engine
TD	Testing Descriptor
TMF	Tele Management Forum
TVCS	Test, Validation and Certification Services
TRVD	Test Results Visualization Dashboard

UE	User Equipment
UI	User Interface
URL	Uniform Resource Locator
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
vOBU	Virtual On-Board Unit
WP	Work Package
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

1. Introduction

1.1. Objectives of this document

5GASP aims to automate the process of testing and validation, lowering costs associated with testing and certification of NetApps in a telecommunications-oriented environment. 5GASP CI/CD Service is defined in D5.1 [1], and introduces aspects such as the methodologies of the testing, validation, and certification process. The integration guide and open APIs are defined in D5.2 [2]. D5.2 also addresses the methodologies of the cooperation between the different Services in 5GASP to enable the automatic onboarding, validation, and certification of NetApps, in 5GASP facilities.

The primary goal of this document is to describe all the efforts of developing the automation and flexible testing in 5GASP, not only the high-level automation architecture, but also the lower-level testing plan, open-source testing tools, and the test cases, along with their well-defined KPIs and the expected values.

Finally, the workflows are presented in this document, addressing the automation execution of the testing, validation, and certification in 5GASP. These workflows are open, which means they could easily be reused in any other 5G ecosystem or adapted through the open APIs and tools used in the workflows.

1.2. Approach and methodology

This document provides an overview of the testing framework for the automotive and PPDR verticals within the WP5 scope. In this way, it uses the architecture defined in WP2 as foundations to make all the testing procedures compliant. It also takes into account the integration of the different infrastructures provided by the 5GASP testbeds to make testing possible across different domains and scenarios. Besides, it also works with WP4 to ease the design and development of NetApps by introducing the definition of the 5GASP Test Plan. With this planning, developers and experimenters will be able to design the testing procedures that will validate the deployment and operation of the NetApps in the 5GASP ecosystem.

1.3. Document structure

This document is composed of 5 sections, as follows:

Section 1 presents the main goals, structure, and the underlying approach and methodologies for the 5GASP Test, Validation and Certification Services (TVCS).

Section 2 introduces the efforts implementing the 5GASP TVCS, presenting the test types, automation architectures, and the TVCS components. The TVCS components automatically manage and execute the test cases based on the Testing Descriptor (TD), keep the test logs, test results, and test reports in the local database, and finally share them among the other Services in 5GASP, i.e., NetApp Onboarding and Development Services (NODS) in WP3 and 5GASP Certification (5GASP-C) Service in WP6.

Section 3 defines the test plan with (i) detailed test scope based on the definitions in WP2, (ii) test case lists which could be used as certification criteria, and the validation of the verticals, (iii) Key Performance Indicators (KPI) with the expected value, which cover not only the network but also the service layer. The current tests are designed for both Automotive and PPDR verticals, in which the currently approved 5GASP NetApps in WP4 provide the business service.

Section 4 focuses on the testing workflows that present an overview of the test process with the activities step by step, so that the developer could fully understand how a NetApp would be tested in 5GASP TVCS. It could also be used as a user's guideline to the tester.

Section 5 presents the conclusion, which summarizes the content introduced in this deliverable.

2. 5GASP Test, Validation and Certification Services

5GASP provides an automatic Test, Validation, and Certification Service (TVCS) for a NetApp which is following the requests of D4.1 [3] and D4.2 [4] in WP4. 5GASP TVCS is based on 5GASP CI/CD Service defined in D5.1 [1] and cooperates with NODS defined in D3.1. 5GASP TVCS aims to compose such 5GASP Services as an automation platform for flexible testing. 5GASP TVCS is an open platform employing open-source tools as much as possible via the open APIs defined in D5.2 [2].

2.1. Flexible Testing

5GASP NetApp testing refers to the process of testing any developed 5G network application using scripts, tools, or any test automation frameworks [5] to identify errors and applications behavior in general. It aims to help NetApp developers release bug-free and robust NetApps into the real world. It would also enable NetApp developers to identify bugs in the early stages of development and save development time.

5GASP TVCS provides a flexible test methodology, which offers two types of testing service, **Developer-defined Tests,** and **5GASP-Predefined Tests.** Table 1 compares these two types of testing in the different fields.

Table 1 - Developer-defined and 5GASP-Predefined Tests

Testing Type	Developer-defined Tests	5GASP-predefined Tests
Source Format	Test scripts Test tools	Test scripts Test tools
Source from Whom	NetApp Developer	5GASP NetApp Tester
Test Method	White box	Black box
Test Executor	Robot framework	Open-source tools
Test layers	Layer 1-7	Layer 1-4
Test Domains	Validation on the dedicated domains of one NetApp in his vertical, e.g., feature, service, business, etc.	Certification on the common domains of all NetApps in the 5G ecosystem, e.g., interoperability, performance, security, etc.
Test Propose	Achieve the max industry requests as much as possible	Meet the minimum industry requests
Test Goal	5GASP Showcase	5GASP Certificate
Test Plan	Test cases in sections 3.2.3.1, 3.2.3.2, 3.2.3.3	Certification criteria in sections 3.2.1, 3.2.2

There is no difference in the source format between Developer-defined Tests and 5GASP-Predefined Tests, as derived from the presentation in Table 1**.** Both could be either test scripts which are the sets of test commands be executed by Robot test automation framework, or test tools. These two different types are from the view of the 5GASP external potential stakeholder (i.e., application developers and service providers) in the marketplace. From the view of the 5GASP internal stakeholder (i.e., NetApp developers and facility owners), the main difference of the flexible testing is the source format. The following subsections 2.1.1 and 2.1.2 present more detailed information about these two formats.

5GASP CI/CD Service provides one centralized CI/CD Manager, and one distributed CI/CD Agent in each Testbed. 5GASP TVCS reused the components of 5GASP CI/CD Service, but they are working as different logic elements from the view of the testing.

Table 2 presents the details of the logic elements in 5GASP TVCS.

Table 2 - 5GASP TVCS logic elements

		CI/CD Service Component	
Test Format		CI/CD Manager	CI/CD Agent
	Test Scripts	Pipeline Manager	Jenkins platform with Robot framework
	Test tools	Automation Server	Testing Tools Server

Section 2.3.1.1 presents more details of the component of CI/CD Manager, and CI/CD Agent in section 2.3.1.2.

2.1.1 Testing by the scripts

A t the date of publication of this deliverable, 5GASP's CI/CD Service relies entirely on Robot Framework [6] defined tests. We anticipate, in the future, support for other Test Automation Frameworks (TAFs) will be provided.
Robot Framework is a free and open-source tool with a vast community that continuously contributes to enhance its capabilities. These can easily be extended through Java and Python implemented libraries. One other great advantage is the easier installation of this Testing Automation Framework and the fact that it is very lightweight.

Regarding the testing process output, Robot Framework produces XML and HTML interactive report files, which can be provided to the NetApp developers, to detail the results of the validation processes.

Using Robot Framework, a myriad of tests can be implemented. For instance, infrastructure tests that validate the bandwidth, packet loss, latencies, and many other aspects.
In 5GASP, the development of these tests is achieved using Python. The tests are composed of two files: (i) a robot file, which describes all the test cases for the test being performed, and (ii) a library file, addressing the test being executed, which contains the test implementation, imported in the Robot file. Both files are stored in the LTR and are gathered and executed by the CI/CD Agent.

Figure 1 presents the Robot file for a packet loss test as an example, and Figure 2 for its library file. Given these figures, it is easy to understand that the library file implements a function to obtain the percentage of lost packets. This value will then be consumed in the

Robot file and compared to a defined threshold, thus, getting the final test result: success or failure.

```
*** Settings ***
Library         PacketLoss.py

*** Test Cases ***
Testing the packet loss percentage
    ${COMPARATOR}=       Run Keyword If      '%{packet_loss_comparator}' == 'more_than'        Set Variable    >
    ...     ELSE IF     '%{packet_loss_comparator}' == 'more_or_equal_than'        Set Variable    >=
    ...     ELSE IF     '%{packet_loss_comparator}' == 'less_than'        Set Variable    <
    ...     ELSE IF     '%{packet_loss_comparator}' == 'less_or_equal_than'        Set Variable    <=
    ...     ELSE     Fail  \nComparator has not been defined


    ${loss_percentage}=    Packet Loss

    IF     '${loss_percentage}' != '-1'
    Should Be True    ${loss_percentage} ${COMPARATOR} %{packet_loss_threshold}
    ELSE
    FAIL    \nImpossible to compute packet loss percentage
    END
```

Figure 1 - Robot file for a packet loss test

```
'''
5GASP - NetApp Surrogates - Packet Loss KPI Test
@author: Rafael Direito <rdireito@av.it.pt>
'''

import paramiko, re
import os

host1 = os.getenv('packet_loss_host1_ip')
username1 = os.getenv('packet_loss_host1_username')
password1 = os.getenv('packet_loss_host1_password')
host2 = os.getenv('packet_loss_host2_ip')


def packet_loss():
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        client.connect(hostname=host1, username=username1, password=password1)
    except:
        print("[!] Cannot connect to the SSH Server")
        exit()

    stdin, stdout, stderr = client.exec_command(f"ping -c 20 {host2}")

    pingResult = stdout.read().decode()
    regex = re.compile(r',( [0-9,.]+)% packet loss,')
    result = regex.search(pingResult)

    if result:
        loss_percentage = float(result.group(1))
        return loss_percentage
    else:
        return -1


if __name__ == '__main__':

    packet_loss()
```

Figure 2 - Python library file for a packet loss test

In the presented test, several parameters are needed. This is common among all the tests of the 5GASP ecosystem. These parameters must be defined in the onboarded Testing Descriptors. Section 4 provides further information on the onboarding of the Testing Descriptors, also detailing the process of the onboarding of the developer-defined tests.

2.1.2 Testing by the tools

5GASP aims to set up a reproducible TVCS environment which offers open-source tools as much as possible. This section focuses on how to use the tools for testing in 5GASP TVCS. Section 2.3.1.5 provides the list of the tools and the details of each tool.

5GASP CI/CD Agent is a pre-installed Virtual Machine (VM) in each testbed, which will be reused to launch the predefined testing tools by 5GASP project. The dedicated testing tools from the developers could be launched on the testing VNF, which could be developed and eventually removed at the same time with the NetApp under test. 5GASP has the CI/CD Agent set up as one rich resource VM (6 vCPU, 16 GB RAM, 50 GB disk space, and 3 network interfaces), so that multiple testing tools can run at that same. 5GASP TVCS supports parallel testing and

Table 3 presents the parallel testing scenarios.

Table 3 - 5GASP TVCS parallel testing scenarios

Scenario ID	Number of NetApp under test	Number of test tools	Test domains
Scenario 1	1	Multiple	Tool 1: functional testing Tool 2: performance testing ...
Scenario 2	Multiple	1	Only functional testing: Tool 1 - NetApp 1, 2, ...
Scenario 3	Multiple	Multiple	Only functional testing: NetApp 1 - Tool 1a, 1b, ... NetApp 2 - Tool 2a, 2b, ...
Scenario 4	1	1	Only performance testing

The scenarios could be managed by the parameters in the onboarded Testing Descriptors which are also used by the test scripts as described in section 2.1.1. The current example of a Testing Descriptor is only for script testing. One new parameter *toolList* should be added for the selection of the tools. Figure 3 presents an example of the updated Testing Descriptor with a new parameter, i.e., toolList.

```yaml
# Test general information and relations with netapp, NS and testbed
test_info:
  test_id: 1
  netapp_id: vOBU # Set by Openslice during the onboarding
  network_service_id: vOBU_ns # Set by Openslice during the onboarding
  testbed_id: gaia5G # Set by Openslice during the onboarding
  description: Infrastructure tests for vOBU NetApp
  startTime: startTime # Set by Openslice during the onboarding
  endTime: endTime # Set by Openslice during the onboarding
  toolList: iPerf, TRex # Set by Openslice during the onboarding

# Test definition in three phases
test_phases:
  # Test setup: deploy vnfs if required and define testcases
```

Figure 3 - Testing Descriptor example with toolList

The parameter *netapp_id* in a Testing Descriptor identifies the NetApp under test. The parameters *netapp_id, startTime*, *endTime*, and *tool List* could be used for scenario selection of each NetApp (i.e., scenarios 1 and 4 in Table 3). Finally, from the viewing of 5GASP TVCS, when multiple NetApps are under test in the same time slots, scenarios 2 and 3 in Table 3 would be covered. The next step is to define an automation controlling mechanism for the scenario controlling in 5GASP TVCS.

For each test case in the Testing Descriptors, it would either be script testing or tool testing, but never both. One new parameter *method* should be added with only two values - either script or tool. One test case should only be executed by one script file or tool, never by multiple ones. The existing parameter *file* would be renamed to *Robot_file* which should be available only for script testing; one new parameter *Test_tool* should be added and available only for tool testing. Figure 4 presents a unified example of the updated Testing Descriptor which could be used for both script testing and tool testing.

```yaml
test_phases:
  # Test setup: deploy vnfs if required and define testcases
  setup:
    deployments:
    # Define the testcases
    testcases:
      # Developer defined tests
      - testcase_id: 1
      # 5GASP predefined tests
      - testcase_id: 2
        type: 5GASP predefined
        scope: infrastructure
        name: bandwidth
        method: script # one test case could be either script testing or tool testing
        Robot_file: "testBandwidth" # only available only for script testing
        # CI/CD service will retrieve the test with that name for the respository
        Test_tool: iPerf # only available only for tool testing
        # CI/CD Manager will call for the tool with that name in the CI/CD Agent
        parameters:
          - key: server_ip
            value: 10.0.0.1 # The remote access IP address of the host where the server is running
          - key: host1_username
            value: root
          - key: host1_password
            value: password
          - key: Client_ip
            value: 10.0.0.2 # The remote access IP address of the host where the client is running
          - key: host2_username
            value: root
          - key: host2_password
            value: password
          - key: desiredValue
            value: 100 mbps
          - key: comparator
            value: more than
```

Figure 4 - Testing Descriptor unified example

In the initial setup phase of 5GASP TVCS, each testbed would like to install the test tools according to the recommended tool list in NODS.

Table 4 presents the template of the recommended tool list with some examples. The predefined open-source tools with the same version should be installed on the CI/CD Agent in each testbed before the testing, e.g., iPerf in Table 4. The test cases which could be shared with the other NetApps are also exposed to the same table below.

Table 4 - Rrecommended tool list template

Tool Name	Open Source (Yes/No)	Software Version	Redefined/ Dedicated	Ready in Testbeds	Shared Test Case
iPerf	*Yes*	*V3.9 Server and client on Ubuntu*	Predefined	*Aveiro Patras Bristol Ljubljana Bucharest Murcia*	*testBandwidth, ...*
TRex	*Yes*	*V0.25.9 on Ubuntu*	Predefined	*Not yet*	*testHTTPS*
qMON	*No*	*V1.0.57*	*Dedicated*	*Ljubljana*	*e2eWebMOS*

		Server on Ubuntu *Agent on* ***Android***			

In the final phase, 5GASP aims to automate the installation of the predefined open-source tools (NOT for the non-open-source tools) by CI/CD Manager on the CI/CD Agents.

2.2. Automation Testing Architecture

This section introduces the TRVC architecture. Since the automation of the testing pipelines is complex, this section is divided into multiple subsections that provide focus exclusively on specific parts of the testing pipelines. Section 2.2.1 introduces the base architecture and workflow of the CI/CD Service, addressing its components and their role in the execution of scripted tests. Then, in Section 2.2.2, the tools used to perform several of these tests are presented. This section also introduces the architecture of the test execution via testing VNFs - VNFs specifically created to perform tests on NetApps. Lastly, Section 2.2.3, addresses the overall architecture of 5GASP's validation service, consolidating the concepts introduced in Sections 2.2.1 and 2.2.2.

2.2.1 Automation Testing Architecture (Scripts)

5GASP TVCS supports the execution of two different tests: (i) 5GASP-Predefined tests and (ii) Developer-defined tests. Both are possible to be configured using Robot Framework and then are available in the LTRs via several testing scripts.

The automated execution of these scripts is mainly enabled by 4 different components: (i) the NODS, (ii) the CI/CD Manager, (iii) the CI/CD Agents, and (iv) the LTRs. Besides these, the TRVD also takes a key role in this phase since it presents the results of the tests to the end developers.

The execution of a new validation process is triggered by the NODS. During this phase, the NODS posts a TMF [7] payload to the CI/CD Manager's API, providing a list of tests to be executed and a myriad of variables required to define the test cases. After 5GASP NODS triggers a validation process on the CI/CD Manager, the Manager will create a testing pipeline configuration file, based on the Testing Descriptor onboarded on NODS, and submit it to the CI/CD Agents responsible for testing the NetApp. Then, the CI/CD Agent will gather all the required test files from the LTR deployed on the testbed. These tests will have to be further augmented using the testing variables initially provided by the NODS. Once this process is finished, the Agent will perform the tests, collect their results, and send them back to the CI/CD Manager, which will parse the results, store them, and make them available via the 5GASP TRVD.

It is also important to mention that, if the NetApp developer onboard developer-defined testing scripts, these will have to be transferred to the LTRs before the tests' execution. And it is recommended to re-use the existing test scripts in the LTRs if possible.

2.2.2 Automation Testing Architecture (Tools)

The testing tools running in CI/CD Agent can be configured using the API provided by the Automation Server (i.e., CI/CD Manager). After being configured, these tests become available in the CI/CD Agent through several tools.

The automated execution of these tools is similar to the execution of the testing scripts. Thus, the execution of tests, using testing tools, is also enabled by the same components involved in the execution of the testing scripts: (i) the NODS, (ii) the CI/CD Manager, and (iii) the CI/CD Agents. After the execution of tests using testing tools, the TRVD is also responsible for presenting the results of the validation process to the end developers.

The execution of a new validation process is triggered by the NODS, for both scripts and tools. During this phase, the NODS posts a TMF payload (i.e., Testing Descriptors) to the CI/CD Manager's API, providing a set of tests to be executed by each tool and the test configuration parameters required to define the test cases in the related tool. To let the developers choose the testbed where they wish to deploy their NetApps, the NODS will provide information on the tools that are available in each testbed, allowing an informed decision by the NetApp developers. This information will also be available in 5GASP's Community Portal, to increase the engagement of the NetApp developers, regarding the 5GASP ecosystem. Through the community Portal, the NetApp developers will also be able to suggest the tools they wish to see available in our TVCS. Then, these suggestions will be analyzed by the 5GASP consortium and, if possible, the suggested tools will be made available in our testbeds.

After 5GASP NODS triggers a validation process on the CI/CD Manager, the Manager will create a test queue file with some testing configuration files for each tool, based on the Testing Descriptors onboarded on NODS, and submit them to the CI/CD Agents responsible for testing the NetApp. Then, the CI/CD Agent will configure the testing tools based on the received configuration files. Once this process is finished, the Agent will perform the tests via the tools one by one according to the test queue from the Manager. All the following activities are the same as the scripts.

It is also important to mention that, if the NetApp developer wishes to use the dedicated tools and/or the non-open-source testing tools which are not in the recommended tool list in

Table 4, the developer should install them in his testing VNF but not the CI/CD Agent.And it is highly recommended to re-use the existing testing tools running in the Agent if possible.

2.2.3 Automation Testing Architecture (full case)

The CI/CD Service architecture designed in 5GASP gives two ways of developing the tests of a NetApp. It is inspired by software development techniques in which some common modes of testing are unit and system-level tests. Developers can choose the option that better suits the testing needs of the service offered by their NetApp, or the one that is more aligned with their development methodology and philosophy.

This first one is with testing scripts, which are prepared by the NetApp developer using Python and adjusted to the Robot Framework and Jenkins solution provided by 5GASP. With this option, the developer can design fully customizable tests that can be adapted to the necessities of NetApp in an effortless way. This test script programming has the total flexibility of the Python ecosystem and the multiple libraries that it provides. Section presents the detailed architecture of the testing with the scripts.

The second way is the testing tools on the Testing VNF. This solution permits the preparation of test specific VNFs that can be onboarded together with NetApp itself and hosts testing tools or services to enhance the testing process. The adoption of this approach permits more realistic and flexible testing possibilities, as the deployed VNFs can act as actors in the NetApp test scenarios. Currently, CI/CD Agents are reused as the default testing VNF. Furthermore, as these VNFs are also possible to be prepared by the developers, they can be fitted to the specific needs of each NetApp. However, not only specific VNFs can be used, but general VNFs can also be designed to be included in this architecture. Such general-purpose Testing VNFs can be reused by multiple NetApp testers, easing the adoption of this technology, and leveraging the previous knowledge required to operate in a complex architecture.

The adoption of these two kinds of testing procedures by the 5GASP Automation Testing Architecture gives substantial benefits to the developers and experimenters, especially in

terms of flexibility and reusability. This is currently being reflected in the testing design and development of the NetApps of the project, which are easily adopting both methods in an effortless way.

Figure 5 presents the main architectural components involved in the automated execution of both the testing scripts and the testing tools.

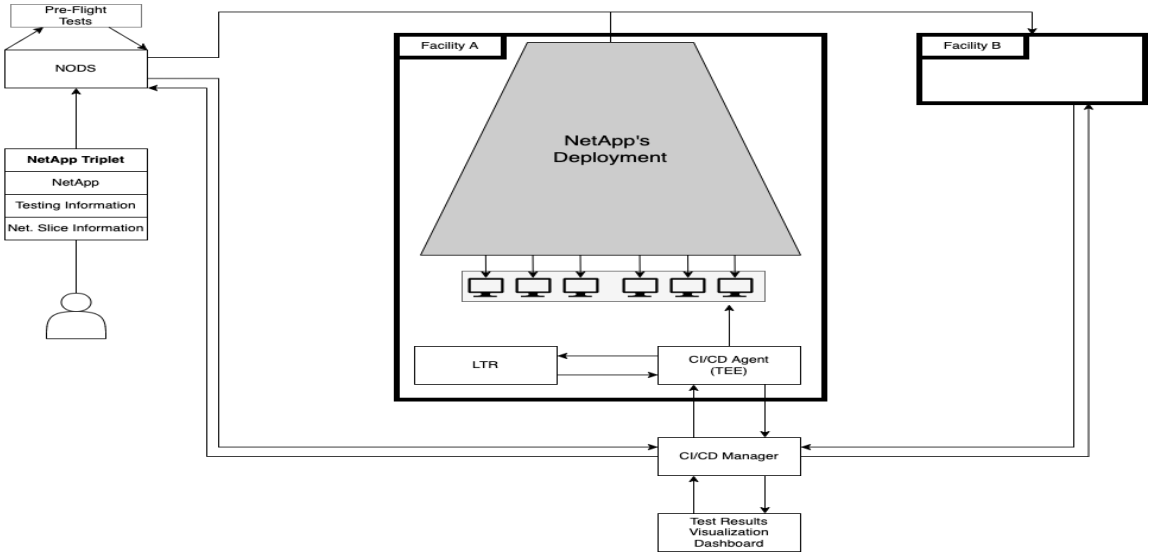


Figure 5 - Automation Testing Architecture

2.3. Automation Testing Implementation

In this section, the different components that formulate the 5GASP TVCS are detailed. The role of each one within the service operation is also defined.

As a reproducible environment, 5GASP TVCS defines a resource example template in Table 5 for all the 5GASP predefined components, except the tools which could be defined by the developers on testing demand. The minimum resource requests would be defined in the following subsections based on this resource template. 5GASP aims to create the Images for the predefined components as an open cloud platform.

Table 5 - 5GASP TVCS resource template with the example values

Virtual Resource Name	Resource Requested Value	Resource Requested Unit
vCPU Core	*2*	-
Virtual memory	*4*	GB
Virtual disk space	*10*	GB
Virtual Network interface	*1*	-

2.3.1 CI/CD Manager

The main components of the CI/CD Service are the CI/CD Manager, CI/CD Agents, Local Test Repositories, and the Test Results Visualization Dashboard (TRVD). Among these components, the most crucial component is the CI/CD Manager.

The CI/CD Manager orchestrates and coordinates all the CI/CD Agents that are responsible for validating the NetApps. The sequence begins by onboarding a NetApp to 5GASP Portal, which causes the pre-flight tests to take place. These tests validate only the structure of the onboarded descriptors, although not being a direct component of the CI/CD Service. If the pre-flight tests are passed, the NetApp can be deployed on the available testbeds. Regarding the deployment of the NetApps, a multi-domain scenario must be considered, which heavily

influences the design and architecture of the CI/CD Service. In the next subsections the remaining components constituting the CI/CD Service, mentioned above, will be described and explained.

This entity, implemented in FastAPI [8], is accessible via a Representational State Transfer (REST) interface and is triggered by the NODS, in order to start a new validation process. FastAPI is an open-source web framework used to build APIs in Python, characterized by its lightweight, performance, and intuitiveness.

The REST API has the following roles: (i) registering the CI/CD Agents, (ii) providing information about the tests available in each testbed, (iii) creating the Jenkins pipeline configuration files that will be submitted to the CI/CD Agents for the testing scripts, and/or the test queue file for the tools, (iv) continuously updating the status of a test and validation process, (v) retrieve information about the available testbeds, as well as registering new ones, and (vi) receiving and validating a TMF ServiceTest.

Figure 6 shows the documentation of the base functions of the CI/CD Manager's API.

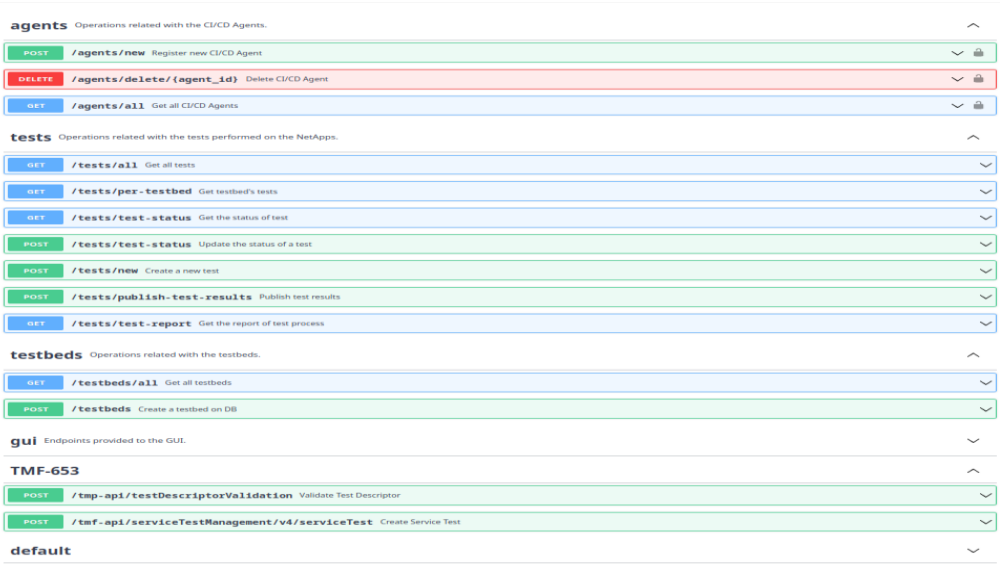


Figure 6 - CI/CD Manager's API Swagger Documentation

The CI/CD Manager receives the Testing Descriptors from the 5GASP NODS and based on these, it will start the activities, (i) creates a Jenkins pipeline configuration file for the testing scripts, (ii) creates a test queue file with some testing configuration files for each tool. Then, it will submit the files to the CI/CD Agent that will perform the validation process for both testing scripts and tools.

The Jenkins pipeline configuration file contains information about (i) how to set up the testing environment, (ii) how to obtain the tests from the LTR, (iii) how to perform these tests, (iv) how to update the status of a testing process continuously, and (v) how to submit the test results to the Visualization Dashboard.

The test queue file contains information about (i) how to set up the testing tools, (ii) how to enable the testing configuration files for each tool, (iii) how to perform these tests in each tool, (iv) how to update the status of a testing process continuously, and (v) how to submit the test results to the Visualization Dashboard.

Due to the lack of existing standards related to the definition of Testing Descriptors to specify the test procedures, there is the need to validate NetApps that requires a design of our own model. We have based our design on ideas from other projects that also worked on testing methodologies, such as 5G-EVE [9]and 5GTANGO [10]. The designed and developed Testing Descriptor is in its inception stage, as we are in the first year of the project. The designed Testing Descriptor will be used in conjunction with TMF's ServiceTestSpecification and ServiceTest models [7]. These models provide a standardized mechanism to place a test with all the necessary parameters to check the quality, performance, or reliability of a service. The onboarding and initial processing details of the ServiceTestSpecification is out of the scope of WP5, as it belongs to WP3. In summary, the Testing Descriptor is uploaded as an attachment of a TMF ServiceTestSpecification, which describes the main aspects of the testing process, such as information about the NetApp under study, the parameters to be configured, and the measures to be taken. Then, 5GASP NODS will process the specification and it will generate a TMF ServiceTest that will be forwarded to the CI/CD Manager. The NODS will be agnostic from the implementation details of the 5GASP Testing Descriptor.
The WP5 scope in this whole procedure is highlighted in Figure 7. The CI/CD Manager will receive the TMF ServiceTest. It will contain the information about the invocation of the test on a NetApp and the deployment data already filled by 5GASP NODS including all the information that has been gathered after the instantiation of the VNFs, NSs and the 5GASP Testing Descriptor. To make this possible, the Testing Descriptor needs to contain empty fields directly related to the deployment information filled by the NODS. Afterwards, the CI/CD Manager will contact the NODS to retrieve the ServiceTestSpecification attachment, the Testing Descriptor, and consequently fill the empty fields with the information gathered from the TMF ServiceTest.

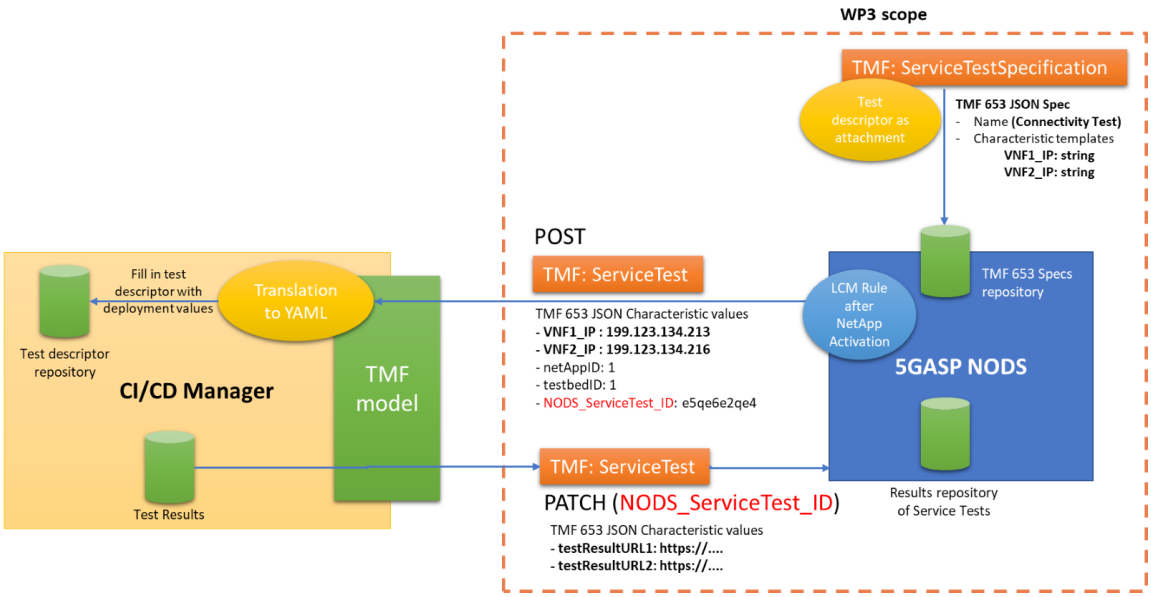


Figure 7 - Onboarding and Handling of the Testing Descriptor

Finally, once the Test Results are available, the ServiceTest will be patched on the NODS with URLs to the Test Results towards the Test Results Visualization Dashboard.

Currently, the CI/CD Manager service is running on IT Aveiro, and therefore, all management and maintenance required are taking place on this location. Its deployment is being performed through Docker Containers, which provide more Portability and Agility to the CI/CD Manager. In terms of resources, Table 6 lists the Docker Image minimum requires.

Table 6 - Minimum resource requires of CI/CD Manager

Virtual Resource Name	Resource Requested Value	Resource Requested Unit
vCPU Core	2	-
Virtual memory	2	GB
Virtual disk space	32	GB
Virtual Network interface	1	-

2.3.2 CI/CD Agent

Another component of the CI/CD Service is the CI/CD Agents, just as it was mentioned in the previous subsection. The CI/CD agents are responsible for the test execution of the NetApps for both testing scripts and tools. These Agents are deployed in the network slices provided by each testbed and heavily interact with the CI/CD Manager during the validation process. To provide the CI/CD Agents, Jenkins is currently being used, although any other CI/CD tool can be used since the CI/CD Service is highly modular and decoupled.

Once a CI/CD Agent is ready to receive work, it will register itself on the CI/CD Manager, submitting information such as its IP address and its login credentials. After this, the CI/CD Agent will be waiting for the CI/CD Manager to create test jobs. A test job is composed of the following steps: (i) setting up the testing environment (including the testing tools), (ii) gathering the test files from the LTR for the testing scripts and enable the testing configuration files for each tool, (iii) performing the tests, (iv) gathering the testing process output, and (v) submitting the output to the Visualization Dashboard.

Regarding the second step of the test job, in other words, the gathering of the test files from the LTR, as the LTR is password-protected, the CI/CD Manager will have to send the Agent the required credentials. On the other side, the tools are open for all tests. The tools are not password-protected, which means they are all without a password or sharing accounts.

On the test execution stage of testing scripts, Jenkins starts by setting some environment variables which will be consumed by the Robot Framework tests, using the Robot Framework test files retrieved on the previous stage, and then, executing the tests. To do so, back on the test job creation step, the CI/CD Manager uses the information present on the Testing Descriptors onboarded on NODS and injects the necessary parameters on the commands to be executed by the CI/CD Agents. Figure 8 presents a portion of a Testing Descriptor, where several parameters are defined, and an excerpt of its mapping in the Jenkins pipeline configuration.

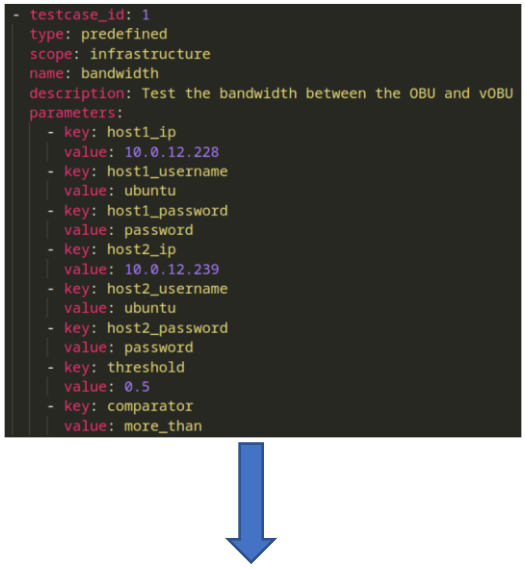


Figure 8 - Mapping of testing variables from TD to Jenkins pipeline configuration

On the test execution stage of testing tools, the Agent loads the configuration files for each testing tool as a testing VNF, and then, executing the tests one by one according to the test queue file. To do so, back on the test job creation step, the CI/CD Manager uses the information present on the Testing Descriptors onboarded on NODS and injects the necessary parameters on the configuration files to be executed by the tools on the CI/CD Agents. Each tool needs a dedicated configuration file. 5GASP is creating the configuration example files for the common open-source tools which are listed in section 2.3.5.

To install and integrate a new CI/CD Agent, firstly, an out of the box QCOW Image with Jenkins already installed has been provided, since it simplifies the process of configuration of this component. The environment where each CI/CD Agent will be installed will require the minimum requests in Table 7 for the test execution of both scripts and tools. Currently, the CI/CD Agents of all facilities share the same characteristics since the installation process has been equal in each case.

Table 7 - Minimum resource requires of CI/CD Agent

Virtual Resource Name	Resource Requested Value	Resource Requested Unit
vCPU Core	6	-
Virtual memory	16	GB
Virtual disk space	50	GB
Virtual Network interface	3	-

2.3.3 Local Test Repository

Following the Figure 5 - Automation Testing Architectur from the Automation Testing Architecture section, the Local Test Repository (LTR) is the component that provides proper interface to the the CI/CD agent on each testbed to connect, execute test files and testing scripts and get all necessary data (including non-open-source tool software) by using the testing framework. The LTR is only an FTP server as its architecture is defined in conjunction with the test cases to be executed, based on the CI/CD Testing Descriptors, information populated also with some testbed specific information, as testbed's location, 5G availability and capabilities and resources under validation. The CI/CD agent (e.g., Jenkins in this case), described previously, interacts with the LTR framework through REST APIs, executing the tests by creating a job who calls the Experiments assets, like test files, test images or testing scripts. Several steps are seen in relation to the LTR interaction with the Agent:

- Set the testing environment/testbed selection (resources)
- Get the LTR test (resources, test files, test images, scripts)
- Execute test based on the available descriptors
- Tests experiments processing, results
- Availability of the tests results for further visualization

Figure 9 presents the high-level architecture of the LTR.

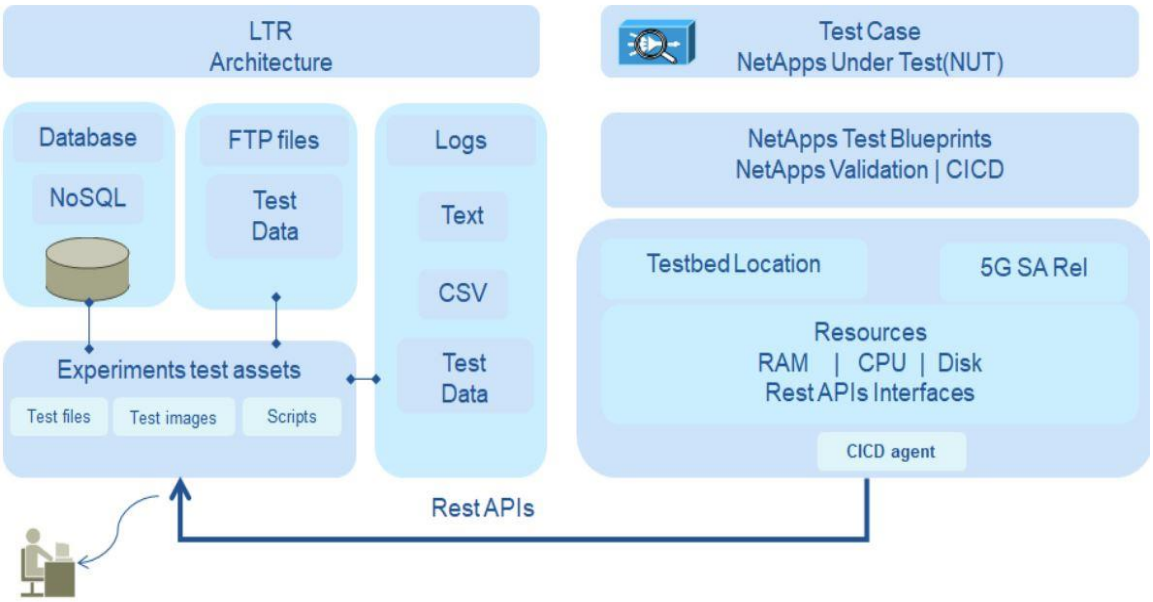


Figure 9 - LTR high-level architecture

In terms of resources, Table 8 lists the LTR minimum host hardware requirement.

Table 8 - Minimum LTR resource requires

Virtual Resource Name	Resource Requested Value	Resource Requested Unit
vCPU Core	2	-
Virtual memory	2	GB
Virtual disk space	50	GB
Virtual Network interface	1	-

The test assets are stored in a separate directory, as can be seen in

Table 9 and follows the following structure:

Table 9 - LTR directory structure

```
test-repo
├── custom
│   └── ...
├── data
│   └── ...
├── data-env
│   ├── environment1
│   │   └── ...
```

```
|  ├── environment2
|  |  └── ...
|  └── ...
├── images
|  └── ...
├── macros
|  └── ...
├── scripts
|  └── ...
└── tests
   └── ...
```

A REST API endpoint is exposed in the LTR (according to Figure 9) as a standardized interface that can be called from a specific CI/CD agent pipeline language to get specific tests and scripts, stored as objects in the different types of storage like a NoSQL database. The test files will be used by the agent in the entire testing framework.

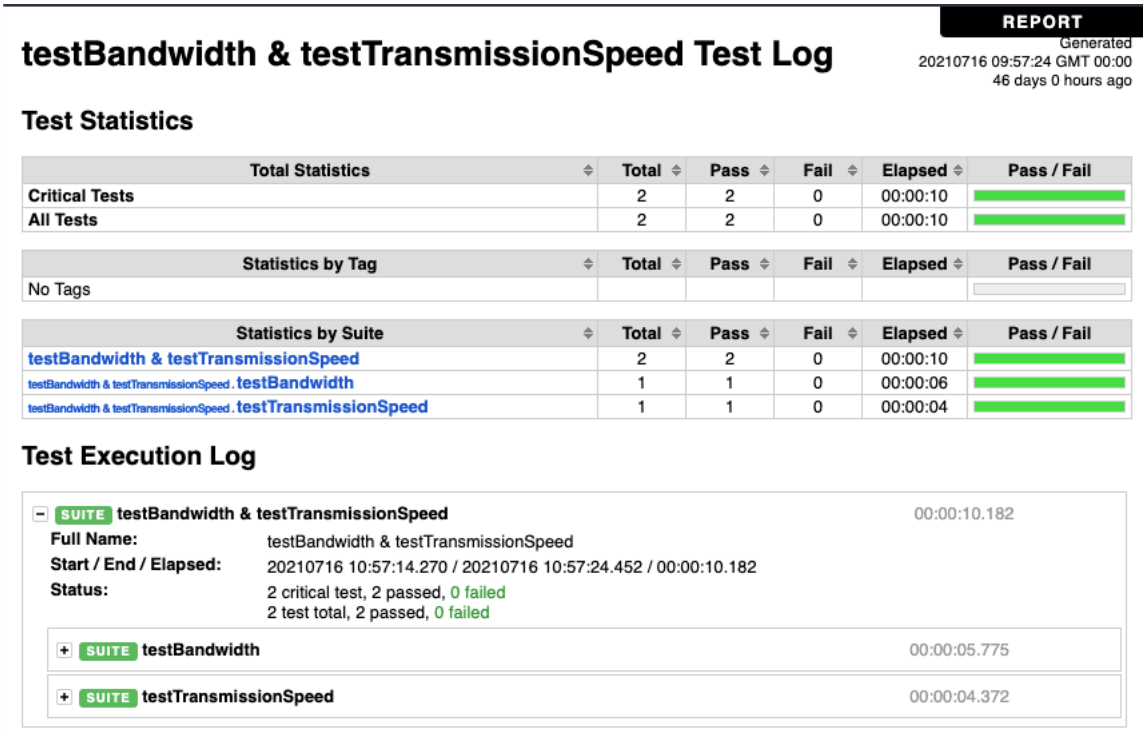
Using a GET HTTP method, a CI/CD agent can obtain scripts to apply it in the NetApp testing, according to the Testing Descriptor. The definition of these descriptors is detailed in WP4 deliverables.

Logging information and access information are stored in backend API files or different data sources and exposed to the users and developers. Some feasible examples of this logging system could be Kibana application with some ELK stack, with Filebeat - a software tool that is able to collect as an input a various type of logs.

2.3.4 Test Results Visualization Dashboard

The last component of the CI/CD Service is the Results Visualization Dashboard, composed of a WebUI and an FTP server. The TRVD allows the NetApp developers to check the outcomes and outputs of the testing phase with great detail. The implementation of this module was done using HTML, CSS, and JavaScript, as these tools are simple to use and lightweight compared to other frontend libraries. Besides this, to increase the performance of this component, and to serve static files on the WebUI itself, NGINX [11] takes place as a Reverse Proxy to the WebUI.

Once the testing phase is completed, the CI/CD Agents will submit the results to an FTP server. Robot Framework presents the results of the testing scripts in two different ways: (i) an XML file and (ii) an HTML file that provides the developers with a GUI to visualize the testing results. Figure 10 shows a rendered HTML results file from the Robot Framework.



testBandwidth & testTransmissionSpeed Test Log

REPORT
Generated
20210716 09:57:24 GMT 00:00
46 days 0 hours ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	2	2	0	00:00:10	
All Tests	2	2	0	00:00:10	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
testBandwidth & testTransmissionSpeed	2	2	0	00:00:10	
testBandwidth & testTransmissionSpeed . testBandwidth	1	1	0	00:00:06	
testBandwidth & testTransmissionSpeed . testTransmissionSpeed	1	1	0	00:00:04	

Test Execution Log

SUITE testBandwidth & testTransmissionSpeed 00:00:10.182
Full Name: testBandwidth & testTransmissionSpeed
Start / End / Elapsed: 20210716 10:57:14.270 / 20210716 10:57:24.452 / 00:00:10.182
Status: 2 critical test, 2 passed, 0 failed
2 test total, 2 passed, 0 failed

SUITE testBandwidth 00:00:05.775

SUITE testTransmissionSpeed 00:00:04.372

Figure 10 - Robot Framework HTML Output File

The WebUI of the Visualization Dashboard will use Robot's HTML files and render them. The Visualization Dashboard can be observed in Figure 11.

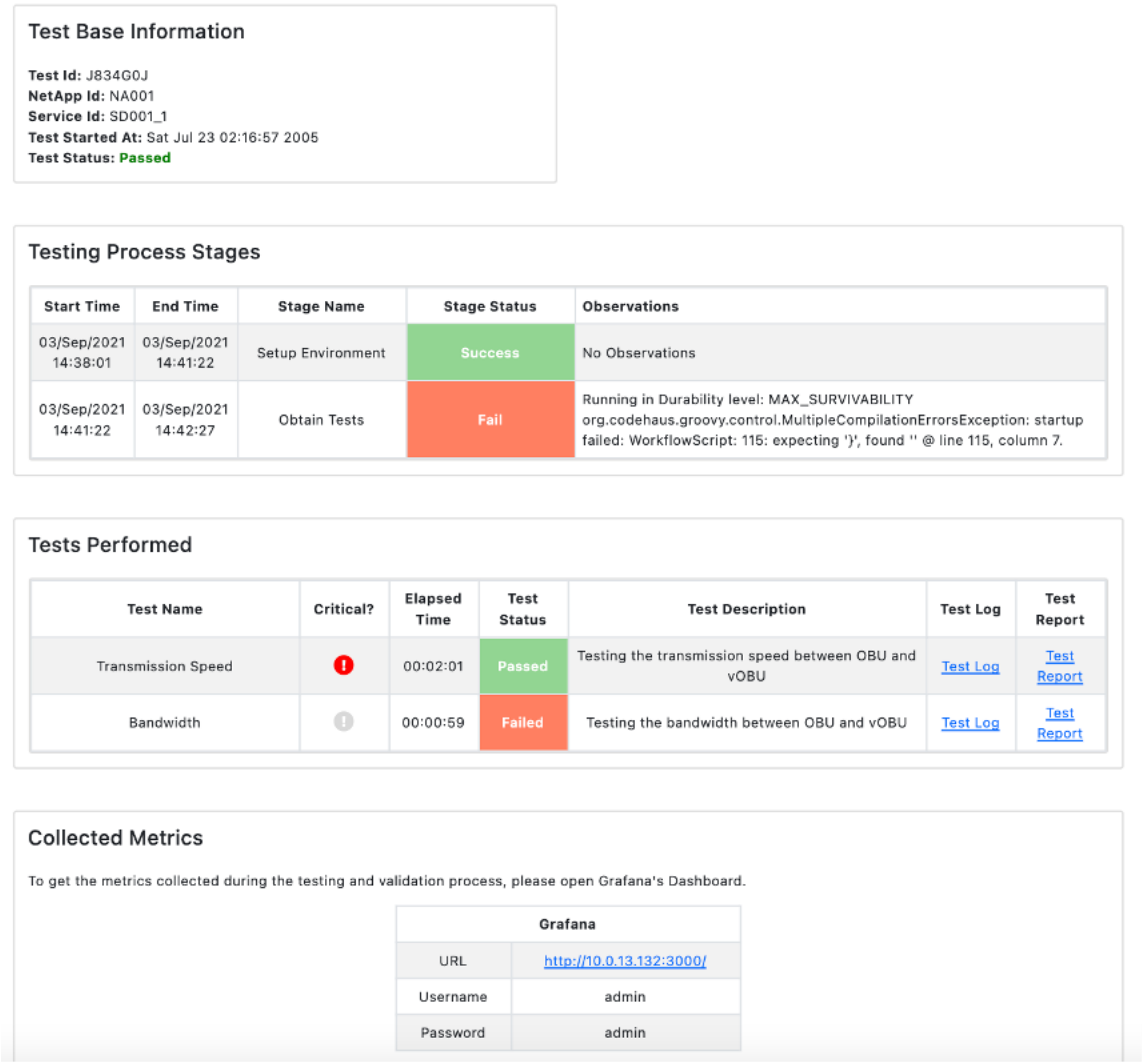


Figure 11 - CI/CD Service - Visualization Dashboard

The results of the testing tools are presented in different ways according to the tools. The CI/CD Agents will store the results in the CI/CD Agent and then share one URL for each tool to the dashboard.

Finally, this dashboard also supplies a URL with the results for each of the tests performed in a pipeline, which could be from this TRVD or CI/CD Agents. These URLs will be used by the CI/CD Manager to patch the TMF ServiceTest on the NODS, as it was mentioned in section 2.3.1.

In like manner to the CI/CD Manager, the TRVD service is located on IT Aveiro using Docker Containers. In terms of resources, Table 10 lists the TRVD Docker Image minimum requires.

Table 10 - Minimum TRVD Docker Image resource requires

Virtual Resource Name	Resource Requested Value	Resource Requested Unit
vCPU Core	2	-
Virtual memory	2	GB
Virtual disk space	12	GB
Virtual Network interface	1	-

2.3.5 Testing tools

Multiple tools will be used to test and validate the successful deployment and operation of the NetApps. Each tool will address different test layers and domains, providing a complete set of instruments to cover all the necessities of the 5GASP experimenters. In the following tables a list of the initial set of offered tools is presented. The listing is divided between the common tools offered by all testbeds and the dedicated tools, offered in certain facilities, or dedicated for a certain NetApp. These lists are for the current tools in 5GASP project until D5.3 issuing. More tools would be expected to be added to the updated lists in D5.4 and D5.5 in future.

Table 11 lists the current common tools, which could be used for all NetApps in any testbed of 5GASP project.

Table 11 - 5GASP common tools

Tool Name	Test Layer	Test Domain	Open Source	External Interface
iPerf	Network	E2E traffic performance	Yes	Open API
Fping	Network	E2E traffic performance	Yes	Open API
TRex	Network	E2E traffic performance	Yes	Open API
Jenkins	CI	Application	Yes	Open API
Grafana	NFVI	Infrastructure Dashboard	Yes	Open API
Prometheus	NFVI	Infrastructure Metrics	Yes	Open API
Curl	App	Application	Yes	Open API

Table 12 lists the current dedicated tools, which are running in one of the testbeds or the labs of the partners. The tools in the labs of the partners could be shared with the selected testbed if needed.

Table 12 - 5GASP dedicated tools

Tool Name	Test Layer	Test Domain	Open Source	External Interface	Comments
qMON	Network Slice Application	E2E traffic NFVI performance	No	Non-Open API	Testbed Ljubljana
ML Model Validation	App	Wireless handover	No	Non-Open API	From NetApp 7

					- Efficient MEC handover
Nats.io	App UE	E2E traffic performance	Yes	Open API	Testbed Patras
AutoTool	MANO	MANO interoperability	No	Open API	EANTC Lab
Amarisoft Simbox	App	Application	No	Non-Open API	Testbed Murcia

3. 5GASP Test Plan

The 5GASP NetApp Test Plan defines the testing procedures that will be performed on NetApp to validate both its deployment and its proper operation. The Test Plan includes the test cases that NetApp will have to successfully pass based on the different test goals, defining the execution order, the KPIs and the expected values.

3.1. 5GASP Test Scope

5GASP provides a flexible testing methodology that presents in section 2.1. According to the different test goals, the test scope could also be flexible - fixed scope or open scope.

Table 13 presents the details of the flexible test scope.

Table 13 - 5GASP flexible test scope

Test Goal	5GASP Certificate	5GASP Showcase
Scope Type	Fixed scope	Open scope
From whom	5GASP Certification Authority	5GASP NetApp Developer
Test Propose	Meet the minimum industry requests	Achieve the max industry requests as much as possible
Test Reference	NetApp Definitions in D2.1	Vertical business requests
Test Case list	Sections 3.2.1, 3.2.2	Sections 3.2.3.1, 3.2.3.2, 3.2.3.3

So far, it is planned that the 5GASP's Certification Pipeline offers the following criteria tests:

Table 14 - Certification Expanded Criteria Tests

ID	Definition	Technical Points	Test Points	Spec
1	Should deliver services to 5G Verticals	1. 5G Service - service (L7) via 5G mobile network (L1-4) - 5G performance KPIs (data rate, latency, packet loss, jitter, etc.) 2. 5G Network - 5G service KPIs (e.g., QoS, Service Setup Time, Service Availability, Service Reaction Time) 3. 5G Vertical - Vertical KPI requests are dedicated, it is not easy to define a common one with the acceptable expected value for all the verticals	1. L7 could be any protocol according to the dedicated NetApp. Not easy to define any common test point; NetApp has to be tested with customized test scripts. The same applies to L1 and L2. The test would focus on L3 and L4 with the minimum expected E2E value of these KPIs: **data rate, latency, packet loss, and jitter**. 2. Network KPI would be **one or two 5QI.**	5G PPP, 3GPP, NGMN, ITU-T
2	May consist of both software and hardware parts	1. Software - NetApp has to release 5GASP triplets: NSD/VNFD, NEST, TD 2. Hardware - some NetApps may need dedicated hardware (GPU, NPU, accelerator card, etc.), we will not test such dedicated hardware	**5GASP triplets** shall be mandatory on 5GASP Portal.	ETSI, GSMA
3	Must embrace the Service Based Architecture paradigm	1. SBI between NetApp and 5GS with the protocol stack IP/TCP/TLS/HTTP2 with JSON/Application 2. NetApp must do service registration and de-registration to NRF	**HTTP2/TLS/TCP** in the interactional messages.	3GPP TS 29.500 3GPP TS 23.501 3GPP TS 29.510
4	Should follow the NFV model	NSD/VNFD (YANG or TOSCA)	1. NSD/VNFD static validation (syntax, semantics, and references) between NODS and CI/CD Manager. 2. TD validation between NODS and CI/CD Manager.	ETSI IFA 011 ETSI SOL 005 ETSI SOL 001 ETSI SOL 006
5	May expose APIs to be consumed by other service consumers. The exposed APIs should be delivered in an Open API model and may follow the 3GPP recommended APIs for applications (i.e., 3GPP CAPIF, Service Enabler Architecture Layer for Verticals – SEAL)	1. NetApp would expose its APIs to NEF (as 3GPP CAPIF entity) 2. NetApp's API would follow OpenAPI Specification (later than 3.0.0), with HTTP/2 over TLS according to 3GPP SEAL. 3. 3GPP SEAL defines a lot of service APIs based on CAPIF and there are dedicated ones based on service (e.g., location management, group management, configuration management, identity management, key management, network resource management). It is not easy to define a common one for all NetApp.	1. **OpenAPI 'version'** in the interactional messages. 2. **HTTP2/TLS**/TCP in the interactional messages 3. NetApp sends a **service API publish request** to NEF (as 3GPP CAPIF entity) with the details of the service API. 4. NetApp receives "**Result: success**" from NEF. 5. doing registration and de-registration to 5GS.	3GPP TS 29.500 3GPP TS 23.501 3GPP TS 23.222 3GPP TS 29.549 IETF RFC 7540 IETF RFC 5246
6	NetApp may be part of one or more vertical application services	NetApp must deliver some 5G vertical services, at least to one vertical service.	doing registration and de-registration to 5GS.	5G PPP, 3GPP, NGMN

7	One or more services of NetApp may be attached to ~~one or more~~ 5G User Plane Function (UPF) data paths	NetApp must attach to some UPFs, at least to one UPF. There are 3 user plane interfaces in one UPF - N3 to RAN/UE, because not all NetApps request UE connection, we do not test the N3 interface. - N9 between I-UPF and UPF to DN, because not all NetApps request more than one UPFs, we do not test the N9 interface. - N6 to DN/Server, in our case, any NetApp should access UPF via the N6 interface for the data transformation. In the case of the N6 connection, the whole 5GS works as an overlayed L1/L2 transport network for NetApp/Server. We could test the connectivity of the IP layer between NetApp/Server and UPF.	1. **Ping N6 IP address** from NetApp. 2. **Ping NetApp service IP address** from UPF.	3GPP TS 29.500 RFC 792
8	May be part of one or more 5G slices. The slices may be shared or not	1. NetApp has to be in some 5G slices, at least in one slice. 2. No matter if the NetApp is in one or multiple slices, the slice could be shared with more than one NetApp and/or 5G VNFs. It is not easy to define any common testing on this point.	1. **Slice Creation/Adaption Time** (E2E view, including 5GS preparation time for NetApp Slicing), the expected value here is the common one with the mix requested value of the current NetApps (i.e., **15 mins**). 2. **Slice quality** of the selected 5QI (i.e., Packet Delay, Packet Error Rate, Maximum Data Burst, Averaging Window).	GSMA, 3GPP
9	Part of a NetApp may reside at the User Equipment (UE) side. The part of the UE side may interact with a NetApp service that resides within the domain network. The UE part may follow the definition of the Vertical Application Layer (VAL) client of 3GPP	When Client (running on UE side), and Server (running on DN side), 3GPP VAL is recommended. VAL Client and Server should be connected via SIP protocol. BTW, there are 2 functional models, On-network via UU interface, Off-network via PC5 interface. We would like to test the On-network model at this moment.	1. UE sends **SIP INVIT** message to NetApp. 2. UE receives **SIP 200 OK** message from NetApp.	3GPP TS 23.434 IETF RFC 2543
10	May be part of the 5G Core. In such case, then it must follow the 3GPP standards	In such a case, NetApp has to work as a 5G Application Function (AF) 1. Controlling plane would use new N5 or existing Rx interface to PCF. Due to the definition of N5 is FFS in 3GPP, so we would like to follow Rx standard. 2. User plane would use N6 interface to UPF. N6 is already included in Def. 7.	1. NetApp sends **AAR message** to PCF for the session establishment. 2. NetApp receives **AAA message** from PCF, with **RAT type of NR** (5G New Radio).	3GPP TS 23.503 3GPP TS 29.513 3GPP TS 29.214

11	May interact with the 5G System by consuming 5G system's APIs, if the 5G system allows. When interacting with the 5G System, it must support relevant 3GPP standards. Such interactions may include location services, Quality of Services (QoS) management, Assured Forwarding (AF) traffic	There are 2 use cases from the view of NetApp. - NetApp is external 5GS, SEAL would be followed as Def. 5 - NetApp is internal 5GS, N5/Rx interface would be followed as Def. 10 In Def. 5 and 10, API and basic service messages were covered. So far, in this definition, it would be better to test some common basic services.	1. Location service would be tested under SEAL. - VAL client (no matter on UE side or not, Def. 9 would be followed) sends **Location reporting configuration request** to VAL Server (i.e., NetApp) - VAL Server (i.e., NetApp) sends **Location reporting configuration response** to VAL client. 2. Traffic management service (e.g., QoS, AF/EF forwarding) would be tested via Rx interface - NetApp sends **AAR message** to PCF with the values of the QoS parameters, i.e., **Max_DR_DL, Max_DR_UL, Gua_DR_DL, Gua_DR_UL, 5QI** **-** PCF guarantees the traffic in 5GS for the service flow of NetApp.	3GPP TS 23.434 3GPP TS 29.513 3GPP TS 29.214
12	May support service continuity by minimizing service interruption when transferring application context	Session and Service Continuity (SSC) would be used in this case - 'Session and Service Continuity support' in NEST according to GSMA NG.116 - 5GS (PCF, SMF, and UPF) guarantee connection continuity based on SSC	1. '**Session and Service Continuity support'** is used in NEST with one of the values (**SSC mode 1, SSC mode 2, SSC mode 3**). 2. Monitoring **packet loss** during VAL client handover in 5GS.	GSMA NG.116 3GPP TS 23.501 3GPP TS 23.502 3GPP TS 23.503
13	Software parts should be deployed either in a virtualized or containerized manner	According to Def. 2, 4, and this Def. 13, and the definition in D4.2, NetApp software would be deployed with NSD/VNFD packages. It would be either VNF with YAML format descriptors, or CNF based on K8s with TOSCA format descriptors	NSD/VNFD static validation (syntax, semantics, and references) on CI/CD Manager.	ETSI IFA 011 ETSI SOL 005 ETSI SOL 001 ETSI SOL 006
14	May have resource and network requirements in terms of hardware, memory, Graphics Processing Unit (GPU), Central Processing Unit (CPU), availability, etc.	The common resource requirements are mandatory in a VNFD, i.e., CPU, memory. This test is included in Def. 4. The acceleration resource requirement is optional in a VNFD, e.g., GPU, FPGA, NPU.	1. '**RequestedAdditionalCapabilityData**' is used to define the acceleration resource **in VNFD**. 2. VNF creation should fail when '**supportMandatory: TRUE**' and the requested acceleration resource is not available.	ETSI IFA 002 ETSI IFA 003 ETSI IFA 004 ETSI IFA 011
15	May have placement requirements (e.g., edge, region, core, etc.). Additionally, a network latency KPI must be specified by NetApp when requesting a slice by the 5G system	1. Placement requirement is managed via SOL005 with the parameter 'locationConstraints' of the context 'VnfLocationConstraint' according to 3GPP and ETSI. 2. Network latency requirement is covered in Def. 1 and 8.	1. SO (i.e., OpenSlice) sends '**locationConstraints**' to MANO (i.e., OSM, ONAP). 2. MANO receives 'locationConstraints' from SO, and **creates VNF** in the correct location (edge, region, or core).	3GPP TS 28.500 3GPP TS 28.531 ETSI IFA 011 ETSI SOL 005
16	May consume monitoring and telemetry data from the 5G System. Such data from the 5G System should be consumed by functions like the Network Data Analytics Function (NWDAF)	According to Def. 3, in this case, NetApp should work as an NWDAF consumer with an open SBI interface.	1. NetApp sends **Nnwdaf_EventsSubscription_Subscribe** request to NWDAF. 4. NetApp receives '**201 Created**' from NWDAF.	3GPP TS 29.520

17	May interact with the VIM/Container Infrastructure Service Management (CISM) of the domain, if this is not restricted	In this case, NetApp would follow ETSI IFA 005 when it works as NFVO, and ETSI IFA 006 as VNFM	Yet to be defined	ETSI IFA 005 ETSI IFA 006
18	May interact with the Service or NFV Orchestrator of the domain if this is not restricted	In this case, NetApp would follow ETSI IFA 007 and SOL 005 when it works as OSS/BSS	Yet to be defined	ETSI IFA 013 ETSI SOL 005
19	Should follow relevant 3GPP security definitions and recommendations	1. When CAPIF is used, one of the 3 TLS methods would be followed: Pre-Shared Key (PSK), Public Key Infrastructure (PKI), or OAuth token. 2. When SBI is used, HTTP/2 would be used with TLS. TLS would use one of the 3GPP profiles, Profiling for TLS 1.3, or Profiling for TLS 1.2; mutual authentication between NRF and NetApp shall be used. 3. OAuth 2.0 should be used for service security between NRF and NetApp.	1. HTTP2 in interactional messages. 2. TLS 1.2/1.3 in the interactional messages. 3. PSK/PKI/Oauth for TLS security in the interactional messages. 4. OAuth 2.0 should be used in the case of the message between NRF and NetApp.	3GPP TS 33.122 3GPP TS 33.501 3GPP TS 29.510 IETF RFC 7540

The testing of the showcase is an open scope testing. 5GASP NetApp developers could extend it in the below cases.

- The existing NetAPP provides any new service in one new version
- Any new NetApp provides new services in the existing Verticals, i.e., Automotive, PPDR.
- Any new NetApp provides new services in one new Vertical.

3.2. 5GASP Test Case List

5GASP defines the test cases based on the test references in section 3.1. The following subsections present the lists of the predefined test cases so far. The lists are open until 2 months before the end of the project. In each of 5GASP test case list, the well-defined fields are expected as follows. Some other fields (e.g., Expected Value) could be added and defined by the test designer.

- Test Case Name - the name of the test case
- Test Description - the detailed description of the test case
- Test Points - the measurable technical check points during the testing
    - Test Method - it should be either a) Automation or b) Manual. 5GASP aims to execute the testing automatically as much as possible.
- Test Format - it should be either a) testing scripts) or b) testing tool xxx). Hereby, xxx is a tool name, e.g., iPerf.
- Test Priority - it should be one of the low 4 priorities: (i) P0 basic test cases, (ii) P1 recommended test cases, (iii) P2 optional test cases, (iv) P3 conditional test cases.

Table 15 presents a template of the test case list with the example value of the well-defined fields. The template should be used for 5GASP certification criteria and recommended for the vertical test case list.

Table 15 - 5GASP test case list template

Test Case Name	Test Description	Test Points	Test Method	Test Format	Test Priority
E2E Packet Data Rate (UL and DL)	*Verify End-2-End Packet Data Rate of both Uplink and Downlink from the view of NetApp*	*1. End-2-End Uplink Packet Data Rate (Mbps) 2. End-2-End Downlink Packet Data Rate (Mbps)*	*Automation*	*testing script*	*P1*

3.2.1 Basic testing - Certification initial criteria

The certification initial criteria contain the test cases on the priority P0 – basic test cases. Any NetApp developer must run these test cases and successfully pass all of them. P0 test cases are the baseline to continue the other tests. One NetApp has the capability to be deployed on a 5G ecosystem and run in a safe way. P0 tests focus on the items below.

1. NFV Model testing - it validates one NetApp with the 5GASP requests, for example, 5GASP triplets in D4.1, Open APIs in D5.2, etc.
2. basic security testing - it validates one NetApp with the minimum application security requests, for example, HTTPS, FTPS, etc.

So far, it is planned that the 5GASP's Certification Pipeline offers the following expanded initial tests:

Table 16 - 5GASP certification initial criteria

Test Case Name	Test Description	Test Points	Test Method	Expected Value	Test Format	Test Priority
5GASP Triplet's check	5GASP NODS check whether all the requested packages are uploaded by NetApp developers	1. NSD/VNFD is uploaded 2. Test Descriptor is uploaded 3. Networks slice template NEST is selected	Automation	1. yes, in YANG or TOSCA format 2. yes, in YAML format 3, yes, in GST format	NODS Portal	P0
NFV Descriptor validation	5GASP NODS check whether the uploaded descriptors meet the drafted requirements (e.g., SOL006 for OSM 10 descriptors)	Static syntax checking on the uploaded descriptors	Automation	Valid/Invalid	NODS Portal	P0
Testing Descriptor validation	5GASP NODS check whether the uploaded Testing Descriptor (YAML) is valid against robot framework's requirements	Static syntax checking on the uploaded descriptor	Automation	Valid/Invalid	NODS Portal via request to CI/CD Manager	P0
Open port check	Verify if a NetApp is only exposing the ports that	1. Is a certain port open (True/False)	Automation	1. False	testing script	P0

	should be exposing, and no other communication port is opened					
Dictionary Attacks - SSH Access	Verify if a VNF SSH password can be cracked using a dictionary attack	1. Is it possible to obtain SSH access to a VNF, using a dictionary attack (True/False)	Automation	1. False	testing scripts	P0
HTTPS check in external APIs	Verify if a VNF is only exposing HTTPS connection in the external APIs to the other VNF, but not HTTP	1. send HTTP connection request to each external API in the list 2. send HTTPS connection request to each external API in the list	Automation	1. HTTP connection request: rejected 2. HTTPS connection request: successful	testing tool TRex	P0

3.2.2 Value added testing - Certification expanded criteria

Besides the test mentioned in Section 3.2.1, the NetApp developers will also have the possibility of executing some additional tests. These are not mandatory in the scope of the 5GASP Validation Pipeline, but provide additional, and important, feedback regarding the NetApps' status. In general, the presented test cases can be split between automated and manual execution scenarios. Furthermore, based on the priority, test cases are categorized as follows:

- P1 – recommended test cases, meaning each NetApp developer should run these test cases if possible since they provide some basic network performance KPIs.
- P2 – optional test cases that should be used by the NetApp developer if the NetApp is using or is dependent on the tested service (e.g., DNS name resolution).
- P3 – conditional test cases can be executed by the NetApp developer in selected test beds providing certain functionalities required to run the test scenario.

So far, it is planned that the 5GASP's Certification Pipeline offers the following expanded criteria tests:

Table 17 - 5GASP certification expanded criteria

Test Case Name	Test description	Test points	Test Method	Expected Value	Test Format	Test Priority
E2E TCP-based IP Traffic Rate (UL/DL)	This test validates that the infrastructure of 5GS can meet the minimal requirement of TCP-based IP traffic rate for the NetApp service traffic.	1. End-2-End Uplink IP/TCP Packet Data Rate (Mbps) 2. End-2-End Downlink IP/TCP Packet Data Rate (Mbps)	Automation	1. Uplink 50 Mbps 2. Downlink 100 Mbps	testing script	P1
E2E UDP-based IP Traffic Rate (UL/DL)	This test validates that the infrastructure of 5GS can meet the minimal requirement of UDP-based IP traffic rate for the NetApp service traffic.	1. End-2-End Uplink IP/UDP Packet Data Rate (Mbps) 2. End-2-End Downlink IP/UDP Packet Data Rate (Mbps)	Automation	1. Uplink 50 Mbps 2. Downlink 100 Mbps	testing script	P1
E2E Packet Latency	This test validates that the infrastructure of 5GS can meet the minimal requirement of Packet Latency for the NetApp service traffic.	1. End-2-End User Plane Packet Latency (ms) 2. End-2-End Control Plane Packet Latency (ms)	Automation	1. User Plane 4 ms 2. Control Plane 20 ms	testing script	P1

E2E Packet Loss	This test validates that the infrastructure of 5GS can meet the minimal requirement of Packet Loss for the NetApp service traffic.	1. End-2-End Uplink Packet Loss (%) 2. End-2-End Downlink Packet Loss (%)	Automation	1. Uplink 0.025 % 2. Downlink 0.025 %	testing script	P1
E2E Packet Jitter	This test validates that the infrastructure of 5GS can meet the minimal requirement of Packet Jitter for the NetApp service traffic.	1. End-2-End User Plane Packet Jitter (ms) 2. End-2-End Control Plane Packet Jitter (ms)	Automation	1. User Plane 1 ms 2. Control Plane 10 ms	testing script	P1
E2E Frame Loss	This test validates that the infrastructure of 5GS can meet the minimal requirement of Frame Loss for the NetApp service traffic.	1. End-2-End Uplink Frame Loss (Number) 2. End-2-End Downlink Frame Loss (Number)	Automation	<50 frame skips per hour	testing script	P1
E2E DNS Resolution Latency	This test validates that the infrastructure of 5GS can meet the minimal requirements to provide DNS name resolution to the NetApp service components	1. End-2-End User Plane DNS resolution (ms) 2. End-2-End Control Plane resolution (ms)	Automation	1. User Plane 50 ms 2. Control Plane 10 ms	testing script	P2
E2E Web Mean Opinion Score (MOS)	This test validates that the infrastructure of 5GS can meet the minimal requirements to use HTTP/HTTPS-based web services.	End-2-End User Plane WEB MOS	Automation/ Manual	MOS > 4	testing tool qMON	P2
E2E VoIP Packet Latency and Packet Loss	This test validates that the infrastructure of 5GS can meet the minimal requirements to provide VoIP-based services.	End-2-End User Plane (packet loss, jitter)	Automation/ Manual	Jitter < 1ms Packet loss < 0.025 %	testing tool qMON	P2
E2E VoIP Service	This test validates that the infrastructure of 5GS can support VoIP-based services.	End-2-End User Plane	Manual	PASS/FAIL	testing tool qMON	P3
E2E Messaging Service	This test validates that the infrastructure of 5GS can support messaging services.	End-2-End User Plane	Manual	PASS/FAIL	testing tool qMON	P3

3.2.3 Vertical Service testing

In this subsection, an overview of the service level testing for the automotive and PPDR verticals is presented. These tests focus on two main categories: application protocol, covering OSI layers from 5 to 7; and business call flows, which aim to test the behaviors of end users and application servers. In this section, service tests are classified depending on the use case: automotive, PPDR, or cross-vertical.

3.2.3.1 Automotive service test cases

In the context of the automotive ecosystem, service test cases are focused on the behavior of the end users of this vertical, which are the vehicles themselves and associated Road-side-Units (RSUs). In this way, testing is mainly related to On-Board Unit (OBU) register, handover between RSUs, application server operation, etc.

Table 18 lists a set of service test examples in the context of vehicular scenarios.

Table 18 - 5GASP automotive service test cases

Test Description	Test Points	Expected Value	Test Method	Test Priority
This test validates that the OBU of the vehicle can effectively register in the 5G network	Control Plane	PASS/FAIL	Automation	P1

This test validates that the OBU of the vehicle can effectively register in the 802.11p network	Control Plane	PASS/FAIL	Automation	P3
This test validates that the OBU of the vehicle has 5G connectivity	Control Plane	PASS/FAIL	Automation	P1
This test validates that the OBU of the vehicle has 802.11p connectivity	Control Plane	PASS/FAIL	Automation	P3
This test validates that the OBU of the vehicle can effectively handover among RSUs in the vehicular network	Control Plane	PASS/FAIL	Automation	P3
This test validates that the OBU of the vehicle can effectively handover among gNBs in the 5G network	Control Plane	PASS/FAIL	Automation	P2
This test validates that the application server can consume monitoring and telemetry data from the vehicular network performance	Control Plane	PASS/FAIL	Automation	P2
This test validates that the application running in the vehicle does not interrupt its service after a handover	Application Plane	PASS/FAIL	Automation	P3
Verify the system initiation - the modules are up and reporting as running	Control Plane	PASS/FAIL	Automation	P0
The streamer and connector are connected to relay and node and reporting as connected	Control Plane	PASS/FAIL	Automation	P2
A C-ITS message originated from the UE is received by other nodes	Application Plane	PASS/FAIL	Automation	P3
A C-ITS message originated from an external server is received by other nodes.	Application Plane	PASS/FAIL	Automation	P3
Under high traffic load, a C-ITS message originated from the UE is received by other nodes	Application Plane	PASS/FAIL	Automation	P3
A C-ITS message originated from the UE out of the coverage zone of the vRSU is not received by other nodes	Application Plane	PASS/FAIL	Manual/Automation	P3
A C-ITS message originated from a NetApp deployed on the edge is received by other nodes	Application Plane	PASS/FAIL	Manual/Automation	P3

3.2.3.2 PPDR service test cases

In the PPDR-based environment, the test cases should verify the end-to-end operation over the PPDR-specific slice and its features (i.e., IOPS operation) and provide additional end-to-end network performance KPIs. This involves connecting different types of PPDR UE devices that offer different levels of capabilities to use them in test scenarios (i.e., 5G smart phone running Android vs 5G industrial gateway running Debian/Ubuntu which can be used for test automation) on one side and testing IOPS scenarios where gNB reconnects to its own local IOPS CN after a connection to its primary CN fails.

The selected test cases are presented in

Table 19.

Table 19 - 5GASP PPDR service test cases

Test Description	Test Points	Expected Value	Test Method	Test Priority
This test validates that the 5G UE (smart phone) can register and attach in the 5G PPDR network	End-2-End User Plane	PASS/FAIL	Manual	P2
This test validates that the 5G UE (5G gateway) can register and attach in the 5G PPDR network	End-2-End User Plane	PASS/FAIL	Automation (under investigation)	P3

This test validates that the 5G PPDR slice provides IOPS functionality (i.e., gNB switch to IOPS CN after connection to primary CN fails)	Control Plane	PASS/FAIL	Automation	P3
This test validates that the 5G PPDR slice IOPS operation meets the requirements to detect 5G primary CN connection failure in under certain time	Control Plane	< 30 sec	Automation	P3
This test validates that the 5G PPDR slice IOPS operation meets the requirements to reconfigure and reconnect to IOPS CN in under a certain time	Control Plane	< 300 sec	Automation	P3
This test validates that the 5G UE (smart phone) can register and attach in the 5G PPDR network after a primary CN failure and gNB switches to its local IOPS core	End-2-End User Plane	PASS/FAIL	Manual	P2
This test validates that the 5G UE (5G gateway) can register and attach in the 5G PPDR network after a primary CN failure and gNB switches to its local IOPS core	End-2-End User Plane	PASS/FAIL	Automation (under investigation)	P3

3.2.3.3 Cross-vertical service test cases

In the context of the cross-vertical use cases, service test cases are focused on the behavior of the end users of both Automotive and PPDR verticals, which are the vehicles themselves and services providing. In this way, testing is mainly related to main system modules, 5G connectivity and synchronization, application server operation, etc.

Table 20 lists a set of service test examples in the context of cross-vertical scenarios.

Table 20 - 5GASP cross-vertical service test cases

Test Description	Test Points	Expected Value	Test Method	Test Priority
Verify the system initiation - the modules are up and reporting as running	Control Plane	PASS/FAIL	Automation	P0
The streamer and connector are connected to relay and node and reporting as connected	Control Plane	PASS/FAIL	Automation	P0
The system is synchronized using the cellular network	Control Plane	PASS/FAIL	Automation	P1
Connectivity to reporting service is established	Control Plane	PASS/FAIL	Automation	P1
Beginning of the video stream – the streamer is generating a video pattern pushed through the streamer and 5G system to the relay and node (terminated and measured)	End-2-End User Plane	PASS/FAIL	Automation	P1
Verify End-2-End Packet Data Rate of both Uplink and Downlink from the view of NetApp	End-2-End User Plane	1. Uplink >50 Mbps 2. Downlink >100 Mbps	Automation	P1

Verify End-2-End Packet Latency of both User Plane and Control Plane from the view of NetApp	End-2-End User Plane	1. User Plane <4 ms 2. Control Plane <20 ms	Automation	P1
Verify End-2-End Packet Loss of both Uplink and Downlink from the view of NetApp	End-2-End User Plane	1. Uplink <0.025 % 2. Downlink <0.025 %	Automation	P1
Verify End-2-End Packet Jitter of both User Plane and Control Plane from the view of NetApp	End-2-End User Plane	1. User Plane <1 ms 2. Control Plane <10 ms	Automation	P1
Verify End-2-End Frame Loss of both Uplink and Downlink from the view of NetApp	End-2-End User Plane	<50 frame skips per hour	Automation	P1
To verify the stability of the service while video BW is being changed according to the predefined pattern (between 2Mbps to 20Mbps)	End-2-End User Plane	PASS/FAIL	Automation (under investigation)	P1
This test validates the NetApp has REST interfaces of the EMHO NetApp that are accessible for the supported/enhanced NetApp using the 5G network	Control Plane	PASS/FAIL	Automation (under investigation)	P1
This test validates the availability of the interface of NetApp to receive 5G monitoring data	Control Plane	PASS/FAIL	Automation (under investigation)	P1
The test validates that the UEs or their associated proxies communicate with Privacy analyzer's inbound REST endpoint.	Application plane	PASS/FAIL	Manual	P0

3.3. Key Performance Indicators

This section introduces the KPIs that are validated by the 5GASP's Validation pipeline. Thus, this section emerges as a continuation of section 3.2, where the test cases performed by the 5GASP's Validation Service are addressed. Section 3.3.1 focuses on the KPIs used to evaluate E2E Networks, while Section 3.3.2 focuses on Verticals-specific KPIs.

3.3.1 E2E Network KPIs and the expected values

The 5GPPP Test, Measurement and KPI Validation Working Group (TMV-WG) identified an initial set of 5G KPIs. They were selected based on the feedback obtained from multiple 5GPPP Phase 1 and 2 projects which collaborated with the TMV-WG. In the following table, a set of 5G performance parameters related to the network and core components of the 5G infrastructure are presented. Furthermore, these KPIs are linked with the test cases as presented in past the section 3.2, alone with suggestions of tools that can be used for the measurements. Table 21 lists a set of E2E network KPIs and the related tools to measure the results.

Table 21 - 5GASP E2E network KPI List

KPI	Recommended Tools
E2E Network Latency	Iperf, ping, TRex
Network Packet Loss	Iperf, ping, TRex
Guaranteed Network Data Rate	Iperf, TRex

Availability	TRex, Curl
Network Slice Creation/Adaption Time	Jenkins
Connection Density	Jenkins
Network Data Volume	TRex, Jenkins
Network Jitter	Iperf, ping

3.3.2 Vertical Service KPIs and the expected values

Table 22 presents a set of performance parameters related to the vertical service KPIs. Furthermore, these KPIs are linked with the suggestions of tools that can be used for the measurements.

Table 22 - 5GASP 5GASP vertical service KPI list

KPI	Recommended Tools
E2E Application Latency	Iperf, ping, TRex
Application Packet Loss	Iperf, ping, TRex
Guaranteed User Data Rate	Iperf, TRex
Service Availability	TRex, Curl/Postman
User Frame loss	To Be Defined
User Data Jitter	Iperf, ping

4. Automation Testing Workflows

This section details the testing automation workflows, presenting all the involved stages from the onboarding of the NetApp triplet to the validation process results collection.

As stated throughout this document, all the validation process starts with the onboarding of a NetApp Triplet, composed by (i) the NetApp Entities (VNFs, NSs, etc.), (ii) information regarding the network slice where the NetApp will be deployed, and (iii) the testing information required for the validation process. After the onboarding of this bundle, via the NODS, a myriad of validations regarding the NetApp descriptors (i.e., NSD and VNFD) will have to be performed. The NODS will start by validating the VNFDs and NSDs through its Pre-flight check service. Then the Testing Descriptor will have to be validated. This validation is a mixed effort between the NODS and the CI/CD Service. Figure 12 details the onboarding and pre-flight validations that are executed in the scope of the 5GASP Validation and Certification Pipeline.

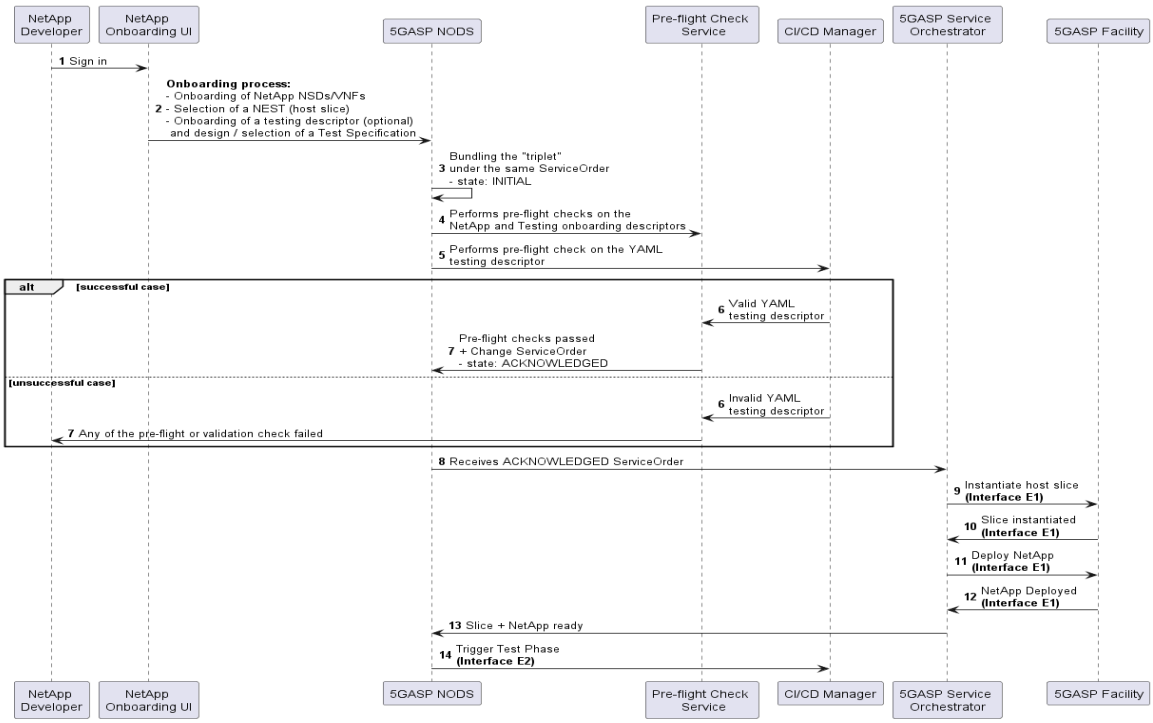


Figure 12 - Testing Automation - Onboarding, Pre-Flight Validations

If the pre-flight validations are successful, the NODs will orchestrate the deployment of the NetApp in a 5GASP testbed, which is also depicted in Figure 12. Only after the deployment of all the NetApp's components, a new validation process will be triggered. To achieve this, the NODS will create a TMF653 payload and send it to the CI/CD Service, which will then start a new validation process.

Since the validation process has already been described in this document, in this section, no details regarding it will be described. Only an interaction diagram is presented in Figure 13. Through this diagram it is possible to understand all the phases of the validation process.

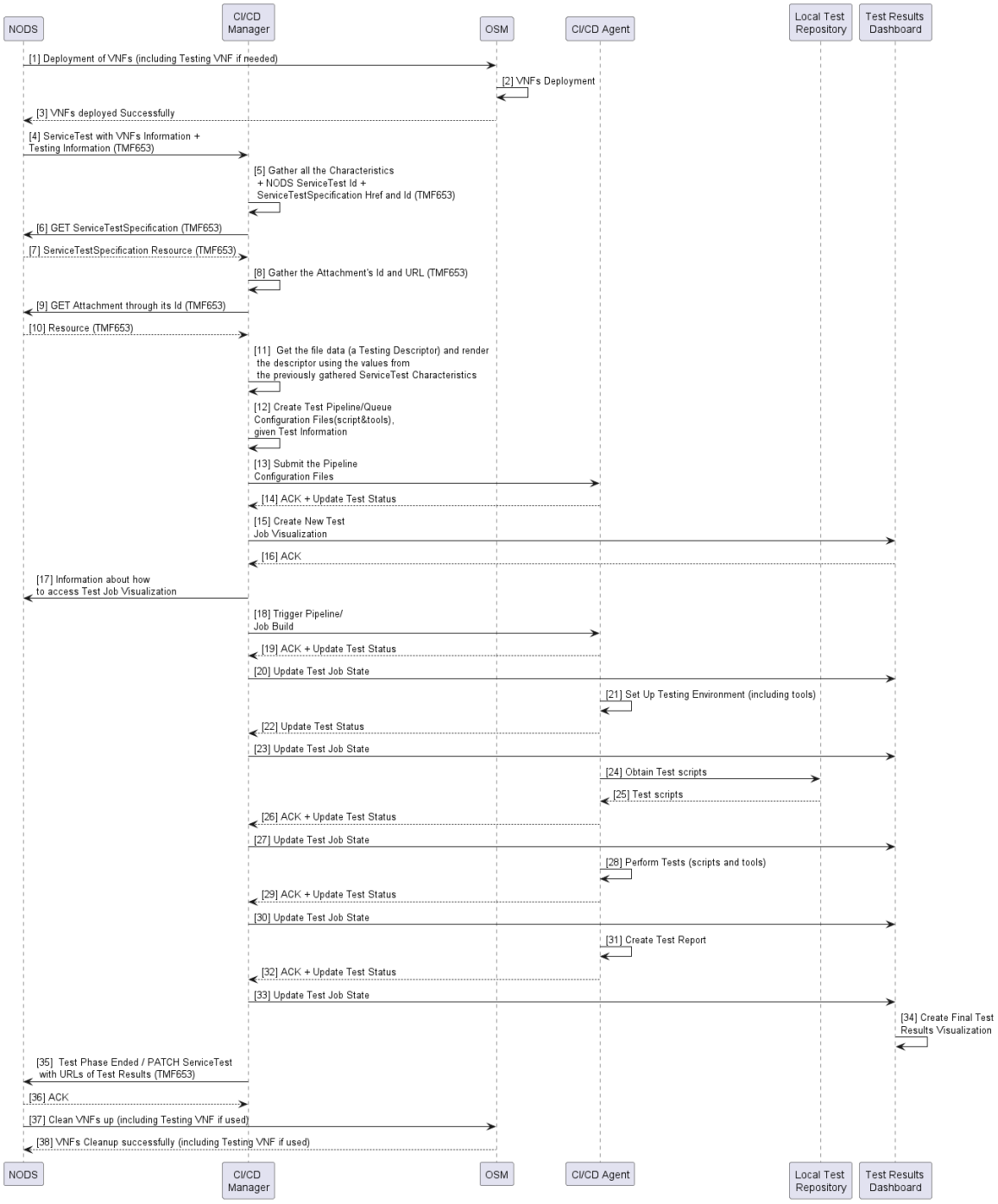


Figure 13 - Testing Automation – CI/CD Service Validation Process

It is worth presenting how the developer-defined testing scripts are obtained during the execution of the NetApp validation tests. This is presented in Figure 14.

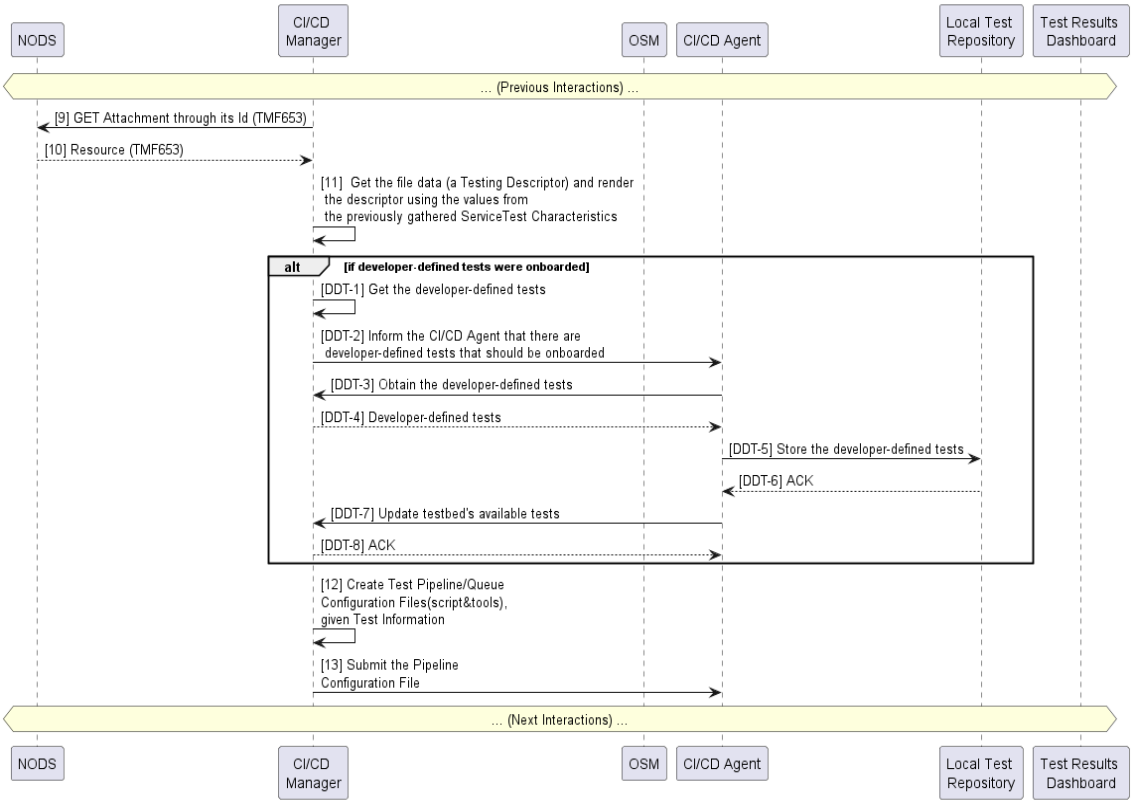


Figure 14 - Testing Automation – Developer-defined Test Scripts Gathering

Although, it is also worth presenting how the tests via the tools are obtained during the execution. This is presented in Figure 15.

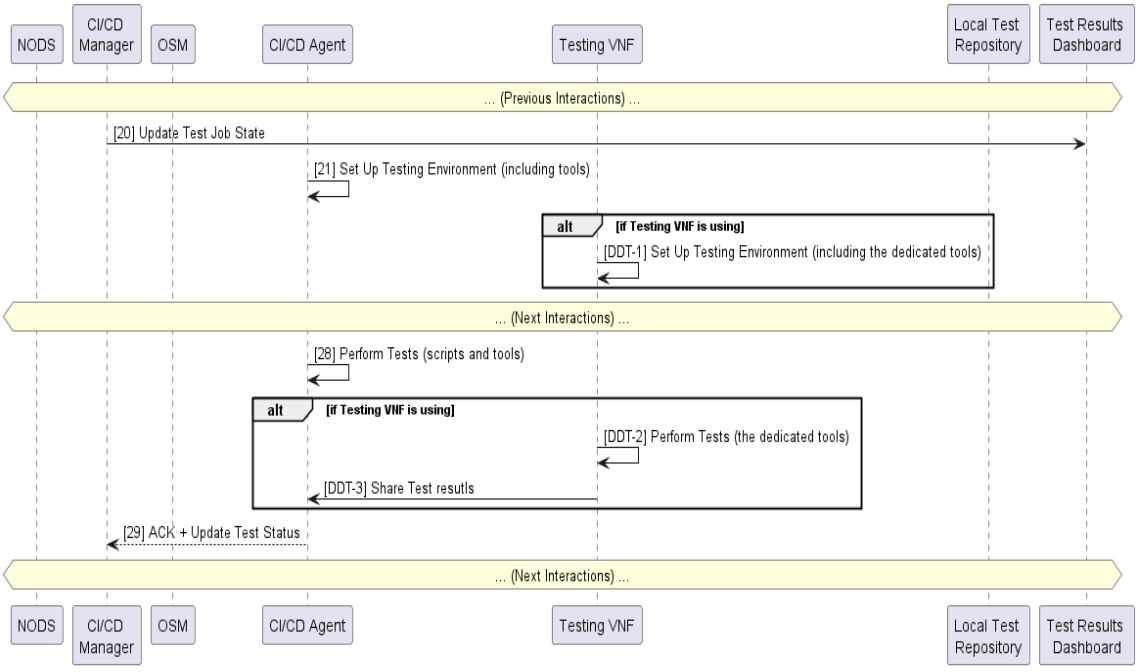


Figure 15 - Testing Automation – tools testing

5. Conclusions

The 5GASP testing methodology and test plan aim to comprehensively define how to test and validate the NetApps in the 5G ecosystem. The correct definition of these items is of utmost importance to provide a reliable and useful framework in the context of the 5GASP project. This is because a well-defined testing environment will enable the fast and easy migration of traditional applications and services to the 5G ecosystem, as the adopted CI/CD approach shortens the design, development, and testing process.

In this deliverable, the 5GASP TVCS is presented and detailed. The proposed service permits the flexible and dynamic instantiation of tests and considers the use of two different kinds of tests, namely, testing scripts and testing VNFs. This service is perfectly integrated into the 5GASP architecture defined in WP2 and compliant with the efforts made in the other WPs. The components of the architecture, as well as the interaction among them, are also detailed to give a clear view of the functioning. Also, the tools that will be used in the architecture to support and perform the testing are explored.

Besides, the 5GASP Test Plan is also presented. Its aim is to list down the test cases and the KPIs that will be used to evaluate the functionality, performance, scalability, and interoperability of the NetApps. The test plan is dissected and detailed depending on the test layer to which it applies. It covers from the basic testing of the NetApps to the vertical-specific testing in the context of the ecosystem in which NetApp will be deployed. In this case, the automotive and PPDR verticals are considered. Regarding the proposed KPIs to consider in the automation procedures, they are listed considering the E2E network communications and, again, the vertical to which the NetApp belongs.

Finally, the automation testing workflows are presented and detailed. Showing the full flow for both approaches (testing scripts and testing VNFs) to exemplify how the whole process works in the context of the 5GASP framework.

The next steps for WP5 will focus on applying this methodology and test plans to the NetApps of the project. In addition, this work will pave the way for the NetApp certification of the specific NetApps participating in the project.

References

[1] 5GASP, "D5.1 Initial Report on Test Plan Creation and Testing Methodologies," September 2021.

[2] 5GASP, "D5.2 Integration guide and API reference manual," January 2022.

[3] 5GASP, "D4.1 Initial Methodology for Designing and Developing NetApps," June 2021.

[4] 5GASP, "D4.2 Final Methodology for Designing and Developing NetApps," December 2021.

[5] A. Badkar and P. Krishnakumar, "Popular Test Automation Frameworks," July 2019. [Online]. Available: https://www.browserstack.com/guide/best-test-automation-frameworks.

[6] Robot Framework, [Online]. Available: https://robotframework.org/.

[7] TMF, "TMF653 Service Test Management API User Guide v4.1.0," April 2021.

[8] FastAPI, "FastAPI," [Online]. Available: https://fastapi.tiangolo.com. [Accessed 14 March 2022].

[9] 5G-EVE, "5G-EVE," [Online]. Available: https://www.5g-eve.eu/. [Accessed 14 March 2022].

[10] 5GTANGO, "5GTANGO," [Online]. Available: https://www.5gtango.eu/. [Accessed 14 March 2022].

[11] NGINX, "NGINX," [Online]. Available: https://www.nginx.com/. [Accessed 14 March 2022].