Test, Measurement, and KPIs Validation Working Group

White Paper

Basic Testing Guide

A Starter Kit for Basic 5G KPIs Verification

DOI: 10.5281/zenodo.5704519

URL: https://doi.org/10.5281/zenodo.5704519

Date: 30/11/2021 Version: 1.0

Executive Summary

This document (Basic Testing Guide) is a practical guide describing the starter kit developed in the context of the 5G PPP Test, Measurement and KPI Validation work group. The guide enables the interested developer to understand how this can be applied to measure and verify basic 5G KPIs. The document starts from describing the idea, the intention to measure, up to the actual realization of the test. To enable the test, a description of the environment, how to install it, the test tools and the methodology is provided. It guides the developer, step by step, to run the intended test. The proposed framework with the support of the described android agents can in principle support the validation of application level KPIs running at the android UEs, while for other application-specific KPIs running at other systems, the development of specialized plugins is necessary.

The 5G PPP Test, Measurement, and KPIs Validation Working Group

The Test, Measurement, and KPIs Validation (TMV) Working Group was founded as part of the 5G PPP effort to promote commonalities across projects that have strong interest in the T&M methodologies needed to provide support to the vertical use cases in the 5G Trial Networks. Such efforts include the development of Test and Measurement methods, test cases, procedures and KPI formalization and validation to the greatest possible extent, ensuring a unique European vision on how to support the entire lifecycle of the 5G network, from R&D to actual deployed environments.

The Group is comprised by several Phase II and Phase III 5G PPP projects, and it considers the following research areas and technology domains:

- Testing KPI definition, KPI sources, collection procedures and analysis
- Testing frameworks (requirements, environment, scenarios, expectations, limitation) and tools
- Testing methodologies and procedures
- KPI validation methodologies
- Testing lifecycle (i.e. testing execution, monitoring, evaluation and reporting)
- Common information models for 5G T&M

Another important topic is the use of and contribution towards open-source projects such as OSM, OPNFV or ONAP and identification of relevant exploitation and dissemination targets to promote the European vision on T&M towards a more global adoption.

Motivations

It is often the case that there exists a gap between theory and application which also is the case when it comes to KPI verification. This comes to show when studying material on KPI definitions and formalized test cases and considering how to apply this in a specific scenario. This process might cover tools selection, deployment and configuration, scripting and automation, test execution and result extraction. These steps might seem like a high entry barrier and require significant resources both in selecting and applying the approach. This means that there are many things that need to be in place even before starting to execute test cases.

This is the background for creating this basic testing bundle which will allow the test engineer to hit the ground running when starting to do KPI verification. The bundle provides the test engineer with an initial setup of tools and means to automate the control of the tools.

The main focus is to provide the foundation for automation of tests, which is critical when wanting to expand the basic tests with more advanced tests. This is also highly useful if integrating the testing as part of CI/CD pipelines for developing and deploying 5G VNFs or services.

Testing and Monitoring: Principles

Testing (or Active Testing) provides a greater observability due to the active control over the type and intensity of traffic that is pushed through the network and through subsets of the network elements. This provides more degrees of freedoms in selecting what can be tested and measured (e.g. scalability or security resilience). Monitoring is instead a generally passive process that is providing metrics from various components/layers of the 5G network. For this reason, the KPIs that can be measured via testing are substantially different than through monitoring alone. An example of these differences can be seen in two

similar documents coming from NGMN [1] and 3GPP [2]. The former is focused on testing aspects, while the latter provides an overview of KPIs to be measured during normal network operations.

Testing is normally applied during the network rollout, during the onboarding process of new Virtualized Network Functions (VNFs) or new software updates, and for the validation of newly deployed network services. Monitoring is instead constantly active during network operations and it is a key enabler of network management actions and processes.

Essential KPIs for Service Validation

The first essential step for creating the needed Test Cases is to identify which KPIs shall be stressed by the test. The priority was, for the TMV group, to identify those technical KPIs that were supporting the 5G PPP contractual KPIs validation [3]. The initial identified KPIs support mostly the following two contractual KPIs:

- **P1:** Providing 1000 times higher wireless area capacity and more varied service capabilities compared to 2010.
- **P4:** Creating a secure, reliable and dependable Internet with a "zero perceived" downtime for services provision.

In practice, those two contractual KPIs can be translated in a series of **Technical KPIs**, as displayed in Table 1.

Table 1 - Identified Technical KPIs

Type	KPI name	KPI measurement points	5G PPP KPI Validated
SLA	Minimum Expected Upstream Throughput	UE transmitting IP packets to the N6 interface.	P1
SLA	Minimum Expected Downstream Throughput	UE receiving IP packets from the N6 interface	P1
SLA	Maximum Expected Latency	RTT of UE IP packets transmitted to the N6 interface.	P1, P4
SLA	Network Reliability	Transport layer packets are lost between the UE and the N6 interface	P4
SLA, Technology Validation	Quality of Experience	Measured at the UE side at application or application API level	P1, P4
Technology Validation	UL Peak Throughput	Single UE transmitting IP packets to the N6 interface.	P1
Technology Validation	DL Peak Throughput	Single UE receiving IP packets from the N6 interface	P1

The definition of these KPIs has mostly been derived or reworked starting from [7] and [8]. The "Type" column remarks for which purpose the KPI is useful. The SLA KPIs are the ones used to validate if the service design can support the SLA agreed with the vertical, and they can used as well during the network

monitoring phases to trigger alarm and network management actions. The Technology Validation KPIs are instead focused on providing the proof that 5G is delivering the promised performances, mostly peak ones.

From Theory to Practice: Create Your Test Case

A test case is a formalized description of a known list of stimuli and observations to be used to evaluate whether a system operates properly. When creating a test case, a good starting point is to define what is being measured. Here Table 1 can be a good guide for choosing KPIs that are relevant to the scenario and system under test. Next the test case can be formally defined in terms of a list of inputs, execution conditions, testing procedures and methodologies, and expected results. In Table 2 is given a format for how to formalize test cases.

Table 2 - Test case formalization table

Test Case Name	Name of test case
Test Case ID	unique test case identifier
Purpose	Short description of the purpose and what is measured
Description	High-level description of the test case
Initial Conditions	List of conditions to be in place before executing the test
Parameters	List of parameters to be used in execution of the test
Procedures and Expected Results	Chronological list of actions to be carried out as part of the test execution, i.e. detailed list of actions.
Measurements	Description of the measured KPIs and how they are calculated

With the test case defined formally it can now be translated into a test plan consisting of several steps that define the inputs, perform the actions, apply the stimuli, extract outputs, and evaluate the results.

The test plan should be reproducible and should be straight forward to follow to perform the test actions. The sequence of steps in the test plan is what will be automatized such that it can easily be re-executed in the same context or in another context.

This translation is where OpenTAP comes into play and excels. A test plan in OpenTAP is comprised of a sequence of test steps, which each perform a specific action, often interacting with a specific tool. Furthermore, the resources to be used by the test plan are defined externally to the test plan as instruments and DUTs. This makes it easy to define new resources and apply the same test plan in a different context.

From Practice to Application: The “Hello Test” Toolkit

In previous section the process was explained of creating a formalized test case and how to translate into an actual executable test plan using OpenTAP. This section will provide the technical details of OpenTAP and plugins included in the basic testing bundle, and introduce the tools that can be controlled by the included plugins.

Test Automation Using OpenTAP

OpenTAP is an open-source software solution, developed originally by Keysight, for fast and easy development and execution of automated test and/or calibration algorithms. These algorithms control

measurement instruments and possibly vendor-specific Devices Under Test (DUTs). By leveraging the features of C#/.NETcore and providing an extendable architecture, OpenTAP minimizes the code needed to be written by the programmer. OpenTAP is distributed under the Mozilla Public License (MPL).

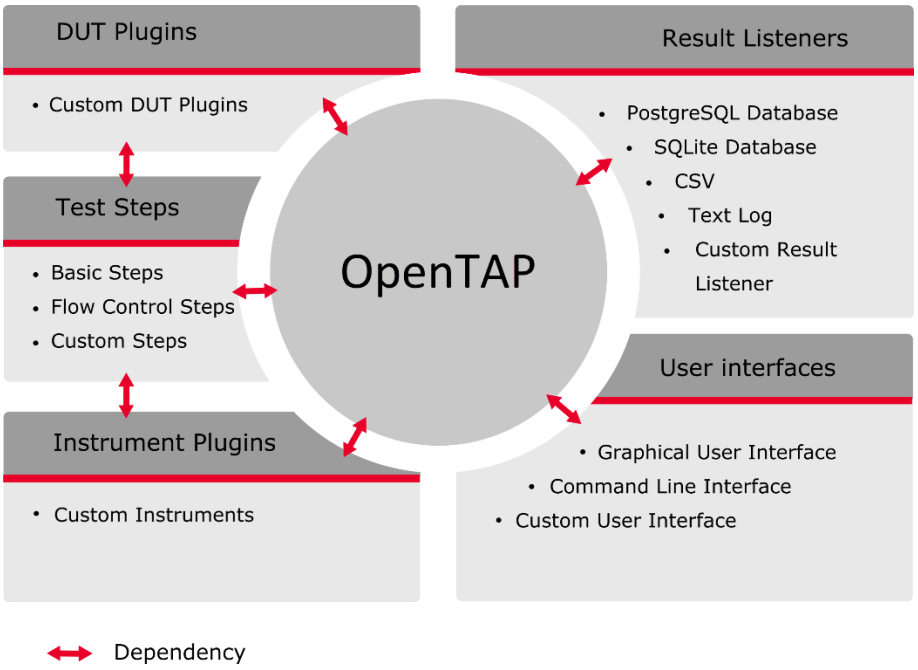
OpenTAP offers a range of functionality and infrastructure for configuring, controlling and executing test algorithms. OpenTAP provides an API for implementing plugins in the form of test steps, instruments, DUTs and more.

OpenTAP consists of multiple executables, including:

- OpenTAP (as a dll)
- Command Line Interface (CLI)
- Package Manager

Steps frequently have dependencies on DUT and instrument plugins.

The illustration below shows how OpenTAP is central to the architecture, and how plugins (all the surrounding items) integrate with it.



If a graphical user interface is needed you can download the Keysight Test Automation Developer's System (Community or Enterprise Edition). It provides you with both a Software Development Kit (SDK) as well as an Editor GUI. Please check the EULA is the use of Community Edition GUI applies to you.

An essential feature of OpenTAP is its flexible architecture that lets users create plugins. OpenTAP plugins can be any combination of Test Step, Instrument, and DUT implementations. Other OpenTAP components such as Result Listeners and Component Settings are also plugins. By default, OpenTAP comes with plugins covering basic operations, such as flow control and result listeners.

Plugins are managed by the Plugin Manager, which by default searches for assemblies in the same directory as the running executable (the GUI, CLI or API). Additional directories to be searched can be specified for the GUI, the CLI and the API.

A test plan is a sequence of Test Steps with some additional data attached. Test plans are created via the Editor GUI. Creating test plans is described in the Graphical User Interface Help, accessible within the Editor GUI. Test plan files have the .TapPlan suffix, and are stored as xml files.

Test Plan Control Flow

To use OpenTAP to its full potential, developers must understand the control flow of a running test plan. Several aspects of OpenTAP can influence the control flow. Important aspects include:

- Test plan hierarchy
- TestStep.PrePlanRun, TestStep.Run, TestStep.PostPlanRun methods
- Result Listeners
- Instruments and DUTs
- Test steps modifying control flow

TapPlan Settings

Editable OpenTAP step settings can be marked as TapPlan Settings. The value of such settings can be set through the Editor GUI, through an external program (such as OpenTAP CLI), or with an external file. This gives the user the ability to set key parameters at run time, and (potentially) from outside the Editor GUI.

OpenTAP Documentation

OpenTAP documentation is provided from the OpenTAP project [9]. Here is also provided developer guides and information about how to join and contribute to the community. Here can also be found links for the publicly available OpenTAP Package repository from where plugins can be downloaded.

OpenTAP Plugins Included in Basic Testing Bundle

In Table 3 are listed the plugins that are included in the Basic Testing Bundle and that are pre-installed in the docker container made available in the Gitlab project.

Table 3 - OpenTAP plugins included in Basic Testing Bundle

Name	Type	Description
TUI	Utility	The Text UI (TUI) plugin enables a text, terminal-based UI that allows graphical TapPlan composition in e.g. a container application.
SSH	Utility	Enables to connect and send generic shell commands to a remote system
ADB/Android	Testing tool	Allows to connect and control the UMA Android Testing App for E2E testing
IPerf	Testing tool	Allows control over generic IPerf client/server for throughput testing via SSH remoting
InfluxDB	Utility	Allows storing results as time series in an InfluxDB database
Yardstick	Testing tool	Allows controlling a Yardstick instance and executing test cases and test suites via REST API
OpenStack HEAT	Utility	Allows deploying entire stacks via HEAT orchestrator

Testing tools

Below are short descriptions of the tools that can be controlled by the OpenTAP plugins pre-installed in the Basic Testing bundle.

IPerf

iPerf3 is a tool for active measurements of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, buffers and protocols (TCP, UDP, SCTP with IPv4 and IPv6). For each test it reports the bandwidth, loss, and other parameters. This is a new implementation that shares no code with the original iPerf and is not backwards compatible.

iPerf features

- TCP and SCTP
  - Measure bandwidth
  - Report MSS/MTU size and observed read sizes.
  - Support for TCP window size via socket buffers.
- UDP
  - Client can create UDP streams of specified bandwidth.
  - Measure packet loss
  - Measure delay jitter
  - Multicast capable
- Cross-platform: Windows, Linux, Android, MacOS X, FreeBSD, OpenBSD, NetBSD, VxWorks, Solaris,...
- Client and server can have multiple simultaneous connections (-P option).
- Server handles multiple connections, rather than quitting after a single test.
- Can run for specified time (-t option), rather than a set amount of data to transfer (-n or -k option).
- Run the server as a daemon (-D option)
- Use representative streams to test out how link layer compression affects your achievable bandwidth (-F option).

Yardstick

Yardstick is an open-source project developed by the OPNFV community. The goal of the Yardstick Project is to verify the infrastructure compliance when running VNF applications.

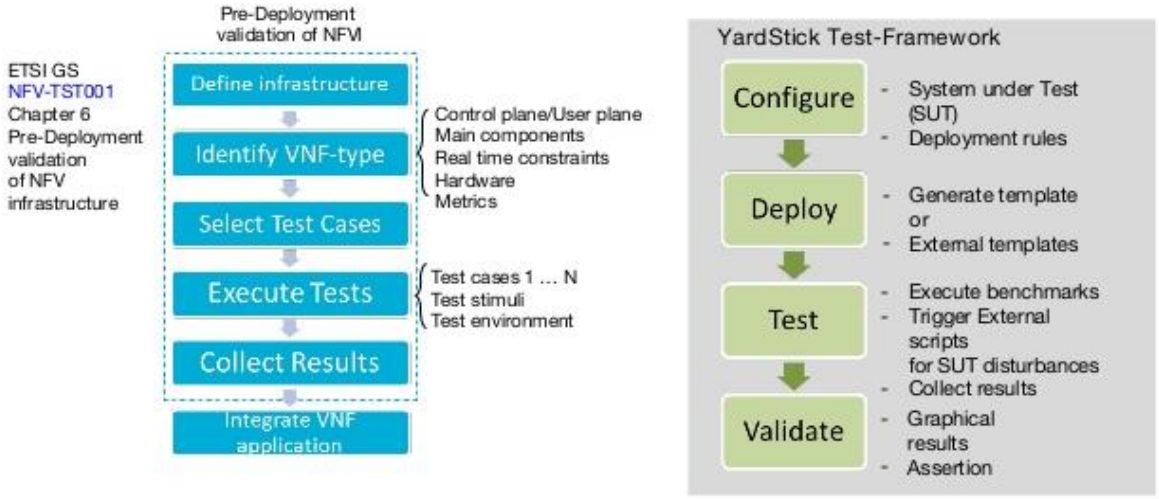
NFV Use Cases described in ETSI GS NFV 001 show a large variety of applications, each defining specific requirements and complex configuration on the underlying infrastructure and test tools. The Yardstick concept decomposes typical VNF work-load performance metrics into several characteristics/performance vectors, which each of them can be represented by distinct test-cases.

The project's scope is to develop a test framework, test cases and test stimuli.

Yardstick allows to:

- Decompose VNF work-load performance metrics into a number of characteristics / performance vectors, identifying and categorizing the metrics related to characterization of the infrastructure, develop test case examples to realize the metrics;

- Enable verification of more complex test cases by developing functionality to run: parallel testing, inject fault, test multiple topologies, test scenarios.



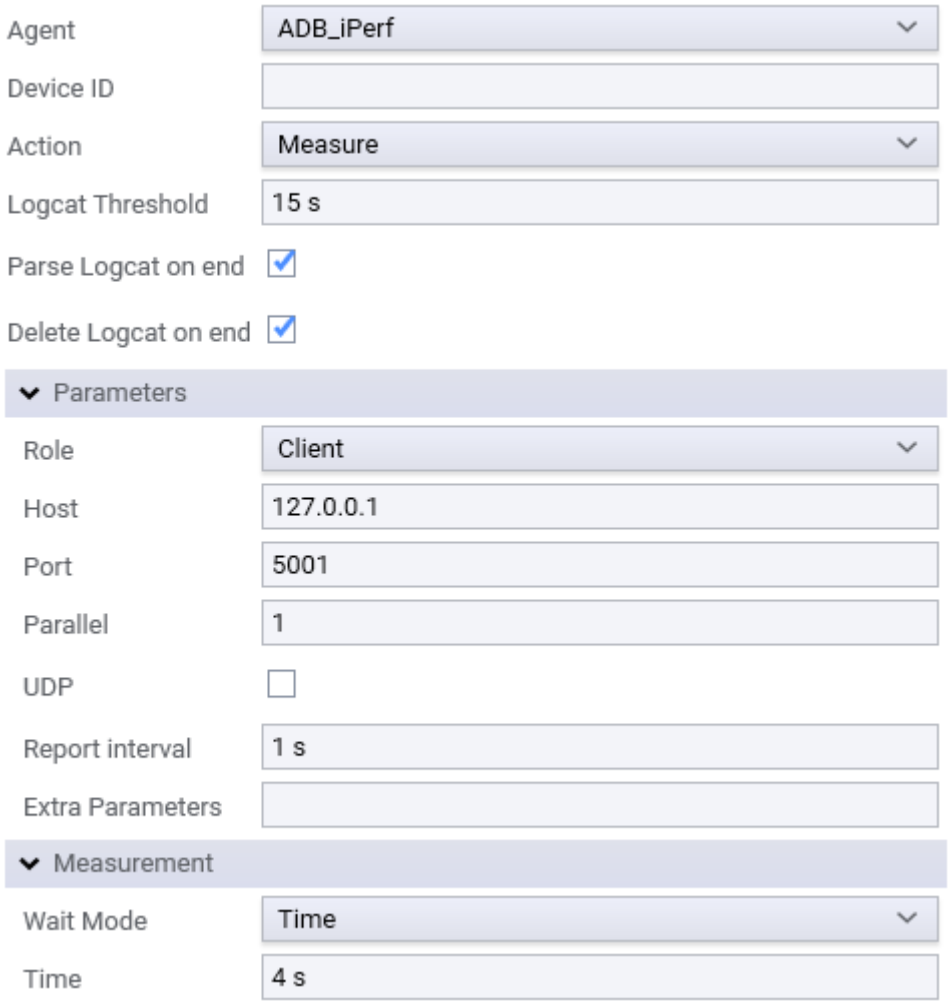
The methodology used by the Project, to verify the infrastructure form the perspective of a VNF, is fully aligned with ETSI NFV-TST 001. Visualization of the test results is performed using Grafana as shown below:

Android Agents

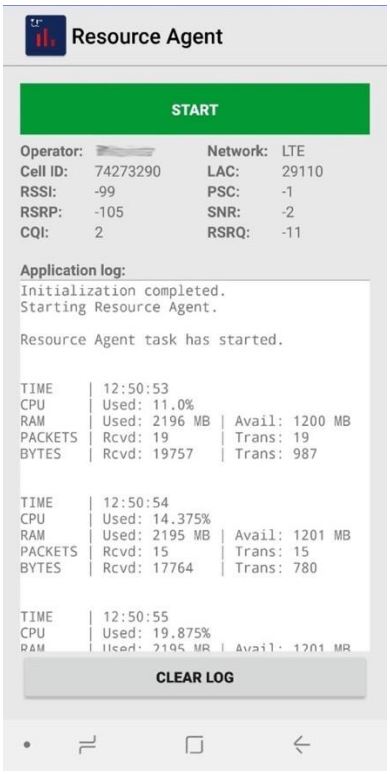
The Android Agents plugin contains steps and instruments for controlling four different agents that can be installed in Android devices. Additionally, the Android plugin, which is a requirement for the use of the plugin provides several generic steps for using android devices through ADB (Android Debug Bridge). The following agents are included:

- **iPerf Agent**: The iPerf agent is an Android application that acts as a wrapper for the iPerf measuring tool, which is bundled with the provided apk, and is able to control two separate instances at the same time, one acting as client and the other acting as server. The included test

steps allow the configuration of both instances separately, as well as the retrieval and publishing of the generated results.



- **Ping Agent**: Likewise, this agent also acts as a wrapper to the native ping utility included in Android devices, automatically generating results based on the output of the command.
- **Resource Agent**: The resource agent allows the monitoring of the radio parameters along with the usage of the device resources.



- **Exoplayer Agent**: Exoplayer is an alternative MediaPlayer API for Android devices, with support for DASH and SmoothStreaming adaptive playbacks. The Android Agents plugin provides a customized version of the demo application of the ExoPlayer library that is prepared for the generation of measurements. The included test step is able to initiate and stop the playback of a video using the application, as well as publishing the resources obtained.

Basic Testing Bundle installation and use

This section describes how to use the Basic Testing bundle. The basic testing bundle project along with documentation is hosted at Gitlab [10], from where the code and example files can be cloned.

Deploying Containers using Images

The project provides 2 container images; one with OpenTAP, and one with tools installed.

The OpenTAP container comes with a number of open-source plugins installed (also from the 5GENESIS project github.com/5genesis), which can be used to control and interact with tools or entities, but also with a TapPlans editor (TUI) to create and modify test plans.

The Tools container comes with ssh, iperf3 and ping-utils installed.

Deployment of the containers can be done using any container runtime, but as an example the project provides a docker-compose file. This file defines an example of how docker-compose can be used to deploy a simple infrastructure with 3 containers - 1 OpenTAP and 2 tools.

Install Packages via OpenTAP Package Manager

The project collects list of plugins that have been released as open source and are published in the package repository of OpenTAP.io. This list contains SSH, Iperf, TUI, Yardstick and OpenStackHeat plugins. This bundle can be installed using OpenTAP Package Manager, e.g. via command line "tap package install --dependencies --repository https://packages.opentap.io/ --version any HelloTestBundle".

The same approach can be used can be used to install additional OpenTAP plugins from OpenTAP repositories. Similarly packages can also be installed based on local .TapPackage files.

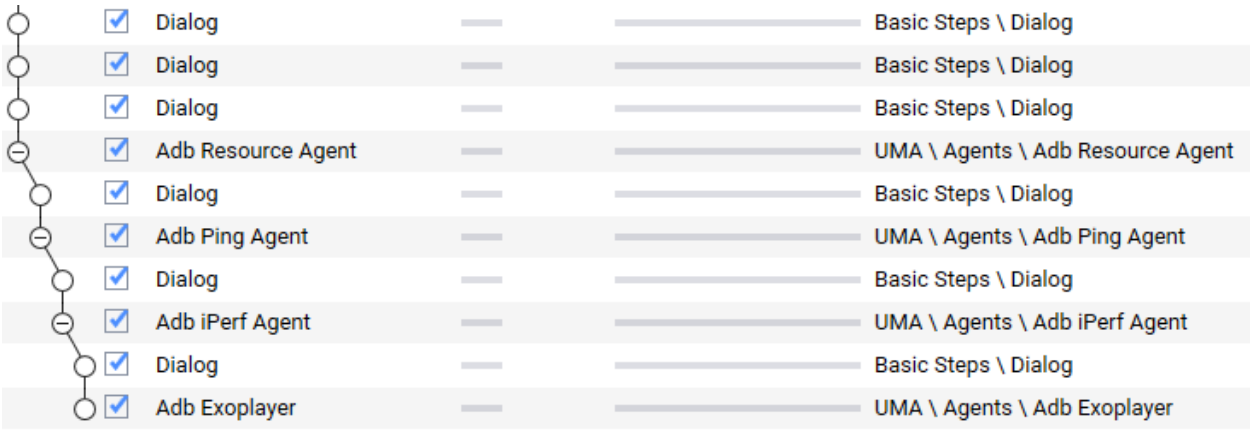
Run Packaged TapPlans

To execute TapPlans using the OpenTAP container provided in this context the TapPlan must be reachable for the tap application. The same requirement is true if a specific bench settings profile is used. The OpenTAP container comes with an example TapPlan and bench settings file that can be used to run test actions between tools. The example TapPlan can be executed using the command " tap run -v --settings /opt/tap/Settings/hellotest-bench -e "Test Duration DL"=12 -e "Test Duration UL"=14 /opt/tap/hellotest-iperf-example.TapPlan".

The example TapPlan performs an Iperf upload and download test, prints the unix name and performs a ping between the 2 deployed tools containers. The Iperf OpenTAP plugin allows for basic configuration of TCP/UDP uplink and downlink throughput tests initiated from an Iperf3 client and server.

Android Agents tests

The Android Agents plugin includes an example TestPlan that contains a sample of the configuration of the different agent's test steps. It also includes additional information in the form of dialog messages. The example TestPlan is located in the path \AdbAgents\Example.TapPlan of the OpenTAP installation folder.



The TestPlan is structured as a set of nested test steps, with each one configuring one of the different agents. The duration of the TestPlan is controlled by the last one (Adb Exoplayer), while the others are run for the duration of the child steps.

In order to be able to run the TestPlan, an Android device must be connected to the test machine via USB, the Agents must be installed in the device, and the corresponding Instruments must be configured

in OpenTAP. Also note that the Adb iPerf Agent step must be configured with the Host and Port values of a running iPerf 2 server in order to generate any results.

Conclusions

By reading this guide, the authors hope you are now able to create and run useful tests to validate your 5G KPIs. The authors welcome contribution or package expansion proposed via Gitlab. If you have any issue, please post your comments in the Gitlab repository.

Contributing Projects

- 5G-VINNI
- 5GENESIS

Contacts

WG Chair: Evangelos Kosmatos, WINGS ICT Solutions vkosmatos@wings-ict-solutions.eu

WG Co-chair: Michael Dieudonné, Keysight Technologies michael_dieudonne@keysight.com

5G PPP WGs: 5g-ppp.eu/5g-ppp-work-groups

Contributors

Name	Company/Institution/University	Country
Editors		
Andrea F. Cattoni	Keysight Technologies,	Denmark
Lars Nielsen	Keysight Technologies	Denmark
Contributors and Reviewers		
Almudena Diaz Zayas	University of Malaga	Spain
Bruno García García	University of Malaga	Spain
Lars Nielsen	Keysight Technologies	Denmark
Anastasius Gavras	Eurescom	Germany
Michael Dieudonne	Keysight Technologies	Belgium

References

[1] NGMN, Definition of the Testing Framework for the NGMN 5G Pre-Commercial Network Trials (Version 2), https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2019/190111_NGMN_PreCommTrials_Framework_definition_v2_small.pdf

[2] 3GPP, TS 28.552: Management and orchestration; 5G performance measurements, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3413

[3] ETSI Zero touch network and Service Management (ZSM) Industry Specification Group (ISG), https://www.etsi.org/technologies/zero-touch-network-service-management

[4] Open Platform for NFV (OPNFV), Linux Foundation Project, https://www.opnfv.org/

[5] Open Network Automation Platform (ONAP), Linux Foundation Project, https://www.onap.org/

[6] "Contractual Arrangement Setting up a Public Private Partnership in the Area of Advanced 5G Network Infrastructure for te Future Internet between the European Union and the 5G Infrastructure Association", December 17, 2013 (https://5g-ppp.eu/contract/ latest access 07/05/2019)

[7] 3GPP, TS 28.554: Telecommunication management; Management and orchestration; 5G end to end Key Performance Indicators (KPI), https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3415

[8] EU H2020 TRIANGLE Project, Deliverable D2.6: Final Test Scenario and Test Specifications, https://www.triangle-project.eu/wp-content/uploads/2018/11/TRIANGLE_D2-6.pdf

[9] OpenTAP - An Open Source Project for Test Automation https://www.opentap.io/

[10] 5GPPP TMV Basic Testing Bundle Gitlab Project https://gitlab.com/OpenTAP/Plugins/5g-ppp-test-measurement-and-kpis-validation-wg/5gppp-tmv-basic-testing-bundle