

H2020-ICT-688712



Project: H2020-ICT-688712

Project Name:

5G Applications and Devices Benchmarking (TRIANGLE)

Deliverable D3.4

## **Report on the implementation of testing framework Release 3 and specification of testing framework Release 4**

Date of delivery:	31/03/2018	Version:	1.0
Start date of Project:	01/01/2016	Duration:	36 months



## Deliverable D3.4

# Report on the implementation of testing framework Release 3 and specification of testing framework Release 4

<b>Project Number:</b>	ICT-688712
<b>Project Name:</b>	5G Applications and Devices Benchmarking
<b>Project Acronym</b>	TRIANGLE

<b>Document Number:</b>	ICT-688712-TRIANGLE/D3.4
<b>Document Title:</b>	Progress report on the testing framework Release 3 and specification of Release 4
<b>Lead beneficiary:</b>	Universidad de Málaga
<b>Editor(s):</b>	Almudena Díaz Zayas (Universidad de Málaga)
<b>Authors:</b>	Keysight Technologies Belgium (Michael Dieudonne), Keysight Technologies Denmark (Andrea Cattoni, German Corrales Madueño, Marek Rohr), Universidad de Malaga (Alberto Salmerón, Almudena Díaz, Pedro Merino, Cesar A. García, Laura Panizo Jaime, Bruno García, Guillermo Chica, Verónica Tapia, Maria del Mar Gallardo), Redzinc Services Limited (Jeanne Caffrey, Donal Morris, Ricardo Figueiredo, Terry O'Callaghan, Pilar Rodríguez), DEKRA Testing and Certification S.A.U (Carlos Cárdenas, Janie Baños, Oscar Castañeda, J.C. Mora), Quamotion (Frederik Carlier, Bart Saint Germain), TNO (Lucía D'Acunto, Piotr Zuraniewski, Niels van Adrichem
<b>Dissemination Level:</b>	PU
<b>Contractual Date of Delivery:</b>	31/03/2018
<b>Work Package Leader:</b>	Universidad de Málaga
<b>Status:</b>	Final
<b>Version:</b>	1.0
<b>File Name:</b>	TRIANGLE_Deliverable_D3.4_Final.docx

### Abstract

This deliverable provides the description of the third release of the TRIANGLE testbed. The most important feature of this release is the support of the execution of certification campaigns and the computation of the TRIANGLE mark.

### Keywords

Architecture, workflow, deployment, orchestration, test case, portal, measurements tools, RAN, EPC, UEs



## Executive summary

This document is the fourth deliverable of WP3. WP3 is responsible for the development of the testing framework in TRIANGLE. Testing framework covers all the software, coordination/sequencing that controls & connects the test infrastructure. It oversees handling and converting the end user test requests into actionable steps within the software and hardware portion of the testbed.

The document describes the testing framework of the third Release (Rel-3) of the testbed. The testbed architecture, introduced in D3.2, has been consolidated in this new Release. The new features are the natural evolution of the testing framework components identified in D3.2.

The main feature introduced in the Rel-3 of the testbed is the execution of certification campaigns, which involves also the computation of the TRIANGLE mark. The certification campaigns are based on the execution of the test cases specified in D2.2. A test case defines the conditions of the test, the measurements that to be collected and the app user flow which will be used to stimulate the feature under test. The current version of the testbed supports the execution of the test cases specified in three app testing domains: App User Experience (AUE), Device Resource Usage (DER) and Apps Energy Consumption (AEC); and in two domains of mobile device testing: Device Energy Consumption (DEC) and User Experience with Reference Apps (DRA). A complete list of the test cases currently supported is provided in Annex I of this document.



## Contents

1	Introduction .....	2
2	TRIANGLE testbed architecture.....	5
3	Interface and visualization (Portal).....	7
3.1	Test cases supported.....	7
3.2	Traffic capture .....	8
3.3	Allow upload of Powershell scripts as App User Flow .....	9
3.4	Enabling/disabling remote screen.....	11
3.5	Task ID.....	12
3.6	Certification campaign .....	12
3.7	TRIANGLE mark and spider diagram .....	14
4	Orchestration.....	16
4.1	Orcompositor .....	16
4.2	Test Automation platform (TAP) .....	16
4.3	New features in the Quamotion WebDriver .....	27
5	Measurements and data collection .....	28
5.1	KPIs computation.....	28
5.2	Metrics and mark computation.....	30
5.3	Instrumentation library .....	31
5.4	Measurement calculation without using the instrumentation library (UMA) .....	31
5.5	New features in DEKRA TACS Performance Tool (DEKRA).....	31
6	RAN (Radio Access Network).....	33
7	EPC.....	34
8	Transport.....	35
8.1	Cloud infrastructure description .....	35
8.2	MANO – Management and Network Orchestration .....	43
9	UE (User Equipment and accessories).....	51
9.1	Supported UEs.....	51
10	Local applications and Servers (TNO) .....	52
10.1	DANE Local applications and Servers (TNO) .....	52
10.2	Web Client .....	54
10.3	Metrics Database .....	55
10.4	Metrics Visualisation .....	56
10.5	Fake client swarm .....	56
10.6	Obtaining logs .....	57
10.7	Default port assignments .....	57



11	Extensions and new features .....	58
11.1	Booking system.....	58
11.2	Remote screen.....	60
11.3	Power shell support .....	62
11.4	iOS support .....	63
11.5	Model-based testing: Automatic App model extraction.....	63
12	Internal test experiment.....	69
13	TRIANGLE testbed Release 4 specifications.....	70
14	References.....	71
15	Annex 1: Test cases supported in Release 3.....	72
15.1	Apps User Experience (AUE) .....	72
15.2	Mobile Devices Energy Consumption (DEC).....	73
15.3	Apps Energy Consumption (AEC) .....	74
15.4	Mobile devices User Experience with Reference Apps Test Specification (DRA) .....	75
15.5	Applications Device Resources Usage (RES) .....	76
16	Annex 2: Measurements points (Instrumentation library).....	78
16.1	Common Services.....	78
16.2	Content Distribution Streaming Services .....	78
16.3	Live Streaming Services .....	81
16.4	Social Networking .....	82
16.5	High Speed Internet.....	85
16.6	Virtual Reality.....	86
16.7	Augmented Reality.....	87
16.8	Gaming .....	88
17	Annex 3: Robotic Arm Remote Control Interface .....	90
18	Annex 4: OpenStack API access .....	96
19	Annex 5: Sample work-flow.....	99
19.1	Bootstrapping environment.....	99
19.2	Preparing descriptors.....	100



## List of Figures

Figure 1 High-Level architecture of the testbed .....	2
Figure 2 Control and management entities of the testbed .....	6
Figure 3 Testbed workflow .....	6
Figure 4 The Portal identifies the available test cases and presents a description for each one .....	7
Figure 5 Measurement points required to compute the measurements specified in the test cases .....	8
Figure 6: 'Capture Traffic' selector .....	8
Figure 7: Display 'Capture Traffic' selection .....	9
Figure 8: Allow uploading PowerShell files (.ps1) .....	9
Figure 9: Display file content instead of a table with a parsed JSON file in Campaign .....	10
Figure 10: Display file content instead of a table with a parsed JSON file in App Test Case ..	10
Figure 11: Disabled 'View screen feed' in Standard Campaign .....	11
Figure 12: Enabled 'View screen feed' in Custom Campaign .....	11
Figure 13: Show 'Execution Task ID' in Campaign Execution.....	12
Figure 14: Show 'Certification' type in Campaign creation .....	12
Figure 15: Show Campaign type in Campaigns view .....	13
Figure 16: Show Campaign type in each Campaign and remove Scenario and Control Traffic if type 'Certification' .....	13
Figure 17: Show Campaign type in each Campaign and show Control Traffic if type 'Custom' .....	14
Figure 18 TRIANGLE mark .....	14
Figure 19 MANO instrument setup.....	20
Figure 20 Putty key conversion .....	21
Figure 21 Execute remote command .....	24
Figure 22 Execute remote command known host .....	25
Figure 23 Execute remote command to Host with known host .....	26
Figure 24 MANO sample test plan .....	27
Figure 25: Configuration parameters on the Test Case Labeller step.....	29
Figure 26: Configuration parameters on the RES step. All steps include the same basic settings. ....	30
Figure 27: Additional settings on the AUE step.....	30
Figure 28 Snippet for generating measurement points with the same format that the instrumentation library .....	31
Figure 29: Architecture overview.....	36
Figure 30: Hypervisor configuration through virt-manager .....	37



Figure 31: Overview of nodes deployed through MAAS, cloud-related nodes are initiated and controlled through Juju. ....	37
Figure 32: Network overview of networks managed by lib-virt.....	39
Figure 33: Juju model describing functional elements and their connections .....	41
Figure 34 OSM mapping to ETSI NFV MANO .....	44
Figure 35: State of an OSM VM after OSM installation.....	45
Figure 36: OSM GUI sample screenshot .....	48
Figure 37: Instantiation.....	49
Figure 38: Logging and debug via OSM GUI .....	49
Figure 39: Logging and debug via OSM GUI .....	50
Figure 40 The New Reservation Page allows a user to select a slot time to use the Portal Testbed .....	59
Figure 41 Calendar interface. It shows the reservations made. Clicking on any day will take you to the New Reservation Page.....	59
Figure 42 View of the screenshot feed feature provided by Quamotion. ....	61
Figure 43 Model extraction overview.....	64
Figure 44. Universal Music Player model.....	66
Figure 45: Image creation (part 1).....	100
Figure 46: Image creation (part 2).....	101
Figure 47: VNFD creation (adding VNDF 1).....	101
Figure 48: VNFD creation (name and ID).....	102
Figure 49: VNFD creation (connectivity) .....	102
Figure 50: VNFD creation (VDU details) .....	102
Figure 51: VNFD creation (VNF connectivity) .....	103



## List of Tables

Table 1 List of TRIANGLE features.....	3
Table 2 MANO instrument setup.....	19
Table 3 Test step MANO setup.....	21
Table 4 Test step Contact MANO .....	22
Table 5 Test step Set up Network Service Record .....	22
Table 6 Set up Resolve hostnames .....	22
Table 7 Test step Execute remote command.....	23
Table 8 Execute remote command known host .....	24
Table 9 Execute remote command to Host with known host .....	25
Table 10: admin-openrc5.sh.....	42
Table 11: LXD containers after OSM installation .....	45
Table 12: OSM Client installation, configuration and verification .....	46
Table 13: OSM Client modifications .....	46
Table 14: Using OSM client with GUI generated descriptor may cause an error .....	47
Table 15 Current status of devices integrated into the testbed .....	51
Table 16 Messages from MPEG-SAND protocol .....	52
Table 17 List of PowerShell scripts used in the Quamotion Docker deployment .....	62
Table 18 List of PowerShell scripts used in the Quamotion Docker deployment .....	63
Table 19 Model extraction - Configuration .....	67
Table 20. Model extraction - Results.....	68
Table 21: Identity Service API Request Token.....	96
Table 22: Identity API Service Authentication Token Response .....	96
Table 23: Sample client VNFD .....	103





## List of Abbreviations

<b>AUT</b>	App Under Test
<b>AP</b>	Access Point
<b>APNet</b>	Antennas, Propagation and Radio Networking
<b>BER</b>	Bit Error Rate
<b>BLER</b>	Block Error Rate
<b>BS</b>	Base Station
<b>CAPEX</b>	CApital EXpenditure
<b>CDMA</b>	Code Division Multiple Access
<b>CFO</b>	Carrier Frequency Offset
<b>CO</b>	Confidential
<b>CP</b>	Cyclic Prefix
<b>CR</b>	Cognitive Radio
<b>CRS</b>	Cognitive Radio Systems
<b>CSI</b>	Channel State Information
<b>CSMA</b>	Carrier Sense Multiple Access
<b>C2X</b>	Car-to-Anything
<b>D</b>	Deliverables
<b>DL</b>	Downlink
<b>D2D</b>	Device-to-Device
<b>DMRS</b>	Demodulation reference signal
<b>DRX</b>	Discontinuous Reception
<b>DTX</b>	Discontinuous Transmission
<b>DUT</b>	Device Under Test
<b>EIRP</b>	Effective Isotropic Radiated Power
<b>EIT</b>	European Institute for Innovation and Technology
<b>E2E</b>	End-to-End
<b>EVM</b>	Error Vector Magnitude
<b>FDD</b>	Frequency Division Duplex
<b>FD-MIMO</b>	Full-Dimension MIMO
<b>FEC</b>	Forward Error Correction
<b>FR</b>	Frequency Response
<b>GPRS</b>	General Packet Radio Service
<b>GSM</b>	Global System for Mobile communications

<b>HARQ</b>	Hybrid Automatic Repeat Request
<b>ICI</b>	Inter-Carrier Interference
<b>ICT</b>	Information and Communications Technology
<b>IEEE</b>	Institute of Electrical and Electronics Engineers
<b>IMT</b>	International Mobile Communications
<b>IP</b>	Intellectual Property
<b>IPR</b>	Intellectual Property Rights
<b>IR</b>	Internal report
<b>ITU</b>	International Telecommunication Union
<b>ITU-R</b>	International Telecommunication Union-Radio
<b>KPI</b>	Key Performance Indicator
<b>LAN</b>	Local Area Network
<b>LOS</b>	Line of Sight
<b>LTE</b>	Long Term Evolution
<b>LTE-A</b>	Long Term Evolution-Advanced
<b>L2S</b>	Link to System
<b>M</b>	Milestones
<b>Mbps</b>	megabits per second
<b>Mo</b>	Month
<b>MA</b>	Multiple Access
<b>MAC</b>	Medium-access Control
<b>MGT</b>	Management
<b>MIMO</b>	Multiple-Input Multiple-Output
<b>MMC</b>	Massive Machine Communication
<b>M2M</b>	Machine to Machine
<b>MSE</b>	Mean Squared Error
<b>NLOS</b>	Non line of Sight
<b>OFDM</b>	Orthogonal Frequency Division Multiplexing
<b>OPEX</b>	Operational Expenditure
<b>PA</b>	Power Amplifier
<b>PAPR</b>	Peak-to-Average-Power-Ratio
<b>PC</b>	Project Coordinator



<b>PHY</b>	Physical Layer
<b>PU</b>	Public
<b>QAM</b>	Quadrature Amplitude Modulation
<b>QAP</b>	Quality Assurance Plan
<b>QMR</b>	Quarterly Management reports
<b>QoE</b>	quality of experience
<b>QoS</b>	Quality of Service
<b>RACH</b>	Random Access Channel
<b>RAN</b>	Radio Access Network
<b>RAT</b>	Radio Access Technology
<b>RF</b>	Radio Frequency
<b>R&amp;D</b>	Research and Development
<b>RRM</b>	Radio Resource Management
<b>RTD</b>	Research and Technological Development
<b>RTT</b>	Round Trip Time
<b>SDR</b>	Software Defined Radio
<b>SINR</b>	Signal to Interference and Noise Ratio

<b>SRS</b>	Sounding Reference Signal
<b>T</b>	Task
<b>TDD</b>	Time Division Duplex
<b>TDMA</b>	Time Division Multiple Access
<b>TRX</b>	Transmitter
<b>TTI</b>	Transmission Time Interval
<b>UE</b>	User Equipment
<b>UL</b>	Uplink
<b>UMTS</b>	Universal Mobile Telecommunications System
<b>USRP</b>	Universal Software Radio Peripheral
<b>V2V</b>	Vehicle-to-Vehicle
<b>V2X</b>	Vehicle-to-everything
<b>WCDMA</b>	Wide Code Division Multiple Access
<b>WLAN</b>	Wireless Local Area Network
<b>WP</b>	Work Package
<b>WPAN</b>	Wireless Personal Area Networks

# 1 Introduction

The general architecture of the testbed introduced in D3.1 has been consolidated in Release 3.

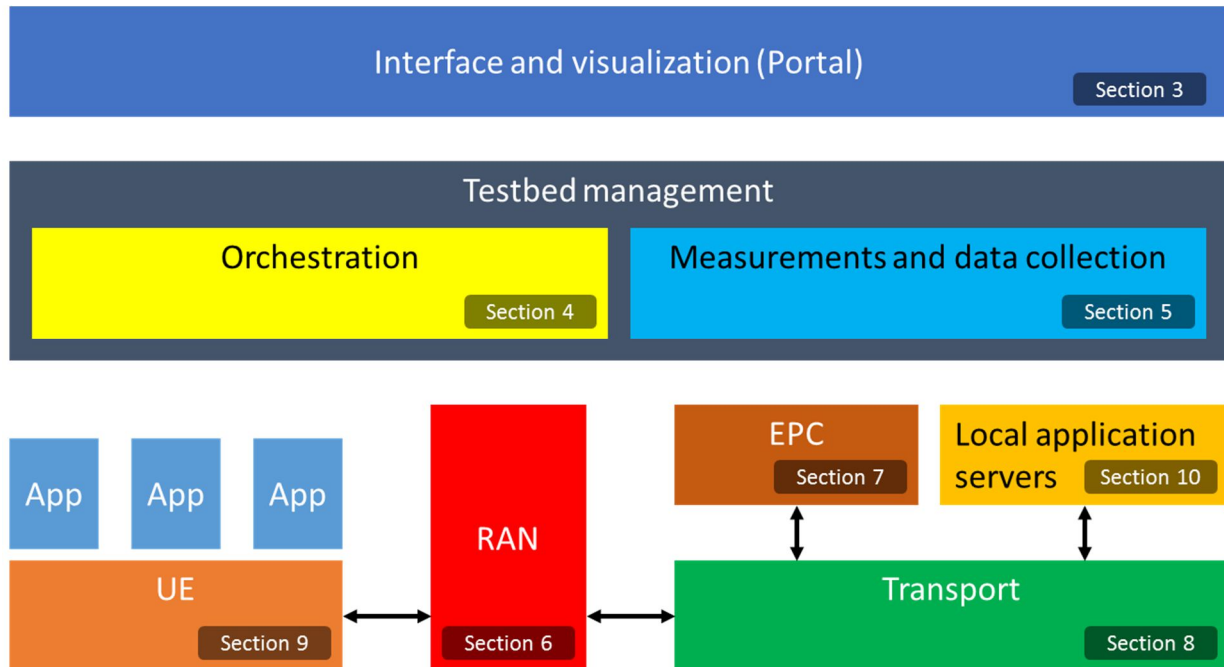


Figure 1 High-Level architecture of the testbed

TRIANGLE project is devoted to the testing and benchmarking of mobile applications and devices. The testbed provides a high-level access based on a Web Portal which offers an intuitive interface for the definition and execution of the testing campaigns. For advanced users interested on the configuration of the low-level parameters of the testing scenarios the testbed offers an additional access based on TAP (Testing Automation Platform), a programmable sequencer of actions whose plugins expose the configuration and control of the instruments and tools integrated into the testbed.

WP2 has identified a set of uses cases, which is used to classify the different apps. This classification is required to test apps and devices programmatically. In each use case, WP2 has identified the most relevant and common features provided by apps. In each feature, WP2 has specified a test case where we measure the performance of such feature. This approach provides a common framework for the certification of applications. The testbed enables also the definition of custom campaigns to test additional features offered by the applications. More specifically, through Release 3 of the Portal, it is possible to define three different testing campaigns:

- **Certification campaigns.** A certification campaign executes all the test cases that apply to the uses cases and the features supported by the app. The uses cases and the features are configured when the application is uploaded to the Portal. No additional configuration is needed.
- **Standard campaigns.** When users opt for standard campaigns they can select between the test cases defined for the uses cases and features supported by app. The user can also configure the scenario and the device used during the



campaign. Remote screen and traffic capture are available during the execution of these campaigns.

- **Custom.** Users can define their own test campaign providing a power shell script which replays the feature under test and can select the scenario and the device. Remote screen and traffic capture are available during the execution of these campaigns.

The most relevant features included in Release 3 of the testbed are the following:

- **KPIs computation:** Raw measurements collected by the software tools and instruments integrated into the testbed are post-processed to compute the KPIs specified in D2.2. The computation of these KPIs has been implemented as test steps in TAP. These test steps are called by the orchestrator of the testbed once the test case has been executed and the measurements are available.
- **Triangle mark computation.** The ETL module maps the KPIs computed into MOS values. One MOS value is provided per domain and the aggregation of all of them is used to obtain the TRIANGLE mark.
- **Certification campaigns.** The user selects per use case the different features supported by the application. Using this information together with the rules standardized in D2.2, the orchestrator of the testbed selects the test cases that the application must pass to obtain the **TRIANGLE mark**.
- **Remote screen.** The possibility of visualizing what is happening in the device during the execution of the tests is a common request from OC1 and OC2 open callers. This feature is now available through the Portal for custom and standards campaigns.
- **Powershell support for the definitions of the app user flows.** This new approach provides greater level of flexibility for app user flows.
- **Additional security.** The test runner (app user flow execution) and the Quamotion WebDriver server are now deployed in Docker containers isolated to avoid malicious attacks to the integrity of the testbed. This was a very important achievement in order to support Powershell.

For a better understanding of the testbed evolution, we summarized the main features in the following table:

Table 1 List of TRIANGLE features

Testbed features	Components	Rel	Comments
Testbed accessibility	Portal	R2	Main entry point of the testbed
	Web Reporting Tool	R2	Raw measurements visualizer
	Booking System	R3	
	Test Automation Platform (TAP)	R2	Testbed access for advanced users
Testbed usage	Certification campaigns	R3	
	Standard campaigns	R2	



Results	Custom campaigns	R2	
	Researcher campaigns	R2	Modification of the templates and configuration of low level parameters
	Raw results	R2	
	KPIs	R3	
	MOS	R3	
Device automation	Triangle mark	R3	
	Quamotion Webdriver	R1	
	Powershell support	R3	App users flows are defined via powershell scripts
Monitoring tools	TestelDroid	R1	Android applications which enables traffic capture at the UE
	DEKRA Performance Tool	R1	Multiplatform performance monitoring tool: device resources (RAM, CPU, GPU), traffic statistics, reference applications
	Instrumentation library	R2	Defines a set of measurements points to correlate the measurements collected with the actions performs by the applications under test
Measurements post-processing and storage	ETL <i>Framework</i>	R3	MOS scoring per domain, aggregated MOS
	KPIs computation	R2	KPIs specified in it D2.2 are computed after the execution of each test case and stored in the general database
	General database	R2	
Backhaul	Emulated Impairments	R3	Core interfaces and servers connections
	SDN	R3	Servers connective
	VNF	R3	Servers deployment
	Openstack	R3	Servers deployment
	Commercial EPC (Polaris)	R2	
<i>Instruments</i>	UXM	R1	LTE-A Pro base station emulator
	N6705B Power Analyzer	R1	
<i>Devices</i>	Android devices	R1	
	NB-IoT devices	R2	
	iOS devices	R3	
<i>Additional features</i>	S1 Interface	R1	eNB Emulator is capable of connecting to commercial core networks
	S1 handover	R2	
	Robotic arm	R2	
	GPS emulation	R2	



Model based testing	R2	
Remote pcap	R2	Enables the capture of traffic in any of the interfaces.
Heterogeneous access	R3	
Remote screen	R3	VNC access to the app and device under test

## 2 TRIANGLE testbed architecture

The control and management components identified and implemented in Release 2 (see Figure 2) and the workflow of the testbed (see Figure 3) have not been modified. The new features of the testbed have been introduced in the components identified in the existing architecture. These new features are described in the following sections.

As a reminder, the main testbed components are briefly described below.

**Portal:** High level end users access the TRIANGLE Testbed through the Portal. In this Portal they can upload new apps (if they are app developers), or declare the devices they have sent to the Testbed (if they are device makers). In addition, they will have to declare the features or capabilities of their apps or devices (implementation statements). These features will define what can be tested through experiments, or which tests specifications will be applicable when they opt for certification. In addition to the implementation statements, end users will have to provide additional information.

Users can then define their own campaigns to test certain features of their app or device. For experimentation campaigns, users can configure have some high-level options: the test scenario, the device on which the test will be carried out (for app developers), the reference app to test (for device makers), and a subset of the applicable KPIs. Most of the information required to execute a test, such as how to measure certain KPIs, should have been provided earlier.

**Orcomposutor:** Once all the required information has been entered in the Portal, the user can run experiments or certifications. In both cases the workflow is similar. The first step would be to take the information entered and turn it into executable TAP test plans. This is the task of the test plan Orcomposutor. The Orcomposutor is also aware of which KPIs are going to be measured with each of the generated TAP test plans.

Depending on the number of selected options, or the test specifications that the product must pass for certification, the Orcomposutor will generate the applicable test plans. For instance, for certification, the test specifications will include testing the same features/KPIs under different network scenarios. For experimentation, the number of test plans will depend on the subset of features/KPIs and other options selected by the user. To create the required TAP test plans, the Orcomposutor uses pre-defined test plan templates. For an app test, the body of the test plan typically includes replaying the user actions contained in an app user flow provided by the app developer.

**ETL framework:** A specialized ETL (Extract, Transform, Load) will perform the mapping between KPIs and MOS values and will general the final TRIANGLE mark for certification campaigns. A MOS value is computed per domain. A final MOS aggregation is done to provide the TRIANGLE mark.

The computed KPIs are stored in the general database of the testbed. The ETL framework will use these values.

The computed KPIs and MOS values will be available for the user in the Portal, along the raw measurements.

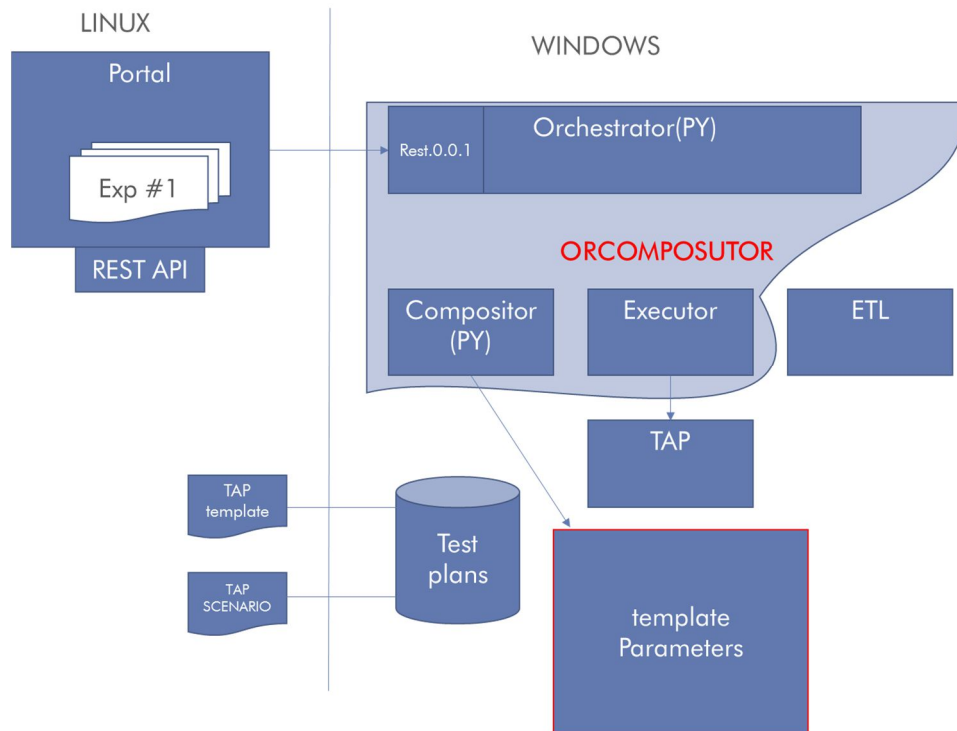


Figure 2 Control and management entities of the testbed

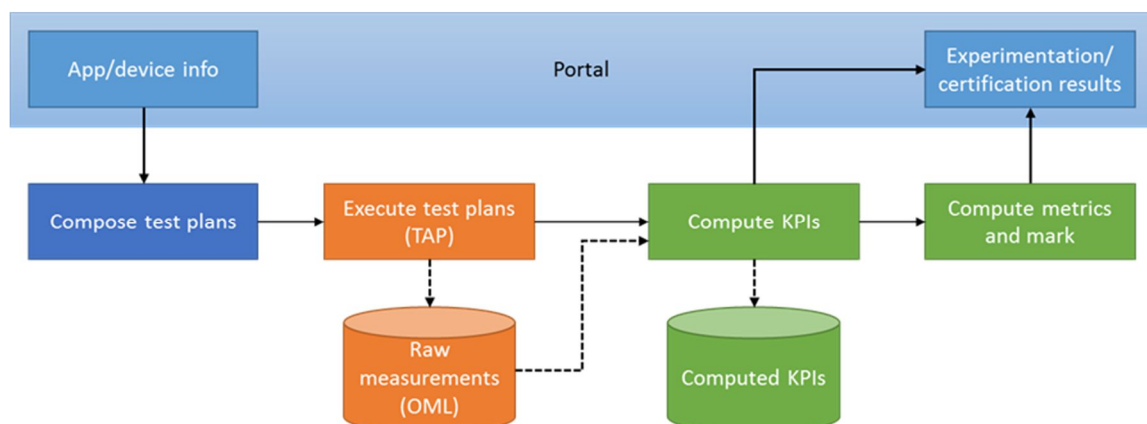


Figure 3 Testbed workflow










### 3 Interface and visualization (Portal)

#### 3.1 Test cases supported

The Portal offers full support for the test cases defined in D2.2 for the App User Experience (AUE) domain, the Device Resources Usage (RES) domain and the App Energy Consumption (AEC) domain.

Features	Test Cases
<ul style="list-style-type: none"><li>✓ <b>Download Media Content For Offline Playing</b> Download Media Content For Offline Playing</li><li>✓ <b>Media File Playback</b> Media File Playback</li><li>✓ <b>Play And Pause</b> Play And Pause</li><li>✓ <b>Playlist Skip Forward and Backward</b> Playlist Skip Forward and Backward</li><li>✓ <b>Rewind and Fast Forward</b> Rewind and Fast Forward</li><li>✓ <b>Search And Seek</b> Search And Seek</li><li>✓ <b>Stop And Replay</b> Stop And Replay</li></ul>	<ul style="list-style-type: none"><li> <b>Download Content for Offline Playing</b> Measure the ability of the AUT to download a media file for offline playing. Estimated duration: 5 minutes <a href="#">Download Media Content For Offline Playing</a></li><li> <b>Fast Forward</b> Measure the ability of the AUT to perform rewind and fast forward operations while playing a media file. Estimated duration: 5 minutes <a href="#">Rewind and Fast Forward</a></li><li> <b>Non Interactive Playback</b> Measure the user experience KPIs by the AUT while executing the feature media file playing from the Content Distribution Streaming Services use case. Estimated duration: 5 minutes <a href="#">Media File Playback</a></li><li> <b>Non Interactive Playback with screen off</b> Measure the usage of device resources of the AUT while executing the feature media file playing from the Content Distribution Streaming Services use case and the screen is turned off. Estimated duration: 5 minutes <a href="#">Media File Playback</a></li><li> <b>Play and Pause</b> Measure the ability of the AUT to pause and the resume a media file.</li></ul>

**Figure 4** The Portal identifies the available test cases and presents a description for each one

As shown in Figure 5, the Portal also provides a list of the measurements points that must be included in the source code of the application under test to obtain the measurements specified in the test cases.





<b>Media File Playback - Content Stall Start</b> Media File Playback - Content Stall Start	Non Interactive Playback	eu.triangle_project.appinstr.cs.MediaFilePlayback.mediaFilePlaybackContentStallStart()
<b>Media File Playback - Content Stall End</b> Media File Playback - Content Stall End	Non Interactive Playback	eu.triangle_project.appinstr.cs.MediaFilePlayback.mediaFilePlaybackContentStallEnd()
<b>Media File Playback - Pause</b> Media File Playback - Pause	Play and Pause	eu.triangle_project.appinstr.cs.PlayAndPause.mediaFilePlaybackPause()
<b>Media File Playback - Resume</b> Media File Playback - Resume	Play and Pause	eu.triangle_project.appinstr.cs.PlayAndPause.mediaFilePlaybackResume()

Figure 5 Measurement points required to compute the measurements specified in the test cases

## 3.2 Traffic capture

Traffic capture can be computational intensive and can interfere with the measurements. This feature is configurable through the added field 'Capture Traffic' in the "Create Campaign" form.

When the user creates a custom campaign or standard campaign, the user can enable or disable the traffic capture. For the certification campaigns this option is not available.

Welcome to Triangle Web Portal | John Doe

### Create campaign

Home / Campaigns / Create campaign

**Create new campaign**

Name  
Enter campaign name

App version  
SkyTube Extra | 2.81 Extra

Device  
Samsung Galaxy S4

Scenario  
High Speed - Direct Passenger Connection

Campaign type  
☐ Standard - Execute test cases configured in your app  
☒ Custom - Run your own app user flow

Capture traffic  
☐ Yes  
☒ No

Cancel | Next

Figure 6: 'Capture Traffic' selector

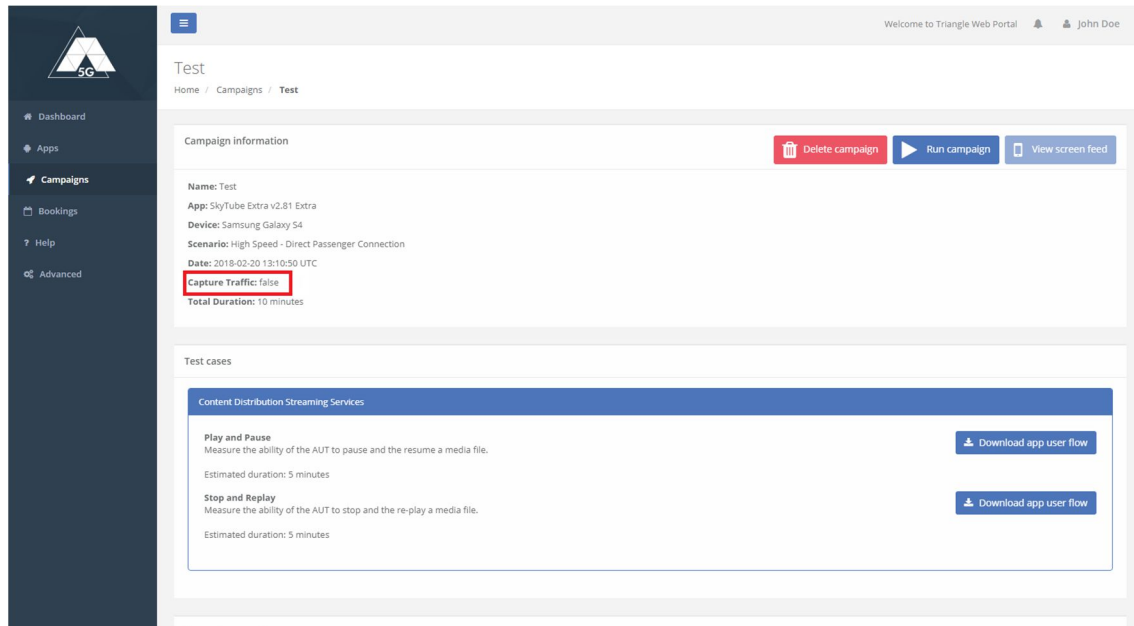


Figure 7: Display 'Capture Traffic' selection

### 3.3 Allow upload of Powershell scripts as App User Flow

Allow PowerShell files (.ps1) to be uploaded as App User Flow. In the previous version only JSON files were supported.

Some views has been modified to show the file content instead of just a table with the parsed JSON file.

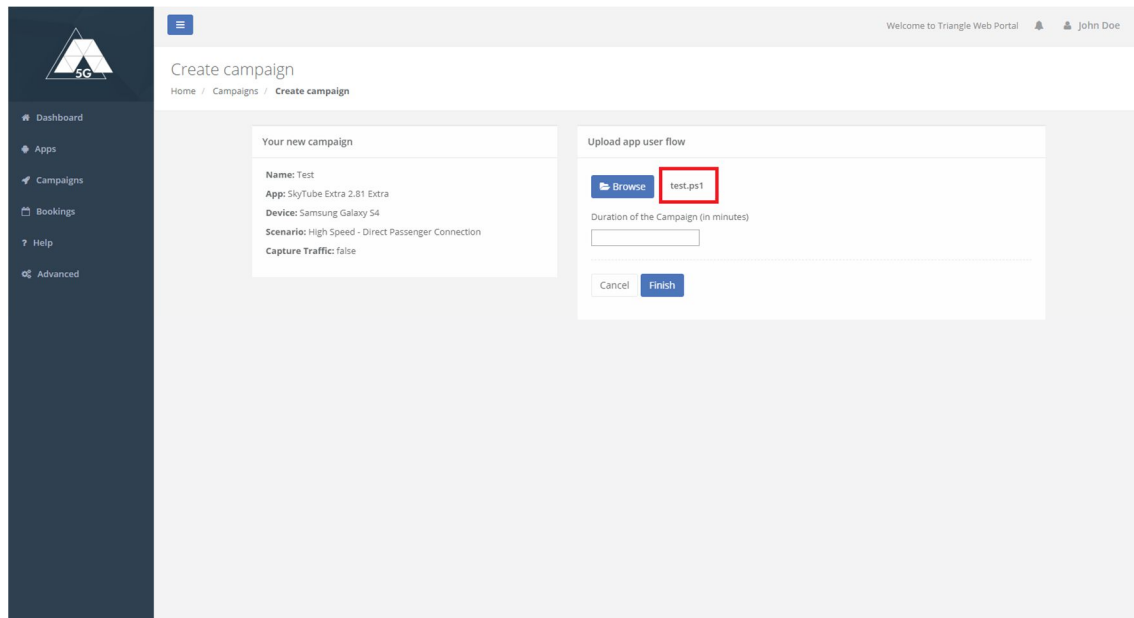


Figure 8: Allow uploading PowerShell files (.ps1)



**Figure 9: Display file content instead of a table with a parsed JSON file in Campaign**

**Figure 10: Display file content instead of a table with a parsed JSON file in App Test Case**



### 3.4 Enabling/disabling remote screen

Remote screen is one of the new features introduced in Release 3. The Portal enables the activation/deactivation of this new feature for standard and customs campaigns.

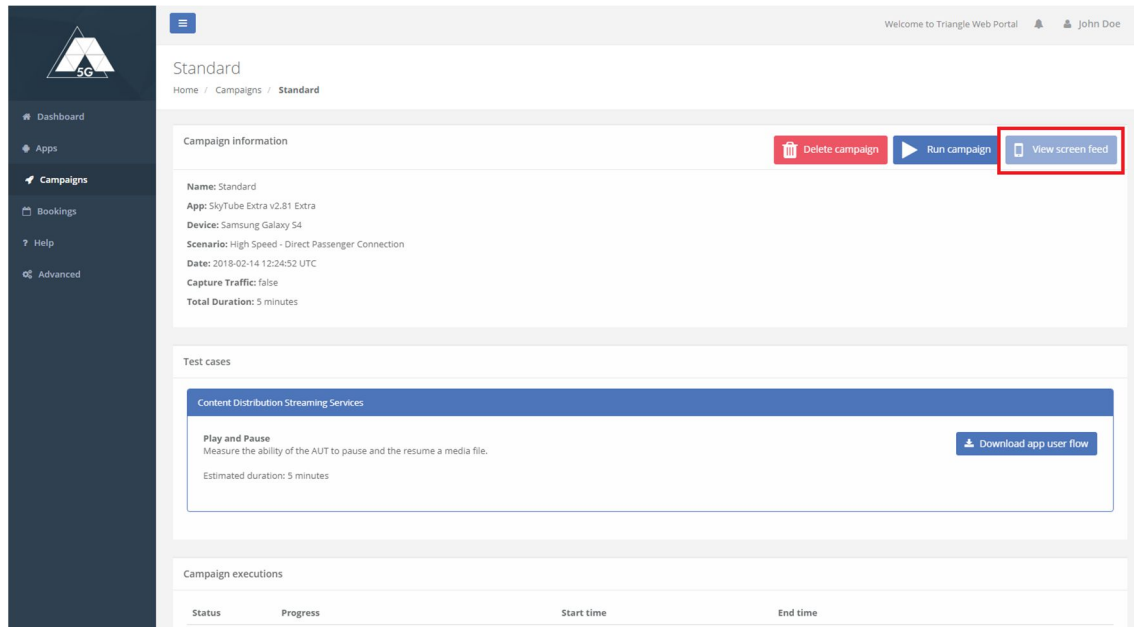


Figure 11: Disabled 'View screen feed' in Standard Campaign

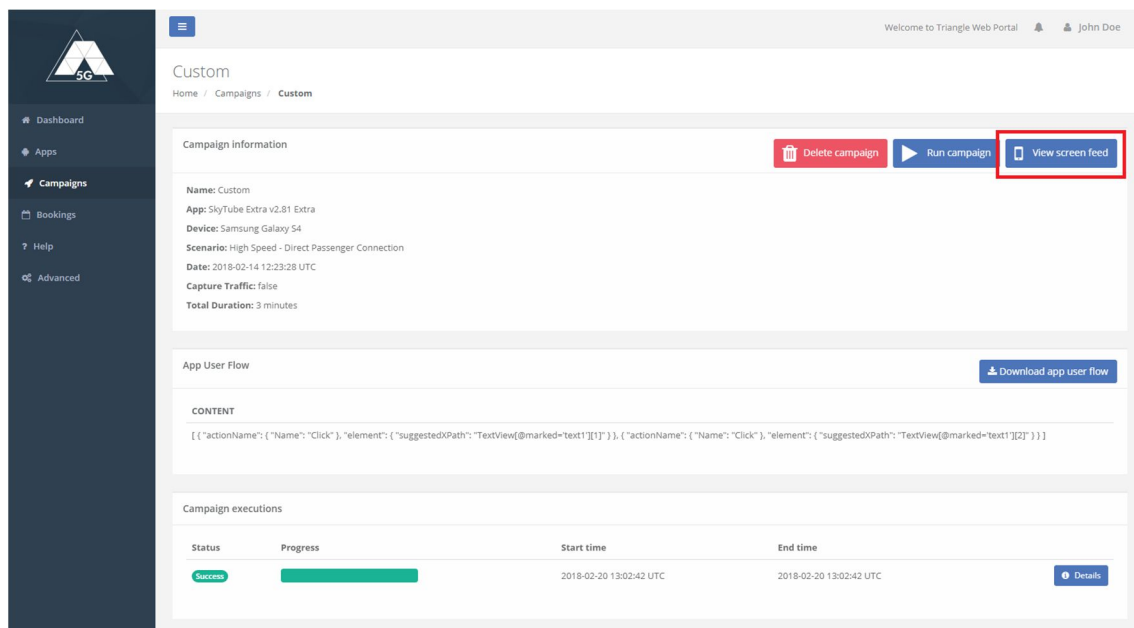


Figure 12: Enabled 'View screen feed' in Custom Campaign



### 3.5 Task ID

Show 'Execution Task ID' when displaying the Campaign Execution. This is the same ID used in the OML visualizer to identify the graphs corresponding to the execution.

The screenshot shows the 'Campaign Execution CE-3-4' page. It includes a sidebar with navigation links (Dashboard, Apps, Campaigns, Bookings, Help, Advanced) and a main content area. The main content area has a 'Test' header and a 'Campaign execution information' section. Below this is a table of 'Campaign execution tasks' with columns: Status, Test Case, Scenario, Start Time, End Time, and Task ID. The 'Task ID' column is highlighted with a red box.

Status	Test Case	Scenario	Start Time	End Time	Task ID
Success	Play and Pause	High Speed - Direct Passenger Connection	2018-02-20 13:10:55 UTC	2018-02-20 13:10:55 UTC	4
Success	Stop and Replay	High Speed - Direct Passenger Connection	2018-02-20 13:10:55 UTC	2018-02-20 13:10:55 UTC	5

Figure 13: Show 'Execution Task ID' in Campaign Execution

### 3.6 Certification campaign

The type 'Certification' has been added in the "Create campaign" form. All the views have been updated to show the information of this new type of campaign.

The screenshot shows the 'Create campaign' form. It includes a sidebar with navigation links (Dashboard, Apps, Campaigns, Bookings, Help, Advanced) and a main content area. The main content area has a 'Create new campaign' form with the following fields:

- Name: Enter campaign name
- App version: SkyTube Extra (dropdown)
- Device: Samsung Galaxy S4 (dropdown)
- Campaign type: ☒ Standard - Execute test cases configured in your app, ☒ Certification - Execute test cases in all applicable scenarios, ☐ Custom - Run your own app user flow
- Buttons: Cancel, Next

Figure 14: Show 'Certification' type in Campaign creation



Type	Name	Date	App	Device	Scenario	Estimated Time
Certification	Certification	2018-03-05 12:35:08 UTC	SkyTube Extra v2.81 Extra	Samsung Galaxy S4	High Speed - Direct Passenger Connection	370 minutes
Standard	Test	2018-02-20 13:10:50 UTC	SkyTube Extra v2.81 Extra	Samsung Galaxy S4	High Speed - Direct Passenger Connection	10 minutes
Standard	Standard	2018-02-14 12:24:52 UTC	SkyTube Extra v2.81 Extra	Samsung Galaxy S4	High Speed - Direct Passenger Connection	5 minutes
Custom	Custom	2018-02-14 12:23:28 UTC	SkyTube Extra v2.81 Extra	Samsung Galaxy S4	High Speed - Direct Passenger Connection	3 minutes

Figure 15: Show Campaign type in Campaigns view

Type	Name	Date	App	Device	Scenario	Estimated Time
Certification	Certification	2018-03-05 12:35:08 UTC	SkyTube Extra v2.81 Extra	Samsung Galaxy S4	High Speed - Direct Passenger Connection	370 minutes

Figure 16: Show Campaign type in each Campaign and remove Scenario and Control Traffic if type 'Certification'

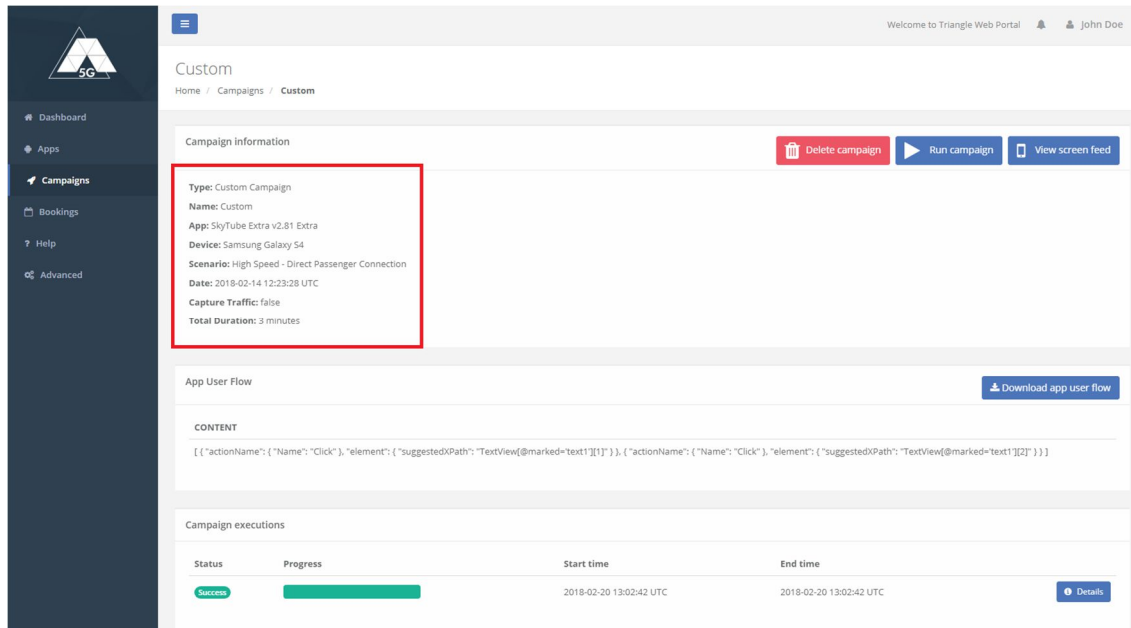


Figure 17: Show Campaign type in each Campaign and show Control Traffic if type 'Custom'

### 3.7 TRIANGLE mark and spider diagram

The Portal displays the Triangle Mark and a spider diagram with the score obtained in each one of the domains currently supported: App User Experience, App Device Resources Usage and App Energy Consumption.

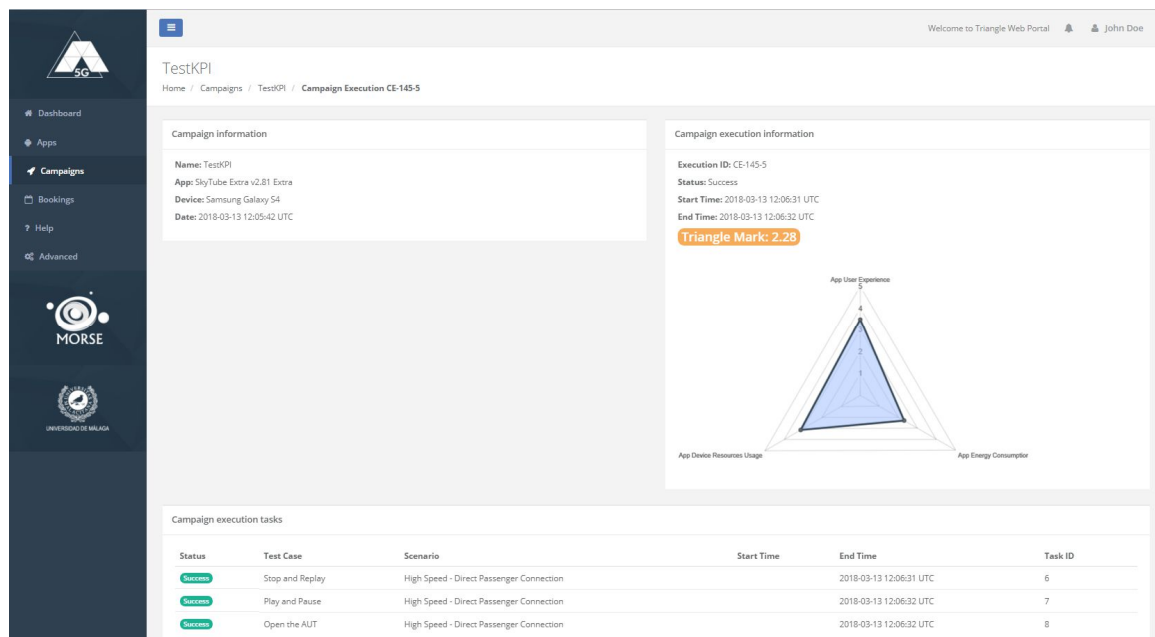


Figure 18 TRIANGLE mark



**Document:** ICT-688712-TRIANGLE/D3.4

**Date:** 29/03/2018

**Dissemination:** PU

**Status:** Final

**Version:** 1.0

---





## 4 Orchestration

### 4.1 Orcomposutor

Many improvements and extensions have been implemented in the Orcomposutor, in order to give support for new the features of the TRIANGLE testbed, to improve the performance of the workflow and to facilitate the error handling and debugging, both for the experimenters and the maintainers of the system. The updates implemented include:

- **Support for dynamic scenarios and the new TAP master template:** The new template requires a different set of external parameters. For example, to give support for dynamic scenarios it is necessary to use several testplan references: one for configuring the initial network conditions and up to five testplans that are executed randomly simulating the changing conditions of the network.

The new template also delegates the configuration of the device under test (for example, it is possible to use the same master template for Android and iOS devices) and other instruments to specific testplan references. The information about which testplans are required depending on the scenario and test conditions have been moved to several files in yaml format, which allow easier customization without the need to edit the source code of the Orcomposutor.

- **Multiple executions for all available domains:** The Orcomposutor is now able to execute each test case multiple times with different configurations for all the available domains. The results generated by each of these executions are uploaded to the TRIANGLE Portal organized in different folders inside the generated zip files.
- **KPI extraction:** The Orcomposutor is now able to automatically execute the corresponding KPI extraction steps for any given domain. These KPIs are used during the TRIANGLE Mark calculation, and are uploaded with the other generated measurements to the Portal.
- **TRIANGLE Mark calculation:** Once the campaign has finished and all the KPIs have been generated the Orcomposutor will initiate the ETL processing, obtaining the TRIANGLE mark and all the intermediate results of the test. These results will be uploaded to the Portal.
- **TAP log upload:** To facilitate the debugging of failed campaign executions the logs generated by TAP are now uploaded to the Portal alongside the results. Using these logs the experimenters can have a better understanding of the issues encountered during the execution.

### 4.2 Test Automation platform (TAP)

#### 4.2.1 TAP8 migration. New features in TAP8

TAP8, beyond enhanced usability and stability, comes with new features and capabilities which were developed to further improve the TRIANGLE testbed.

Examples of such new features are:



- References to Test plans can now be used as external parameters, as well as parameters inside the referenced test plans
- Plugins are now stored in custom folders in the TAP installation directory, defined by the plugin developer, to reduce file conflicts between plugins and facilitate installation and uninstallation of plugins.
- Referenced test plans can now have relative paths compared to the master test plan, for easier porting from one test machine to another.
- REST API for remote controlling of a TAP execution.

#### 4.2.2 New TAP test plan master template

For the first time, the testbed fully embraces TAP8 and the new features it includes. Naturally, the test plan master template has been overhauled with the new capabilities to further enhance the testbed flexibility and increase capabilities.

In summary, the new master template includes:

- Setting of test plan reference as external parameters, to be called via command line
- All parameters of these test plan reference are set as external parameters within these test plans, so that they can be set again as external parameters in the master template
- Simultaneous capture of
  - o Cellular statistics and device resource usage using the TACS4 agent and DEKRA TAP plugin, no matter the application flow.
  - o Testdroid logs (delivers PCAP files useful for traffic analysis).
  - o Android Logcat (provides Android debug logs useful for all application domains).
  - o Power consumption of the DUT via USB as well as fake battery connector.

The overall master template follows the high-level structure below:

1. Test case labeler
2. Instruments configuration
  1. UXM configuration
  2. Power analyzer configuration
  3. Other instruments configuration
3. Connect the DUT
4. Configure the DUT
5. Initialization of the TACS4 agent
  1. Configure the device automation profile to idle
6. Test body (repeat over X iterations)
  1. Test execution (parallel execution)
    - A. Application flow
      1. Start of non-blocking measurements (Testdroid, WebDriver)
      2. Execution of blocking measurements (parallel)
        - A. TACS4 test run
        - B. App flow run (Quamotion flow execution)
      3. Stop of non-blocking measurements
    - B. Network scenarios
      1. Subscenario 1
      2. Subscenario 2
      3. Subscenario N
  2. Retrieve data per iteration



3. Connectivity clean-up
7. Retrieve data per test case

**Notes:**

- The bullets marked 1. or 2. are executed successively, whereas bullets A. and B. are running simultaneously (in parallel).
- Most of the bullets above are actually references to other test plans, meaning that the template keeps its generic structure even if the nature of the DUT or test case changes
- The measurements are split into blocking (which will keep the test flow on hold until they are completed) and non-blocking (which are started in a single step and run in the background). For this reason, the blocking measurements are all combined in a parallel loop, to avoid blocking the rest of the test case logic.
- The center-core test step which dictates the actual duration of the test case still is the Quamotion application flow (replay of a JSON file through the Quamotion plugin). All other measurements or network scenario emulation are started before the app flow replay starts, and are stopped right after it stops, as only measurements when the app flow is running will be kept for post-processing.

### **4.2.3 TAP plugins**

#### **4.2.4 New version of DEKRA tool TAP plugin**

From Release 2 onwards DEKRA has provided several updates of the DEKRA tool TAP plugin. The plugin change log can be summarized with the following highlights:

- Support for TAP 8
- Contest Stall measurements (see section 5.5)
- GPU Usage measurements (see section 5.5)
- Robotic arm integration
- Minor bugs and other maintenance updates

The major upgrade has been the implementation of the TAP plugin for the integration of the robotic arm platform [D3.2], which was implemented in the Release 2 timeframe, into the Release 3 of the TRIANGLE testbed.

The robotic arm platform was implemented to meet the technical requirements derived from the Virtual Reality, Gaming and Augmented Reality test specification, mostly on the ability to move the device.

The implementation of the integration has consisted in two phases:

1. Development of a Remote-Control Interface service to expose all the capabilities of the robotic arm platform.
2. Development of the TAP plugin as user of the Remote Control (RC) interface developed in phase 1.



Annex 3 provides more details on the specification of the Remote-Control Interface service. That specification provides a complete view of the measurement capabilities available in Release 3 of the TRIANGLE testbed.

## 4.2.5 MANO plugin

In this Section we describe in detail the Management and Network Orchestration (MANO) TAP plugin. First, the instrument setting will be described, followed by the test steps, in the order they typically appear in the test plan (when the order is relevant). For more information about the MANO itself, the reader is encouraged to read Section 8.

### 4.2.5.1 The instrument

Table 2 provides item descriptions and default values needed to configure the MANO instrument. See Figure 19 for a sample screen shoot.

Table 2 MANO instrument setup

Item	Settings	Description
Address	10.20.2.44	IP of MANO (points to cloud5)
Connection timeout	20s	Time to try connectivity
Domain name	a.cloud5.morse.uma.es	Prefix “a” given by MANO, the rest is configured at OpenStack DNS service. Will be used by VMs instantiated through MANO
Port	8008	MANO port for REST API (default)
Scheme	https	Forced by MANO configuration
Authentication token	YWRtaW46YWRtaW4=	MANO default credentials admin:admin encoded in BASE64
Allow self-signed certificates	YES	Non-strict TLS check
Private keys	See below	See below

The “Private keys” section allows for adding the private keys which will be used to establish secure connections with the Virtual Machines instantiated by MANO which will make sure that the public key matching the private one is injected into the VM. The reference to that key must be placed in Network Service Record json file in the “ssh-authorized-key” section. Moreover, the private key must be in OpenSSH format. Please observe, that the most popular Windows tool, i.e., *putty key generator* produces keys in its own format (ppk – putty private key) which is not compatible with OpenSSH and needs to be converted by the *putty key generator* tool (see Figure 20) and saved into a separate file (most often named *id\_rsa.pub*, note that its counter-

part *id\_rsa* commonly contains the private key which must remain secret). The path to this file needs to be provided when configuring the MANO instrument in TAP (“private key path”).

Furthermore, the private key, when created, may be chosen to be protected by a passphrase (one-time entered password which unlocks the key and allows its usage). If there is no passphrase, the field “Pass phrase” in MANO TAP configuration should be left blank. If there is a passphrase, it can be entered in the appropriate field, however, it will be stored in an insecure way (in clear text) in the configuration.

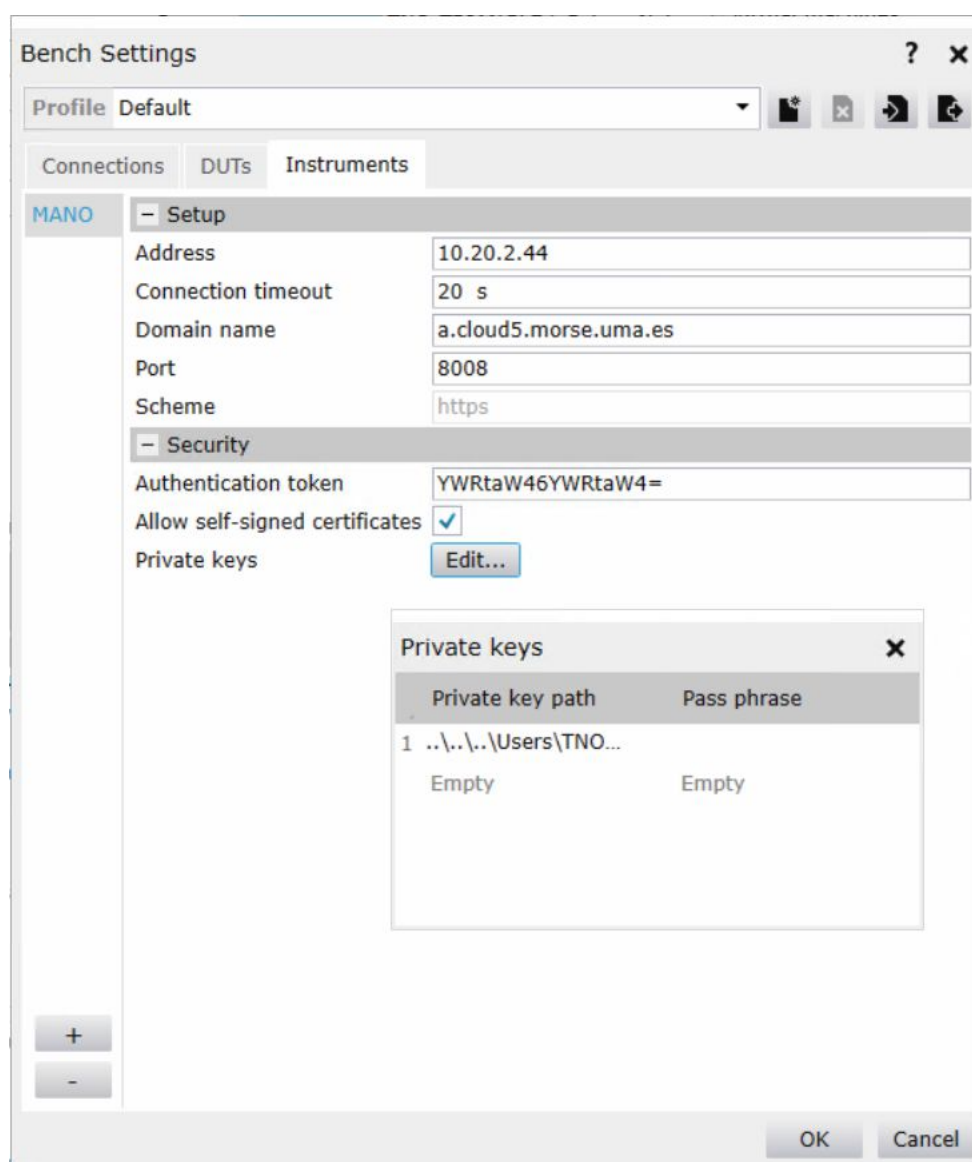


Figure 19 MANO instrument setup

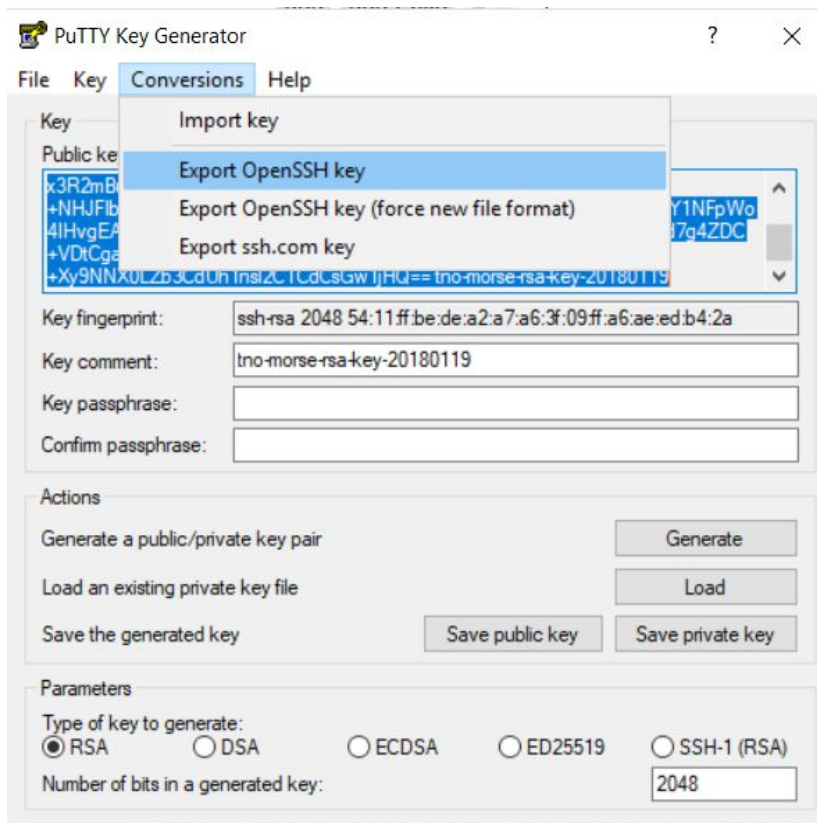


Figure 20 Putty key conversion

#### 4.2.5.2 Test step MANO setup

This is a first and mandatory step in the test plan. Its primary function is to load the Network Service Descriptors.

Table 3 Test step MANO setup

Step	Setting	Description
MANO setup	Instrument	Select which MANO instrument to use
	Network Service Record file	json file containing Network Service Descriptor
	<b>Result</b>	<b>Description</b>
	Virtual machines	Names of VMs, constructed using entries in Network Service Descriptor file. These names will be used to construct FQDNs of the hosts.
	Id	UUID of the Network Service Record
	JSON content	Content of the Network Service Record file
	Comment	Free-form data entry



#### 4.2.5.3 Test step Contact MANO

Typically, a second step, although not mandatory. Makes REST call towards MANO to verify if it is reachable.

Table 4 Test step Contact MANO

Step	Result	Description
Contact MANO	Comment	Free-form data entry

#### 4.2.5.4 Test step - Set up Network Service Record

In this step, the MANO orchestrator is contacted to consume a Network Service Descriptor (NSD) file and create the instances and the networks using the Virtual Infrastructure Manager (an OpenStack cloud in our case). While sending the command towards MANO is fast (hence the default 20s timeout should be sufficient), the full execution of this step, i.e., the state where all the VMs and all the services are up and running, may take several minutes (for example, a typical boot time of an Ubuntu 16.04 VM is about 3-4 minutes).

Note that the Network Service Record id is unique and no equally identified record should exist (e.g., due to previous instantiations). If the MANO contains a Network Service Record identified by the same id as the request, it will reject the message and the step will be aborted.

Table 5 Test step Set up Network Service Record

Step	Setting	Description
Set up Network Service Record	Connection timeout	Time to try connectivity
	Result	Description
	Comment	Free-form data entry

#### 4.2.5.5 Test step Resolve hostnames

This step contacts the MANO and retrieves the hostnames of the virtual machines it has created.

Table 6 Set up Resolve hostnames

Step	Setting	Description
TRIANGLE		PU





Set up  
hostnames

Resolve

Connection timeout	Time to try connectivity
<b>Result</b>	<b>Description</b>
Comment	Free-form data entry

#### 4.2.5.6 Test step Execute remote command

Execute an arbitrary command on an arbitrary (ssh-reachable) host. See Figure 21 for the example.

Table 7 Test step Execute remote command

Step	Setting	Description
Execute command	remote	
	Host	IP address or FQDN of the host on which the command is to be executed
	ssh port	Port on which sshd is listening
	Connection timeout	Time to try connectivity
	Connection retry attempts	Number of retries in case of timeout
	Command	The command to be executed on the host to which ssh session is established (as configured above)
	Command timeout	Timeout for the command which was executed (understand as no output appears as a result of a command)
	Username	Username used for ssh session
	Password	Password (if needed; may be empty is e.g., ssh keys are deployed) for ssh session
	<b>Result</b>	<b>Description</b>
	Regular expression	REGEXP which will be used to process output of the command. Uses stdout of the host to which ssh session is established (as configured above) as its input. In case of match, the test passes; otherwise it fails.
	Comment	Free-form data entry





- Setup	
Host	10.10.10.10
SSH port	22
Connection timeout	20 s
Connection retry attempts	2 times
Command	ping 20.20.20.20 -c 5
Command timeout	60 s
- Security	
Username	ubuntu
Password	
- Results	
Regular expression	(.*5 received.*)
Comment	

Figure 21 Execute remote command

#### 4.2.5.7 Test step Execute remote command known host

Execute an arbitrary command on MANO-instantiated host (must be ssh-reachable). Optionally, two names of the (other) MANO-instantiated VMs can be provided as parameters. A typical usage is “ping Host-2 from Host1”, see Figure 22 for such an example.

Table 8 Execute remote command known host

Step	Setting	Description
<i>Execute remote command known host</i>	Host	IP address or FQDN of the host on which the command is to be executed
	ssh port	Port on which sshd is listening
	Connection timeout	Time to try connectivity
	Connection retry attempts	Number of retries in case of timeout
	Command	The command to be executed on the host to which ssh session is established (as configured above). Parameters' usage is indicated by '{0}' and '{1}' tokens
	Command timeout	Timeout for the command which was executed (understand as no output appears as a result of a command)
	Parameter 1 (optional)	Name of the MANO-instantiated VM
	Parameter 2 (optional)	Name of the MANO-instantiated VM
	Username	Username used for ssh session
	Password	Password (if needed; may be empty is e.g., ssh keys are deployed) for ssh session



Result	Description
Command preview	Preview of the command to execute
Regular expression	REGEXP which will be used to process output of the command. Uses stdout of the host to which ssh session is established (as configured above) as its input. In case of match, the test passes; otherwise it fails.
Comment	Free-form data entry

Figure 22 Execute remote command known host

#### 4.2.5.8 Test step Execute remote command to Host with known host

Execute an arbitrary command on an arbitrary host (must be ssh-reachable). Optionally, two names of the MANO-instantiated VMs can be provided as parameters. An exemplary usage is “ping Host-2 from Host-1”, see Figure 23 for such an example.

Table 9 Execute remote command to Host with known host

Step	Setting	Description
<i>Execute remote command to Host with known host</i>	Host	IP address or FQDN of the host on which the command is to be executed
	ssh port	Port on which sshd is listening
	Connection timeout	Time to try connectivity



Connection retry attempts	Number of retries in case of timeout
Command	The command to be executed on the host to which ssh session is established (as configured above). Parameters' usage is indicated by '{0}' and '{1}' tokens
Command timeout	Timeout for the command which was executed (understand as no output appears as a result of a command)
Parameter 1 (optional)	Name of the MANO-instantiated VM
Parameter 2 (optional)	Name of the MANO-instantiated VM
Username	Username used for ssh session
Password	Password (if needed; may be empty is e.g., ssh keys are deployed) for ssh session
<b>Result</b>	<b>Description</b>
Command preview	Preview of the command to execute
Regular expression	REGEXP which will be used to process output of the command. Uses stdout of the host to which ssh session is established (as configured above) as its input. In case of match, the test passes; otherwise it fails.
Comment	Free-form data entry

**Step Settings** ▼ ×

– Setup

Host

10.10.10.10

SSH port

22

Connection timeout

20 s

Connection retry attempts

2 times

Command

ping -c 5 {0}

Command timeout

60 s

Parameter 1

tri\_test202.u16043\_4c\_8G\_1iface\_vnfd--1

Parameter 2

– Security

Username

ubuntu

Password

– Results

Command preview

ping -c 5 <tri\_test202.u16043\_4c\_8G\_1iface\_vnfd--1>

Regular expression

(.5 received.\*)

Comment

Figure 23 Execute remote command to Host with known host



#### 4.2.5.9 Test step Tear down Network Service Record

Typically the last step, however it is not mandatory. The MANO instrument is instructed to terminate the instantiated Network Service Record. The Id of the NSR is read from a json file, supplied in the Test step MANO setup

Step	Setting	Description
Tear down Network Service Record	Connection timeout	Time to try connectivity
	Id	UUID of the Network Service Record to be terminated.
	<b>Result</b>	<b>Description</b>
	Comment	Free-form data entry

#### 4.2.5.10 Sample test plan

A sample test plan is delivered with the TAP plug-in (file *ssh\_NetworkServiceRecord tri\_test202.TapPlan* along with the sample Network Service Descriptor (file *NetworkServiceRecord tri\_test202.json*).

Test Plan <i>ssh_NetworkServiceRecord tri_test202</i>				
+ - ⬇️ ▶ Run ⏸️ ■ Repeat ▾				
Step Name	Verdict	Duration	Step Type	
<input checked="" type="checkbox"/> TNO   MANO setup			TNO-TRIANGLE \ TNO   MANO setup	
<input checked="" type="checkbox"/> Contact MANO			TNO-TRIANGLE \ Contact MANO	
<input checked="" type="checkbox"/> Set up Network Service Record			TNO-TRIANGLE \ Set up Network Service Record	
<input checked="" type="checkbox"/> Delay			Basic Steps \ Delay	
<input checked="" type="checkbox"/> Resolve host names			TNO-TRIANGLE \ Resolve hostnames	
<input checked="" type="checkbox"/> Check if host is alive			TNO-TRIANGLE \ Execute remote command known host	
<input checked="" type="checkbox"/> Execute remote command known host			TNO-TRIANGLE \ Execute remote command known host	
<input checked="" type="checkbox"/> Tear down Network Service Record			TNO-TRIANGLE \ Tear down Network Service Record	

Figure 24 MANO sample test plan

### 4.3 New features in the Quamotion WebDriver

The latest Quamotion WebDriver release contain several improvements to enable more complex application flows. Gestures and actions to click, enter-text, swipe, get and set values are now available for the all automation types (App, Device and Web automation).

Quamotion added the support to containerize ADB/USBMUXD, the Quamotion WebDriver and the AppFlow execution. This has several advantages:



- **Devices can be isolated:**  
On both ADB and usbmuxd level we can filter out the device communication. This enables to expose only one device to the application flow scripts and avoid any interference with other devices available on the testbed.
- **Script execution can be isolated**  
Containers containing the Quamotion WebDriver and the Quamotion Runner are provided to execute scripts in full isolation. This gives the best possible protection against harming the TestBed infrastructure and prevents extraction of confidential data/application from other users.
- **Improved stability**  
By spinning new containers for each test the test environment is reset completely (except for the physical device at this moment). Lab experiments shows a drastic improvement of stability as each test start from a clean state.

## 5 Measurements and data collection

### 5.1 KPIs computation

The following features have been implemented:

- A generic data management framework (TriangleKpi.Core) that provides the basic functionality required for the generation of the KPIs based on the raw data obtained during the testplan execution.
- A set of TAP steps (grouped on the 'Tap.Plugins.TriangleKpi' plugin) that make use of the core package for calculating the different KPIs defined for each of the available domains.

#### 5.1.1 TriangleKpi.Core

The TriangleKpi.Core package has been developed as a generic data management framework, and, as such, does not have any dependency with the TAP engine or any other component of the TRIANGLE testbed. The framework is based on the concept of Pipelines, which consist on a collection of components that are executed in order, transforming the data in different steps until a set of results can be generated. Each component (except for the Processors, which generate Results) produce a list of StructuredData, which encapsulate a table of information, and these StructuredDatas can be used as input for one or more different Processors. The available components for use on the Pipelines are:

- **Loaders:** Loaders are the starting point of the Pipeline. These components can be used to obtain one or more StructuredDatas from different sources. The Core package includes the CsvLoader, that can extract the data from CSV files. Developers can create custom Loaders by implementing the ILoader interface or extending the Loader base class.
- **Parsers:** Parsers take a set of input StructuredData, which can be generated by Loaders or by previous Parsers, manipulate the data contained in them, and generate one or more StructuredDatas that can be feed to another Parsers or to Processors. The manipulations available allow the modification or removal of existing rows and columns of the table, or the generation of new elements based on the existing information.



- Processors: Processors generate results by using the information contained in the input StructuredDatas. For example, AverageProcessor will report a result for every column in the input, with the average of the data.

Using this architecture, the developer can define custom Pipelines for generating different results.

### 5.1.2 Tap.Plugins.TriangleKpi

The TriangleKpi tapplugin contains a set of test steps that can be used for generating the set of KPIs defined for each domain, using the results generated by the previous campaign execution as source. This plugin also defines an additional Loader based on the interface defined on the Core package: TapDataLoader can generate the initial StructuredData for the Pipelines by querying a set of data from the TAP's result database.

Prior to the execution of the KPI extraction steps the results generated by a testplan execution by using the 'Test Case Labeler' step must be labeled. This step is executed at the beginning of each campaign run and provides the basic information that TapDataLoader requires for loading the desired results.

Step Settings	
Domain	RES
Use Case	CO
Test Case	1
Network Scenario	UR_OF

**Figure 25: Configuration parameters on the Test Case Labeller step**

Each step exposes a similar set of configuration parameters:

- The basic settings include the Test Case id and network scenario of the testplan.
- The Test Case, Campaign and Test Plan sections can be used to define associated metadata for the generated KPIs.
- The Input / Output section defines the input database and optional ETL database: The steps can save the generated KPIs and metadata directly into an ETL database, or only as standard TAP results.

Step Settings	
Res	CO, 001
Network Scenario	UR_OF
- Test Case	
App	Exoplayer
Device	Galaxy S7
- Campaign	
Test Bed	UMA
Author	TestUser
- Test Plan	
DUT Category	Phone
Product ID	sm-g930f
HW Version	1
SW Version	7.0
- Input / Output	
Sql Input	TapResults @ localhost:5432
ETL DB Output	<input checked="" type="checkbox"/> ETLDB

**Figure 26: Configuration parameters on the RES step. All steps include the same basic settings.**

- Additionally, some steps may require additional data to perform the required calculations: For example, the App User Experience step calculates KPIs based on the resolution used by the application (for example, during video playback or gaming). In these cases, the calculation is performed using 4k as the maximum resolution by default, but it is possible to specify the maximum resolution of the DUT for generating these MOS values using more realistic information.

- KPI calculation values	
Device resolution	1080

**Figure 27: Additional settings on the AUE step**

- Each of the provided steps defines a custom Pipeline that is used to generate the desired KPIs

## 5.2 Metrics and mark computation

For the calculation of the TRIANGLE Mark the project is following the process described in section 4 of deliverable D3.3 - Testing and reporting software tools. After each successful testplan execution the Orcomposutor will make use of the steps defined in the previous section to extract all the available KPIs for the different domains. Once all the required testplan executions for the campaign have been completed successfully the Orcomposutor starts the processing.





### 5.3 Instrumentation library

A new version of the instrumentation library has been delivered in time for Release 3 of the testbed. This new version provides measurement points to compute all the KPIs specified in the test cases defined in D2.2 and also support the definition of custom measurements points. App Annex 2 includes a list of all the measurements supported in Release 3.

### 5.4 Measurement calculation without using the instrumentation library (UMA)

In case it is not possible to use the Instrumentation Library on your application (for example, because it has been developed using the Android NDK or it is not possible to include external libraries), app developers can still instrumentalize their applications and take advantage of the automatic measurement calculation provided by the TRIANGLE testbed. This is possible by writing messages that follow the same format as the messages generated by the instrumentation library, and generating them at the same situations in which a method from the library would be included.

For example, the following snippet could be used for generating the required measurement point at the end of an FTP download in an Android NDK application:

```
#define LOGI(...) ((void)__android_log_print(ANDROID_LOG_INFO, "ftp.Native", __VA_ARGS__))
#define INST(...) ((void)__android_log_print(ANDROID_LOG_INFO, "TriangleInstr", __VA_ARGS__))

// [...]

if (downloadFtp(url_c, port_c, user_c, pass_c, buffer_c, binding_c, &speed)) {
    LOGI("FTP download completed, speed %.2f kbytes/s", speed);
    INST("Hs\tDownload\tFile Download - End\t%i\ttrue", transferId);
} else {
    LOGI("FTP download failed");
    INST("Hs\tDownload\tFile Download - End\t%i\tfalse", transferId);
}
```

**Figure 28 Snippet for generating measurement points with the same format that the instrumentation library**

Where “\t” corresponds to the Tab character. The message that corresponds to each of the supported measurement points can be seen on the ‘Measurement point methods’ section on the documentation of the specific Instrumentation Library for each operating system, as “Generated message”.

### 5.5 New features in DEKRA TACS Performance Tool (DEKRA)

There have been two major features included in Release 3 provided by the DEKRA Performance Tool component:

- DRA Content Stall measurements
- RES GPU Usage measurements

In the scope of DRA test specification [D2.2\_AP6], Content Distribution Streaming Reference App use case and more specifically Content Stall KPI, reporting the percentile curve (a.k.a CDF) as KPI summarization was specified for the TRIANGLE Mark scoring. The ability to collect the





measurements for Content Stall KPI resides in the DEKRA Performance Tool component. In Release 2, even though the DEKRA Tool internally measured every Content Stall instance and its duration, only average and maximum values were exposed to the TRIANGLE test bed. In Release 3, all the Content Stall instances as measured by the DEKRA Tool are now exposed and made available to the test bed so that the ETL component can compute the required KPI summarization (i.e., percentile curve) which is necessary to provide support for the TRIANGLE mark scoring process.

In the scope of RES test specification [D2.2 AP5], reporting the device GPU usage was specified. In Release 3 the DEKRA Performance Tool is able to collect GPU Usage from Android devices. That new measurement capability has been integrated into the TRIANGLE test bed thus completing the full coverage of the RES tests specification.

The device interface that the DEKRA Tool uses for reading the GPU load is `"/sys/class/kgsl/kgsl-3d0/gpubusy"`, which reports the total and busy cycles DEKRA Tool uses to calculate the GPU usage: `/sys/class/kgsl/kgsl-3d0/gpubusy` reports two integers `g0` and `g1`. The actual load as a percentage can be calculated as  $(g0/g1*100)$ .



## **6 RAN (Radio Access Network)**

No new features in the UXM.



**Document:** ICT-688712-TRIANGLE/D3.4

**Date:** 29/03/2018

**Dissemination:** PU

**Status:** Final

**Version:** 1.0

---

## 7 EPC

EPC has been updated to Release 13 which includes support for NB-IoT.



## 8 Transport

This section describes the orchestrated cloud platform delivered by TNO to the TRIANGLE testbed. It consists of the two main parts: The Orchestrator and the Virtual Infrastructure Manager.

The Management and Network Orchestration (MANO) stack allows for rapid deployment of Virtualized Network Functions (VNFs), their flexible configuration, lifecycle monitoring and decommissioning in order to automatically operate a potentially complex Network Service (NS). A model driven approach is taken to describe VNFs and NSs. MANO enhances interoperability with other components in the system such as Virtual Infrastructure Managers (VIMs) where the (virtualized) functions are deployed. To allow these deployments, VIM manages storage and networking resources, providing flexible and scalable infrastructure for the modern services such as Virtual Reality or ultra-high definition TV.

### 8.1 Cloud infrastructure description

TNO has realized and integrated a cloud environment based on OpenStack in the TRIANGLE testbed. This section describes how the physical and virtual infrastructure is set up and has been configured.

The implemented cloud environments are running on a single hypervisor described in Section 8.1.1. The hypervisor hosts multiple clouds described in Section 8.1.2 and performs routing and firewalling for their respective networks described in Section 8.1.3. The Juju models describing the cloud configuration are found in Section 8.1.4. An overview of access to the OpenStack Dashboard, command-line Client and REST APIs for using the cloud is given in Section 8.1.5, while Section 8.1.6 gives an overview of all important used IP addresses. An overall overview of the infrastructure is presented in Figure 29.

#### 8.1.1 Hypervisor

Based on the instructions of the TRIANGLE project, the cloud environment has been installed on a single server placed in the testbed location at Malaga University. To provide isolation between the different types of nodes and physical machines usually found in a cloud environment, different functional nodes such as the cloud controllers, compute nodes, storage nodes, networking nodes and gateways, deployment managers and MANO orchestrators are configured to run as KVM instances on the hypervisor server managed by *libvirt*. Overall, there are 3 types of KVM instances:

- Manually-deployed instances, instances that are installed manually from a ISO. These nodes include Ubuntu MAAS and the DANE (note that DANE can and typically will be instantiated as a part of the specific experiment) .
- MAAS-deployed instances, these instances have not been installed manually, but are installed through Ubuntu MAAS configuration and are later configured manually. These instances include Juju and the MANO orchestrators.
- Juju-configured instances, these instances have been installed and configured fully automatically through Juju models containing abstract configuration and functional linking description. Juju employs Ubuntu MAAS to automatically install nodes with an operating system prior to deploying configuration. The instances fully deployed through Juju consist of the Juju-Controller itself that in turn manages the cloud environments, and all Controller, Compute, Networking and Storage nodes found in each cloud.

The hypervisor has 4 SATA hard disks, configured in 2 software RAID-10 arrays for reliability purposes. The first array hosts the hypervisor software and configuration itself, while the second array hosts the storage of the instances. For performance reasons, the TRIANGLE consortium should consider upgrading this system to a two-array, four-hard disk hardware-based RAID system based on SAS instead of SATA.

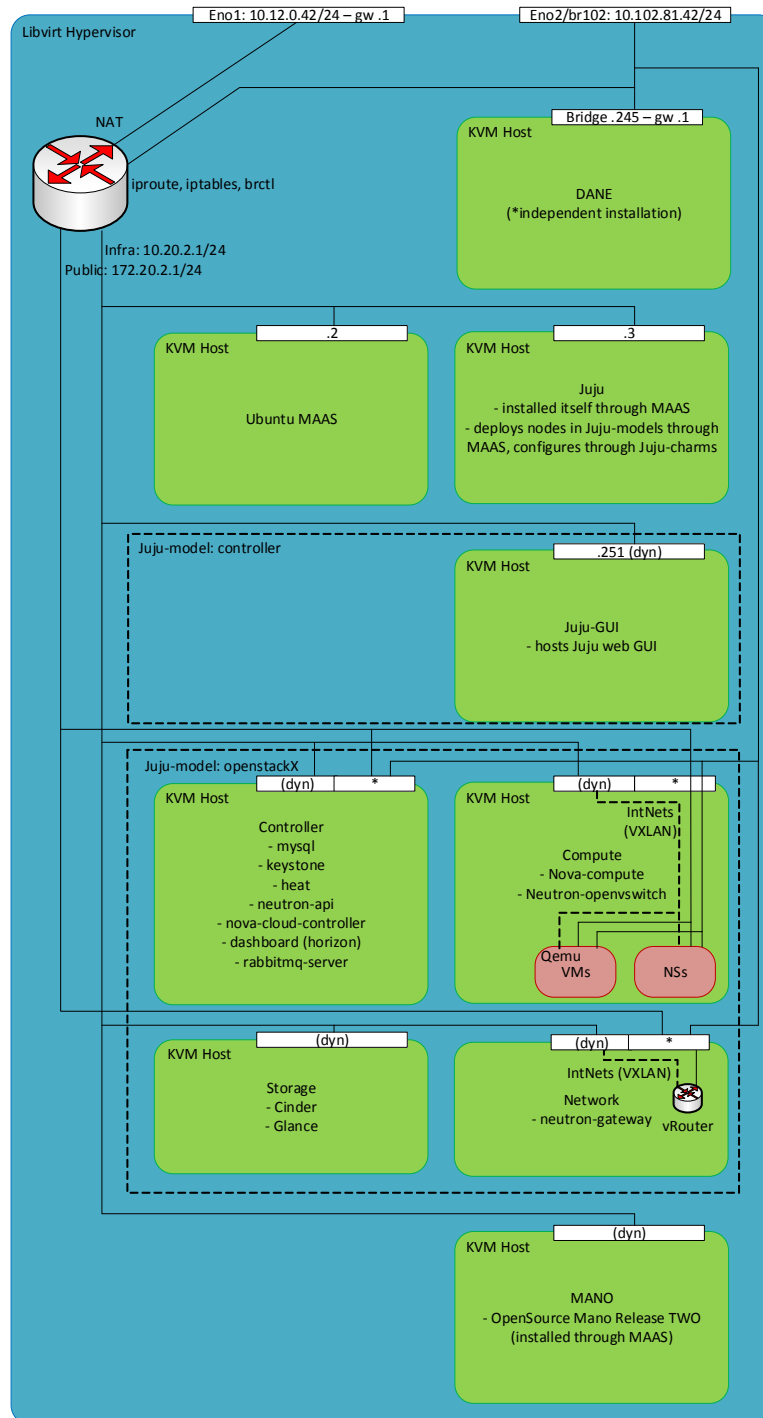


Figure 29: Architecture overview

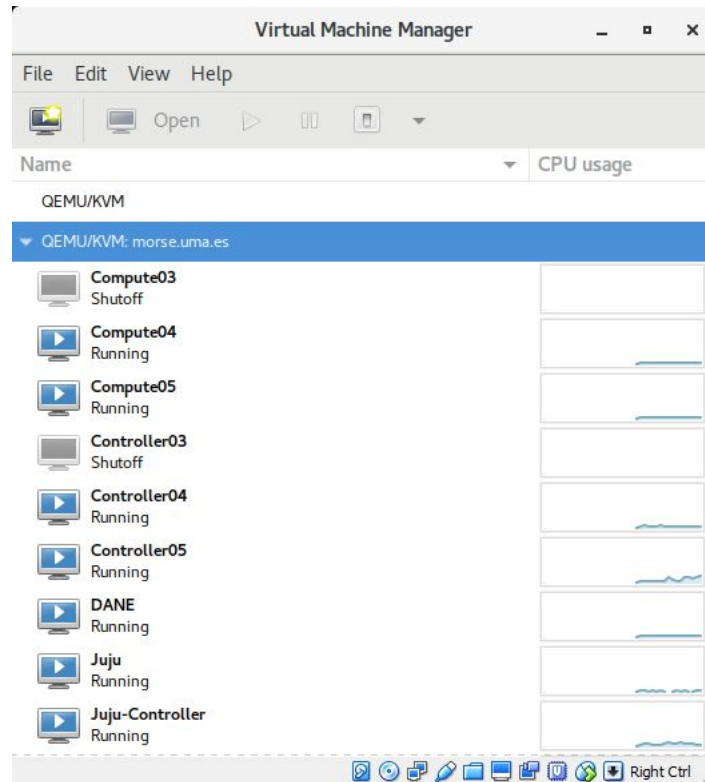


Figure 30: Hypervisor configuration through virt-manager

<input type="checkbox"/>	FQDN   MAC	Power	Status	Owner	Cores	RAM (GiB)	Disks	Storage (GiB)
<input type="checkbox"/>	Compute03.maas	Off	Ubuntu 16.04 LTS	triangle	32	64.0	1	274.9
<input type="checkbox"/>	Compute04.maas	On	Ubuntu 16.04 LTS	triangle	32	64.0	1	274.9
<input type="checkbox"/>	Compute05.maas	On	Ubuntu 16.04 LTS	triangle	32	32.0	1	274.9
<input type="checkbox"/>	Controller03.maas	Off	Ubuntu 16.04 LTS	triangle	16	32.0	1	274.9
<input type="checkbox"/>	Controller04.maas	On	Ubuntu 16.04 LTS	triangle	16	32.0	1	274.9
<input type="checkbox"/>	Controller05.maas	On	Ubuntu 16.04 LTS	triangle	16	16.0	1	274.9
<input type="checkbox"/>	Juju-Controller.maas	On	Ubuntu 16.04 LTS	triangle	2	4.0	1	274.9
<input type="checkbox"/>	Juju.maas	On	Ubuntu 16.04 LTS	triangle	1	1.0	1	274.9
<input type="checkbox"/>	MANO.maas	Off	Ubuntu 16.04 LTS	triangle	4	4.0	1	274.9
<input type="checkbox"/>	Network03.maas	Off	New		0	0.0	0	0.0
<input type="checkbox"/>	Network04.maas	On	Ubuntu 16.04 LTS	triangle	8	8.0	1	274.9
<input type="checkbox"/>	Network05.maas	On	Ubuntu 16.04 LTS	triangle	8	8.0	1	274.9
<input type="checkbox"/>	OSM3.maas	Off	Ubuntu 16.04 LTS	triangle	4	8.0	1	274.9
<input type="checkbox"/>	OSM4.maas	On	Ubuntu 16.04 LTS	triangle	4	8.0	1	274.9
<input type="checkbox"/>	OSM5.maas	On	Ubuntu 16.04 LTS	triangle	4	8.0	1	274.9
<input type="checkbox"/>	Storage03.maas	Off	Ubuntu 16.04 LTS	triangle	8	8.0	2	2199.0
<input type="checkbox"/>	Storage04.maas	On	Ubuntu 16.04 LTS	triangle	8	8.0	2	2199.0
<input type="checkbox"/>	Storage05.maas	On	Ubuntu 16.04 LTS	triangle	8	8.0	2	2199.0

Figure 31: Overview of nodes deployed through MAAS, cloud-related nodes are initiated and controlled through Juju.



### 8.1.2 Clouds

The cloud environment exists out of 3 different clouds, all running on the previously described hypervisor. All clouds are running OpenStack Pike deployed through the Ubuntu MAAS and Juju services described in Section 8.1.4. Due to historic reasons, the clouds have the following names and functionality:

- Cloud5; the Stable cloud that can be used in production
- Cloud4; the Staging/Integration cloud in which we test and verify functionality before moving them to the Stable cloud.
- Cloud3; the Development cloud for experimental testing of functionality before moving it to Staging

### 8.1.3 Networks

The following networks with respective subnets are considered from the cloud environments. Unless stated otherwise, the .1 IP address specify both the gateway and DNS server for that subnet and all these networks exist at the hypervisor through a *virsh* virtual network. DNS requests are forwarded to the UMA DNS server at 150.214.40.11 by the recursive DNS server *dnsmasq* running on the hypervisor.

The following networks are exposed from cloud environment to the testbed network.

- Access Network - 10.12.0.0/24:  
This is the access network used by the hypervisor to access and forward traffic towards the Internet, only the main Network Interface Card (NIC) of the hypervisor has an IP address in this range. Ideally, Network Address Translation (NAT) should be performed by the main router of the TRIANGLE network, however, due to limited functionality of the main router the hypervisor masquerades outgoing traffic on this NIC towards the Internet.
- TAP Network - 10.102.81.42:  
This is the network used for the devices part of and under test by the TAP testbed. From Cloud4 and Cloud5 instances can be fired up that are directly bridged into this network, for this respectively the ranges 10.102.81.100-149 and 10.102.81.150-199 have been reserved. Due to limitations in the main router of the TRIANGLE testbed, hosts besides cloud instances in this network need to manually configure network routes via 10.102.81.42 (the hypervisor IP address) to access the following 4 subnets.
- OpenStack Infra - 10.20.2.0/24:  
Through the Infrastructure network the VMs on which the cloud-infrastructure itself run are configured and maintained.
- OpenStack Public5 (Stable) - 172.16.4.0/24:  
This is the public network connected to the Stable cloud5, instances and floating IPs on this network are dynamically addressable by hostname through the subnet *<hostname>.cloud5.morse.uma.es*.
- OpenStack Public4 (Staging) - 172.16.3.0/24:  
This is the public network connected to the Staging cloud4, instances and floating IPs on this network are dynamically addressable by hostname through the subnet *<hostname>.cloud4.morse.uma.es*.
- OpenStack Public3 (Development) - 172.16.2.0/24:  
This is the public network connected to the Development cloud3.

The following internal networks exist for functional reasons within the OpenStack clouds; but are not exposed outside the cloud environments or hypervisor. They exist as private VXLAN networks within the cloud to interconnect instances with each other. Access to the exposed networks is realized using virtual routers in OpenStack performing NAT.

- OpenStack Internal5 (Stable) – Internal network on Stable Cloud5 interconnected to network OpenStack Public5.
- OpenStack Internal4 (Staging) – Internal network on Staging Cloud4 interconnected to network OpenStack Public4.
- OpenStack Internal3 (Development) – Internal network on Development Cloud3 interconnected to network OpenStack Public3.
- OpenStack TAP-Internal5 (Stable) - Internal network on Stable Cloud5 interconnected to network TAP Network.
- OpenStack TAP-Internal4 (Staging)- Internal network on Stable Cloud4 interconnected to network TAP Network.

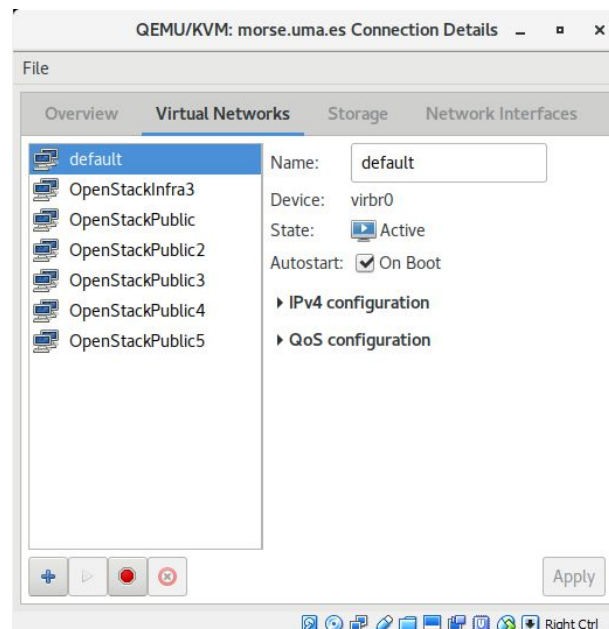


Figure 32: Network overview of networks managed by lib-virt

#### 8.1.4 Juju models describing the cloud infrastructure

The OpenStack clouds available at the TRIANGLE testbed are deployed and configured through Ubuntu Juju, a service configuration and modelling tool that automatically configures servers based on the abstract configuration expressed in a Juju model. Each functionality is expressed and configured through a Juju charm, which can be mapped to a physical machine, a virtual machine or an LXD container. Due to the ease of creating, reconstructing reconfiguring and cleaning up functions, we have a very fine-grained distribution where each function described by a Juju charm is mapped to its individual LXD container in a KVM instance or the root container of a KVM instance if the charm's functionality requires so. Figure 33 presents an overview of the Juju model, containing of the following functions mapped to the following nodes:

- Controller: A node executing the functions related to management, configuration and monitoring of the cloud, consists out of
  - Cinder-API: Offers the API and scheduling functionality for the block storage service and provides management of volumes.





- Heat: OpenStack Orchestration service, a non-ETSI aligned alternative for MANO/NFV orchestration (note: a different solution, i.e., Open Source Mano, is actually used as the orchestrator).
- Horizon: The OpenStack dashboard / web GUI.
- Keystone: Identity, authentication and authorization service.
- MySQL: A Percona database cluster running MariaDB instances
- Neutron-API: Virtual network service, enabling network management, QoS, ACLs, etc.
- Nova-cloud-controller: Cloud computing controller, scheduling, managing and monitoring computing resources through its subservices nova-scheduler, nova-api and nova-conductor.
- RabbitMQ: Advanced Message Queuing Protocol server used by all services to interconnect.
- Compute:
  - Nova-compute (root container): Provides hypervisor service to run instances on.
- Network:
  - Neutron-gateway (root container): Provides central networking services such as virtual routers to interconnect external (bridged or routed) networks and internal (VXLAN) private networks through routing, firewalling and NAT.
- Storage:
  - Cinder-volume (root container): Actual storage of LVM volumes on nodes, managed by Cinder-API.
  - Glance: Image registration and discovery service
- Subordinate functions: Elements that upgrade functionality of the main elements described in the previous 4 categories.
  - Neutron-OpenvSwitch: Provides Neutron-API and Nova-compute with Open vSwitch functionality.
  - NTP: Provides time synchronization to nodes.

The Juju model configuring the clouds is custom-made for the TRIANGLE testbed and in particular contains the following additional configuration:

- L2 and L3 network connectivity between external and private internal (VXLAN) networks using Open vSwitch SDN switches throughout.
- Separate networking nodes to provide gateway and floating IP address functionality.
- DNS integration such that nodes in the routed public networks can be accessed through their hostnames <hostname>cloud[4-5].morse.uma.es.
- Additional non-ETSI aligned orchestration module through the OpenStack Heat package (note: a different solution, i.e., Open Source Mano, is actually used as the orchestrator).
- Has split storage control and storage volume functionality for future scalability through LVM.
- Has per-machine restriction and tag descriptions to automatically select the correct VMs for their respective functions.

The full YAML export of the Juju model and the cloud-config scripts are available and are used to automatically create all cloud configuration once a cloud is reinitiated.

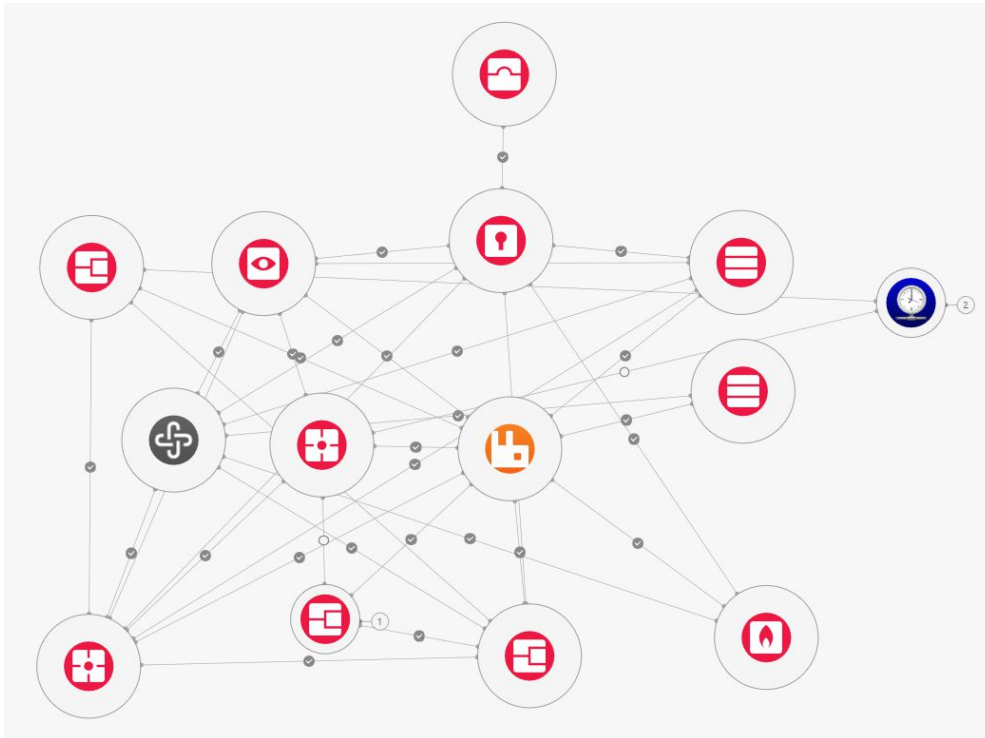


Figure 33: Juju model describing functional elements and their connections

### 8.1.5 Dashboard and API access

The cloud instances can be accessed and configured through both a Web GUI Dashboard (Horizon) and service-specific APIs. This section presents an overview of the available configuration services, section 8.1.6 gives an overall overview of the used IP addresses. Note that these IP addresses will change when elements of the cloud model are deleted and reinstalled through Juju. All addresses are publicly accessible from the networks described in section 8.1.3, given that local static routing is configured when residing in the Access or TAP Network. When connecting from outside the TRIANGLE Testbed network, an SSH Dynamic Tunnel (SOCKS Proxy) towards a node within the network can be used.

#### *OpenStack Dashboard*

The most easy way to configure and administer projects, instances, networks, authentication, storage, images, etc., etc., is through the OpenStack Dashboard supplied by the Horizon package. It is accessible through either <http://<OpenStack-Dashboard>> or <https://<OpenStack-Dashboard>>, with the appropriate IP address selected from section 8.1.6. Annex 5 gives an introduction into the usage of OpenStack Dashboard.

#### *CLI Access*

Additionally, all configuration can be applied through the OpenStackClient (OSC) command-line client. OSC provides a shell interface with all configuration options available and is available under the *python-openstackclient* package in most package repositories. A full documentation of OSC can be found at [3] In a nutshell, after installation the command *openstack* provides a shell that auto-completes and gives information on the fields necessary to complete a command. The following environment variables need to be set to connect to Stable Cloud5, which can be easily saved in a local file *admin-openrc5.sh* that can be sourced through ``. admin-openrc5``:



Table 10: admin-openrc5.sh

```
export OS_AUTH_URL=http://10.20.2.45:5000/v3
export OS_PROJECT_DOMAIN_NAME="default"
export OS_PROJECT_NAME="admin"
export OS_USER_DOMAIN_NAME="default"
unset OS_TENANT_ID
unset OS_TENANT_NAME
export OS_USERNAME="admin"
export OS_PASSWORD="admin"
export OS_REGION_NAME="RegionOne"
if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
export OS_INTERFACE=public
export OS_IDENTITY_API_VERSION=3
unset OS_TOKEN
```

### API Access

It is also possible to have programmatic access to the configuration of the OpenStack Cloud. In fact, both the OpenStack Dashboard and the OpenStackClient rely fully on API access to read and write OpenStack configuration. Additionally, API Access is for example used by the MANO orchestrators (OSM) connected to the OpenStack clouds to administer the appropriate configuration and is useful for custom environments to integrate with OpenStack. For example, we have used direct API access in our sample TAP scripts to configure QoS parameters in OpenStack.

Annex 4 provides a short introduction of this API. A full overview of all OpenStack API Documentation can be found at [4], we will give a short introduction to each API available in the TRIANGLE clouds and how to operate those.

*Additionally*, the cloud configuration can be orchestrated from the MANO orchestrators based on Open Source MANO (OSM) and the TAP client respectively described in sections 8.2 and 8.2.4.5.

### 8.1.6 Overall overview of IP addresses

The following services are accessible at the following IP addresses through the following connection methods

Host Name	IP addresses	Connections	Authentication
OpenStackCloud (Hypervisor)	10.12.0.42 10.102.81.42	ssh://morse.uma.es:11300 ssh://<IP-Address>	morse+tno (Key-auth)
MAAS	10.20.2.2	ssh://triangle@10.20.2.2 http://10.20.2.2/MAAS	triangle:triangle
Juju	10.20.2.3	ssh://ubuntu@10.20.2.3	(Key-auth)
Juju-Controller	10.20.2.215	https://10.20.2.251:17070/gui/u/admin	admin / Run `juju gui` at Ubuntu Juju for pw.
DANE	10.102.81.245	ssh://morse.uma.es:11341 ssh://10.102.81.245	tno (Key auth)
OSM5	10.20.2.44	https://10.20.2.44:8443 ssh://ubuntu@10.20.2.44	admin:admin (Key-auth)



For the Stable Cloud 5, the following services are deployed by Juju on 4 VMs labelled Compute05, Controller05, Network05 and Storage05. On these nodes, Juju creates an LXD container per service with its own IP address to allow fine-grained per-service configuration, deletion and recreation. Access to these web and API services is explained in section 8.1.5. Logging into the nodes through SSH is seldomly necessary as they are configured through the Juju model “OpenStack5” through the Juju Web GUI at *Juju-Controller*. If SSH access is necessary to any of these services, this is possible through the following command when logged into the Juju node: ``juju ssh -m openstack5 <unitname>``, where the unit is build up from the servicename followed by its instantiation (generally “/0” when it is the first instantiation).

Service Name	Location	IP address
Nova-Compute	Root@Compute05	10.20.2.30
(Empty)	Root@Controller05	10.20.2.30
Cinder-API	LXD@Controller05	10.20.2.20
Heat	LXD@Controller05	10.20.2.41
Keystone	LXD@Controller05	10.20.2.45
MySQL	LXD@Controller05	10.20.2.40
Neutron-API	LXD@Controller05	10.20.2.21
Nova-Cloud-Controller	LXD@Controller05	10.20.2.47
OpenStack-Dashboard (Horizon)	LXD@Controller05	10.20.2.43
RabbitMQ-Server	LXD@Controller05	10.20.2.27
Neutron-Gateway	Root@Network05	10.20.2.14
Cinder-Volume	Root@Storage05	10.20.2.18
Glance	LXD@Storage05	10.20.2.34

The previous IP addresses may change when a cloud or elements of the clouds are deleted and reinitiated through Juju. The most recent IP addresses can be found through the Juju Web GUI at Juju-Controller or by running the command ``juju status -m openstack5`` on the Juju node itself. Due to their volatile nature, we have not included all service IP addresses of the Development Cloud 3 and Staging Cloud 4.

## 8.2 MANO – Management and Network Orchestration

In this section we describe in more details the selected orchestrator platform being ETSI OSM (Open Source MANO[6]) in the context of the TRIANGLE testbed. To make the document self-contained, we provide some brief information about more generic aspects of MANO like architecture or deployment, however, the reader is referred to the OSM documentation for more extensive coverage of these topics.

## 8.2.1 Functions

The Management and Network Orchestration (MANO) stack allows for rapid deployment of Virtualized Network Functions (VNFs), their flexible configuration, lifecycle monitoring and decommissioning in order to automatically operate a potentially complex Network Service (NS). A model driven approach is taken to describe VNFs and NSs. MANO enhances interoperability with other components in the system such as Virtual Infrastructure Managers (VIMs) or Software-Defined Networking (SDN) controllers and is able to integrate with multiple instances and variants (solutions such as OpenStack, VMware ESXi, etc) of them.

## 8.2.2 Architecture

Open Source MANO (OSM (ETSI OSM, sd)) has an ambition to be a reference implementation of the ETSI standards. The general architecture is presented on Figure 34, extracted from [7]. Please refer to this document for the detailed architecture description.

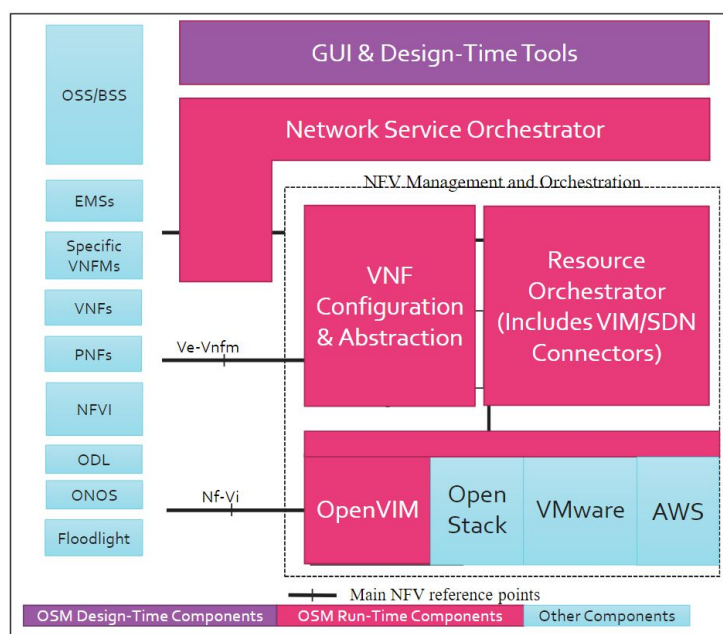


Figure 34 OSM mapping to ETSI NFV MANO

## 8.2.3 Deployment

OSM deployment on the TRIANGLE testbed was performed using the binaries for Release TWO (a stable release during execution of the TRIANGLE Open Call 1 project), following the procedure from [Section 8.1.1] [8]. A dedicated Ubuntu 16.04 Virtual Machine (OSM5, 10.20.2.44) was deployed and configured to use (non-nested) LXD containers. As a result of the installation, three LXD containers are created in the OSM host: RO (Resource Orchestrator), VCA (VNF Configuration and Abstraction), and SO-ub (hosting Service Orchestrator and the User Interface), as shown in Figure 35 (source: [8].) and Table 11.

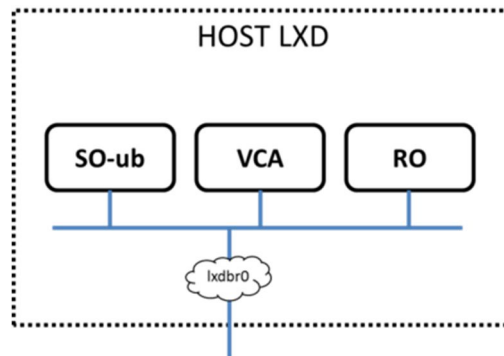


Figure 35: State of an OSM VM after OSM installation

Table 11: LXD containers after OSM installation

```
ubuntu@OSM5:~$ lxc list
```

NAME	STATE	IPV4	IPV6	TYPE	SNAPSHOTS
RO	RUNNING	10.28.33.48 (eth0)		PERSISTENT	0
SO-ub	RUNNING	10.28.33.163 (eth0)		PERSISTENT	0
VCA	RUNNING	10.44.127.1 (lxdbr0)		PERSISTENT	0
		10.28.33.151 (eth0)			

Connectivity to OpenStack was then configured, following the procedure from [Section 8.2.2] [8], with the appropriate parameter for the Keystone address being <http://10.20.2.45:5000/v2.0>

## 8.2.4 Interfaces and integration with TAP

### 8.2.4.1 RO REST interface – Resource Orchestrator

The OSM Resource Orchestrator exposes the northbound REST interface which is documented in [9]. It allows, among others, to perform actions over tenants, data-centers, instances etc.

### 8.2.4.2 SO REST interface – Service Orchestrator

The OSM Service Orchestrator exposes the northbound REST interface which is documented in [10]. It allows, among others, to perform actions such as uploading service descriptors or instantiating a Network Service. At the time of writing this document the only available version refers to OSM Release ONE while the deployed version was Release TWO. While no thorough verification was performed, we have noticed some problems in running the examples (e.g., instantiating the network service using "nsd-ref" resource fails). As a partial remedy for these kind of problems, we decided to make a small modification in OSM Client (Section 8.2.4.3) which under the hood also uses REST. The client now prints a very verbose output which includes the REST URL, request headers, *json* payload, etc. which can in turn be used to compose the REST commands used for example by TAP script (Section 8.2.4.5).





### 8.2.4.3 CLI – Command Line Interface

The command line interface client is available for OSM MANO. We will describe it in more details, along with the modifications we made to it.

#### 8.2.4.3.1 OSM client

The OSM community provided a python-based OSM Command Line Interface client. While for Release THREE it is installed by default, for Release TWO a manual installation is needed, followed by setting correct environmental variables, see both Table 12 and Table 11. The client was installed on the VM instance OSM5 (10.20.2.44) and the environmental variables were put for the convenience to *osmvars.sh* file ( usage: *source osmvars.sh* )

**Table 12: OSM Client installation, configuration and verification**

```
sudo apt install libcurl4-gnutls-dev libgnutls-dev

sudo pip install git+https://osm.etsi.org/gerrit/osm/osmclient@v2.0.2

export OSM_HOSTNAME=10.28.33.163
export OSM_SO_PORT=8008
export OSM_RO_HOSTNAME=10.28.33.48
export OSM_RO_PORT=9090

osm vim-list
+-----+-----+
| vim name      | uuid                                |
+-----+-----+
| openstack-site | 811971ae-c46a-11e7-b9ee-00163e1024a7 |
+-----+-----+
```

#### 8.2.4.3.2 OSM client modifications

OSM client can be modified to display a very verbose output, helpful in inspecting REST calls, see Table 13. While useful, please note this is just a temporary fix and a more elegant solution (e.g., adding “verbose” switch) can be developed. Furthermore, if the descriptor was created in the GUI (see Section 8.2.4.4 and Section 19.2 in Annex 5), it is likely that it contains additional “meta-information”, which is necessary for the graphical layout of the service but has no infrastructural function. This information, if appearing in the *json* file submitted with the REST POST call to start a network service will cause an error and thus has to be removed. This is not a result of our modification but rather an inconsistency in the CLI client and GUI. Our TAP plug-in verifies the existence of the meta-information field, however does not remove it if present. See Table 14 for the details, with the meta-information part marked in red.

**Table 13: OSM Client modifications**

```
git clone -b 'v2.0.2' --single-branch https://osm.etsi.org/gerrit/osm/osmclient
```



```
#in the file

~/osmclient/osmclient/common/http.py

#in the function
def _get_curl_cmd(self, endpoint):

#the following line was added to allow for verbose output
    curl_cmd.setopt(pycurl.VERBOSE, True)

#to see the json content add around line 79 the print statement

        jsondata = json.dumps(postfields_dict)
        print jsondata

#to rebuild
~/osmclient$ sudo pip install . --upgrade

#to verify

$ osm vim-list
* Trying 10.28.33.163...
* Connected to 10.28.33.163 (10.28.33.163) port 8008 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
[output omitted]
```

**Table 14: Using OSM client with GUI generated descriptor may cause an error**

```
ubuntu@OSM5:~/osmclient$ osm ns-create --ns_name rest_test13 --nsd_name nsd_3 --vim_account openstack-
site --admin_status ENABLED --ssh_keys piotr-zuraniewski-tno-nl
* Trying 10.28.33.163...

[output omitted]

* Connection #0 to host 10.28.33.163 left intact
{"nsr": [{"short-name": "rest_test13", "ssh-authorized-key": [{"key-pair-ref": "piotr-zuraniewski-tno-
nl"}], "description": "default description", "om-datacenter": "811971ae-c46a-11e7-b9ee-00163e1024a7",
"nsd": {"id": "nsd_3", "constituent-vnfd": [{"member-vnf-index": 1, "start-by-default": "true", "vnfd-
id-ref": "ubuntu_2c_2G_1iface_vnfd"}], "meta":
{"containerPositionMap": {"1": {"top": 135, "left": 435, "right": 685, "bottom": 190, "width": 250, "height": 55}, {"a8d499e9-ec0c-452b-bcb6-
6a0e8880fa67": {"top": 30, "left": 135, "right": 385, "bottom": 85, "width": 250, "height": 55}, {"v1
d-1": {"top": 300, "left": 447.5, "right": 697.5, "bottom": 338, "width": 250, "height": 38}}},
"name": "nsd_3", "vld": [{"mgmt-network": "false", "vnfd-connection-point-ref": [{"vnfd-connection-
point-ref": "eth0", "member-vnf-index-ref": 1, "vnfd-id-ref": "ubuntu_2c_2G_1iface_vnfd"}], "name":
"vld-1", "id": "vld-1"}]}, "admin-status": "ENABLED", "id": "21e7c5ac-f60e-11e7-bedf-5254009bd772",
"name": "rest_test13"]}]

[output omitted]
```





```
* Connection #0 to host 10.28.33.163 left intact
failed to create ns: rest_test13 nsd: nsd_3 result: {'error': 'Resource target or resource node not found'}
```

#### 8.2.4.4 GUI – Graphical User Interface

OSM also provides a graphical interface (called *launchpad*), in our case accessible under link <https://10.20.2.44:8443/launchpad/>

The GUI simplifies both descriptors design of descriptors as well as managing their catalogue, ssh keys deployment and network services instantiation, see Figure 36. From our experience, Launchpad GUI (service instantiate part) is however not always the best source of information in case of instantiation errors. Frequently, only “Failed” message (along with UUID of the instance) is given and for more insights MANO logs need to be inspected, see Section 8.2.6 for comments on troubleshooting.

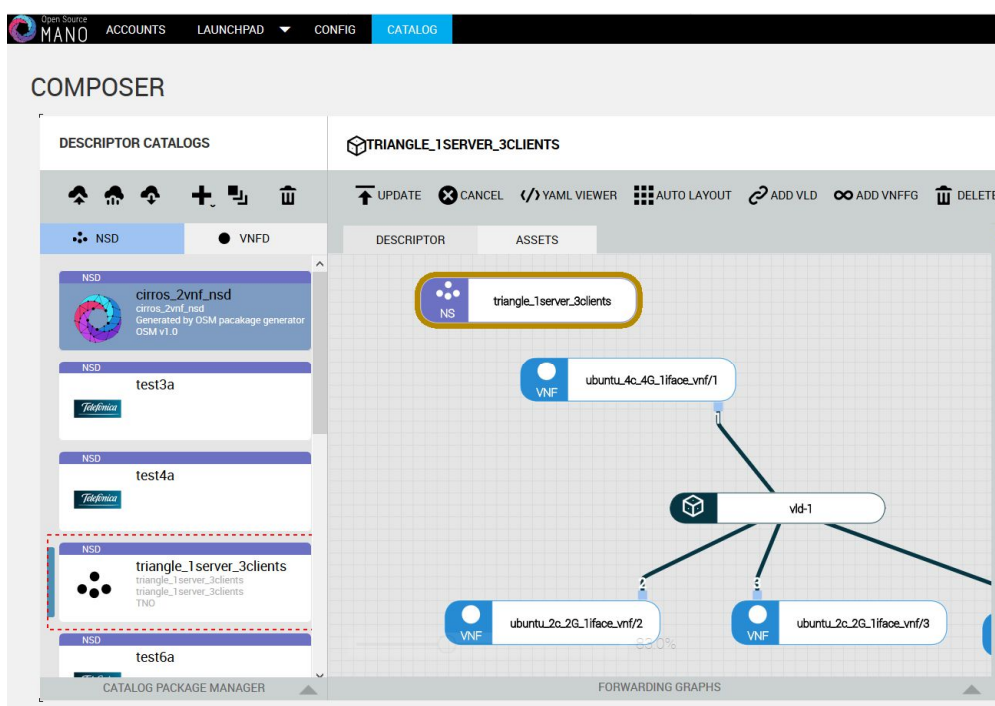


Figure 36: OSM GUI sample screenshot

#### 8.2.4.5 TAP integration

We have integrated MANO with the TAP system. The detailed description of the steps along with the sample test plan is in Deliverable 3.4. Note, due to the bug/limitation of the SSH.NET library regarding implemented HostKey algorithms it is not possible to connect to openSSH servers in Ubuntu 16.04.4 and later (ssh\_dispatch\_run\_fatal: Connection to 172.16.4.13: error in libcrypto). Modern servers offer ssh-rsa, rsa-sha2-512, rsa-sha2-256, ecdsa-sha2-nistp256,



ssh-ed25519 while SSH.NET offers only ssh-rsa and (deprecated) ssh-dss. We have tested TAP plug-in with Ubuntu 16.04.3 (earlier version should also be supported).

## 8.2.5 Instantiation

Network service instantiation is performed using the Launchpad: Instantiate page (see Figure 37)

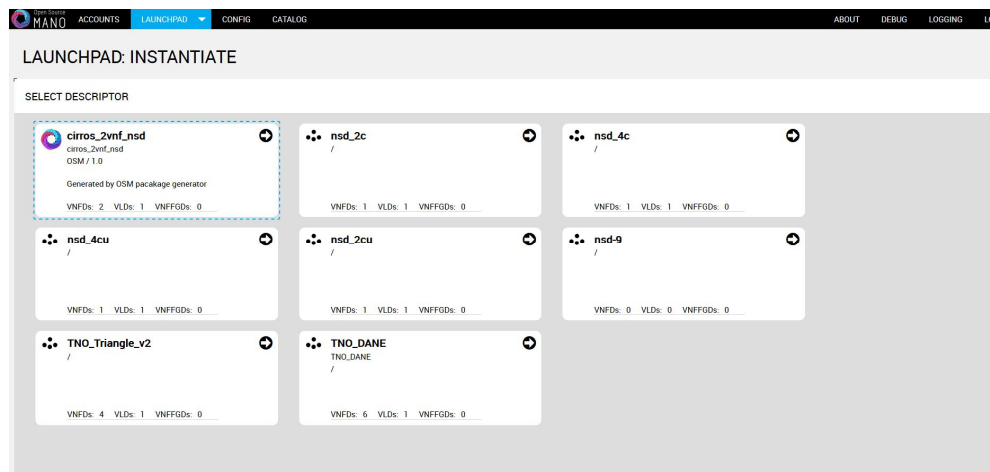


Figure 37: Instantiation

## 8.2.6 Troubleshooting

Most of the useful logs from MANO are stored on the LXD container running the given service (such as Resource Orchestrator). A good point to start with debugging is to log in to the OSM VMs (`ssh ubuntu@10.20.2.44`) and pull the Resource Orchestrator log (`lxc file pull RO/var/log/osm/openmano.log .`). The detailed description of the typical operations can be found in [11]. From the GUI, *Logging* and *Debug* sections are available, see Figure 38 and Figure 39

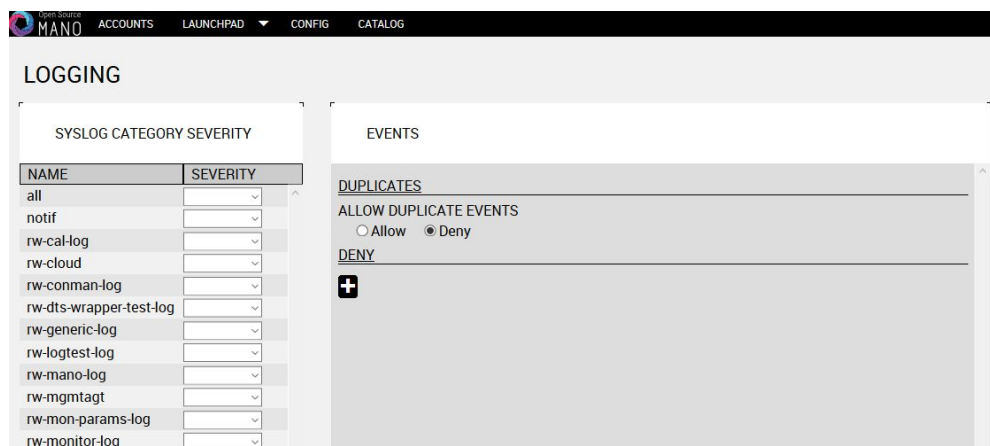
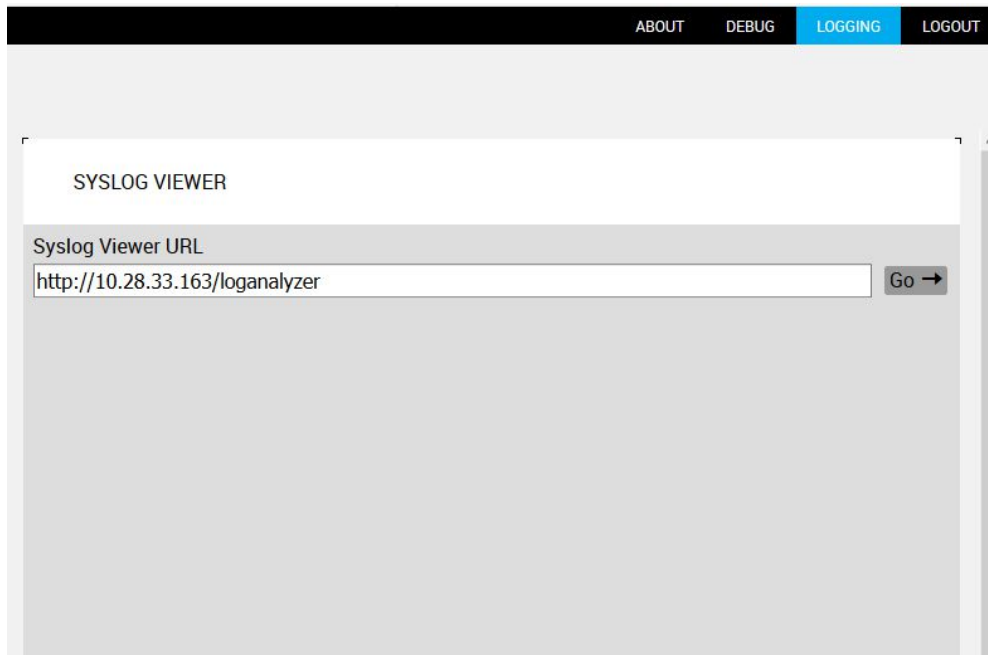


Figure 38: Logging and debug via OSM GUI



**Figure 39: Logging and debug via OSM GUI**

Annex 5 provides a sample workflow which can be used by the experimenter using MANO integrated with TAP.



## 9 UE (User Equipment and accessories)

### 9.1 Supported UEs

Table 15 shows the current status of commercial devices connected to the TRIANGLE testbed.

**Table 15 Current status of devices integrated into the testbed**

<b>Device</b>	<b>Main Ant 1</b>	<b>Main Ant 2</b>	<b>Diversity Ant 1</b>	<b>Diversity Ant 2</b>	<b>Battery</b>
<i>Samsung Galaxy S4</i>	Yes	N/A	Yes	N/A	Yes
<i>Samsung Galaxy S5 Neo</i>	Yes	N/A	No	N/A	Yes
<i>Samsung Galaxy S6</i>	Yes	N/A	Yes	N/A	Yes
<i>Samsung Galaxy S7</i>	Yes	No	Yes	No	Yes
<i>iPhone 7 Plus</i>	Contact Ant	N/A	No	N/A	No
<i>Samsung Galaxy S8</i>	Yes	No	Yes	No	Yes
<i>HTC One</i>	Contact Ant	N/A	Contact Ant	N/A	No
<i>Huawei</i>	Contact Ant	N/A	Contact Ant	N/A	No



## 10 Local applications and Servers (TNO)

### 10.1 DANE Local applications and Servers (TNO)

A DANE implementing the MPEG DASH SAND protocol based on websockets. The DANE listens for WebSocket connections from DASH video clients. Additionally, this DANE exposes a Rest API to set the bandwidth available in the network for the video clients.

#### 10.1.1 Installation

The DANE software library can be installed by issuing the following command:

```
python setup.py install
```

#### 10.1.2 Running

After installation, the DANE can be run by invoking the script installed:

```
dane
```

#### 10.1.3 SAND support

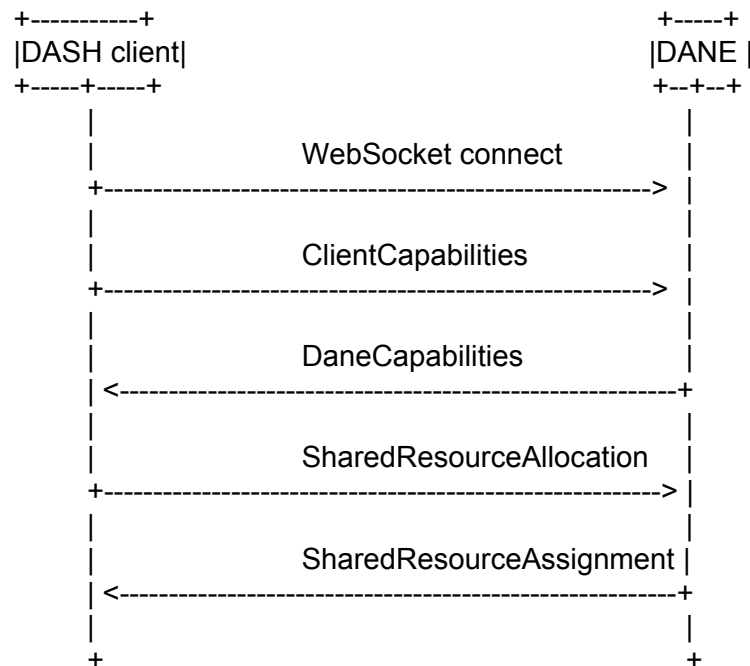
We have implemented the following messages from the MPEG-SAND protocol:

**Table 16 Messages from MPEG-SAND protocol**

<b>Device</b>	<b>Message</b>	<b>Description</b>
<i>Dane</i>	DaneCapabilities	Informs the video client on the messages supported by this DANE.
	SharedResourceAssignment	Informs the video client on the operating point (i.e. DASH representation) allocated to it by the DANE.
<i>Client</i>	ClientCapabilities	Informs the DANE on the messages supported by this video client.
	SharedResourceAllocation	Informs the DANE that this client wishes to receive resource assignment. Furthermore, it informs the DANE of the operating points (i.e. DASH representations) it supports.

#### 10.1.4 Protocol

The following diagram illustrates the operation of the SAND protocol, and the messages used to exchange bandwidth information.



### 10.1.5 DANE REST API

The DANE REST API runs on port 8088.

#### Setting the available bandwidth

```
curl -X PUT -d bw=my_bandwidth http://dane:8088/api/bandwidth
```

Sets the available bandwidth to my\_bandwidth.

Example:

```
curl -X PUT -d bw=12345 http://dane:8088/api/bandwidth
```

#### Getting the available bandwidth

```
curl -X GET http://dane:8088/api/bandwidth
```

Returns the available bandwidth as known by the DANE.

#### Resetting the DANE

```
curl -X POST http://dane:8088/api/reset
```

Resets the DANE state, i.e. disconnect all clients and reset the available bandwidth.

### 10.1.6 Dependencies

The DANE software depends on the following PyPI packages:

- pytz, for formatting datetime as requested in the SAND protocol.
- autobahn, a WebSocket library in Python.



- twisted, a networking framework needed for autobahn to work.
- requests, a HTTP requests library.

## 10.2 Web Client

### 10.2.1 Introduction

This software demonstrates the use of the DASH-SAND protocol in the context of omnidirectional video.

### 10.2.2 Configuration

The configuration file is located at `js/config.js` and allows the following to be configured:

Parameter	Description
<i>sandURI A WebSocket URI to a SAND server (e.g. a DANE).</i>	sandURI A WebSocket URI to a SAND server (e.g. a DANE).
<i>metricsURI A URI pointing to a webserver capable of logging metrics.</i>	metricsURI A URI pointing to a webserver capable of logging metrics.
<i>mpdURI A URI pointing to the MPD file to be played by the player.</i>	mpdURI A URI pointing to the MPD file to be played by the player.

### 10.2.3 Metrics Logging

Metrics can be logged to a HTTP server listening for POST requests at the following endpoints:

- `<url>/MetricsData/`, the complete dash.js metrics object will be sent here formatted as JSON.
- `<url>/QualityData/`, the quality data of the media currently playing will be sent here formatted as JSON.

In the current implementation, both metrics are logged every 5 seconds. For debugging purposes, the full dash.js logs are currently also sent to the DANE.

### 10.2.4 Building

**Preparation** Before attempting a build, make sure that all dependencies have been installed by running:

```
npm install
```

**Standalone** The project can be built using Webpack. Simply running:

```
npm run build
```

will build the project and output into the dist folder.

**Server** The Webpack development server has been configured to disable all caching through response headers. It can be run using:



`npm start`

**Docker** A Docker buildfile has been included which will create a nginx webserver container. The web server will be configured to host the web page while disabling all caching of video segments. Such a container can be built using either:

`sh build.sh`

or

`docker build -t sand .`

For features using multi-range HTTP requests, usage of this container is required.

**Changing the media content** The video can be changed by modifying the mpdURI field in the config file, and/or by modifying the contents of the video folder. While hosting the content, it can be sufficient to change the contents of the dist/video folder instead.

**Disabling DANE influence** To ignore any messages received from the DANE, add the ignoreDane (casesensitive) parameter to the URL.

With DANE:

<http://dane:8080>

Without DANE:

`http://dane:8080/?ignoreDane`

## 10.3 Metrics Database

### 10.3.1 Introduction

Webserver with API for the storage and retrieval of logs created by a dash.js client. This server demonstrates how dash.js client metrics can be logged and retrieved. Note that the storage is in-memory, and will be lost on a restart/crash.

### 10.3.2 Running

Before running, install project dependencies by running:

`npm install`

The program can be run by the following command:

`npm start`

The exposed port can be configured by setting the PORT environment variable before running the program.





### 10.3.3 Docker

The docker image can be built by invoking:

```
docker build -t metricsdb .
```

### 10.3.4 REST API

**Logging metrics data** Metrics data can be logged by making a POST request in the following way:

```
curl -X POST http://dane:8081/{uuid}/MetricsData
```

where {uuid} should be an id which uniquely identifies a client corresponding to the metrics.

**Logging quality data** Quality data can be logged by making a POST request as follows:

```
curl -X POST http://dane:8081/{uuid}/QualityData
```

where {uuid} should be an id which uniquely identifies a client corresponding to the metrics.

**Retrieving data** Data can be retrieved by making a GET request to the same endpoint as used for logging data. The client uids which are currently in the database can be obtained by making a GET request as follows:

```
curl -X GET http://dane:8081/ids
```

**Dumping data** The current state of the log can be dumped by making a POST request in the following way:

```
curl -X POST http://dane:8081/dump/{name}
```

where {name} should be a prefix of an indexed set of logs.

## 10.4 Metrics Visualisation

This service will display from a server hosting dash.js metrics at these endpoints:

- `<url>/QualityData`
- `<url>/MetricsData`

Data will be polled every second, and displayed on a graph.

## 10.5 Fake client swarm

### 10.5.1 Introduction

A simple service to launch and manage multiple SAND clients.  
These clients use a weight of 5 by default. The default port is 10260.

### 10.5.2 Commands

In order to kill all clients, a POST request should be made as follows:



```
curl -X POST http://dane:10260/reset
```

Spawning new clients can be done by issuing a PUT request in the following way:

```
curl -X PUT http://dane/spawnClients/{count}
```

where {count} should be replaced by the desired number of new clients.

## 10.6 Obtaining logs

The logging server provides an overview of all log files at its root endpoint. The URL to view the logging directory is as follows: <http://dane:10000/>

When operating, the DANE logs its communication with the clients, and provides details on its bandwidth assignment algorithm. The log file can be accessed by navigating to the log service URL. From the TAP machine, this will be located at:

<http://dane:10000/dane.log>

Logged metrics can be dumped when the Metrics database service is running. Assuming the id 'test' was used to dump a metrics database for the first time, this file can be obtained at:

<http://dane:10000/test-0.log>

## 10.7 Default port assignments

The ports on the DANE VM are assigned as follows:

Service	Sub-service	Container port	Host port
<i>DANE</i>	SAND WebSocket	9000	90000
	REST	8088	8088
<i>Fake SAND client</i>	REST	10260	10260
<i>Web client</i>	HTTP	8080	8080
<i>Metrics database</i>	REST	3000	8081
<i>Metrics viewer</i>	HTTP	80	8082
<i>Log web service</i>	HTTP	10000	10000



## 11 Extensions and new features

### 11.1 Booking system

The booking system consists in a web calendar service used by the experimenters to book the TRIANGLE Portal Testbed. The system is completed with a script that queries the current owner of the Testbed at the beginning of every hour and updates the Portal Testbed owner accordingly.

#### 11.1.1 Web calendar service

The web calendar service used is Booked Scheduler [2], an open source booking service based in LAMP that has an API to interact with it, providing the possibility to automate the process.

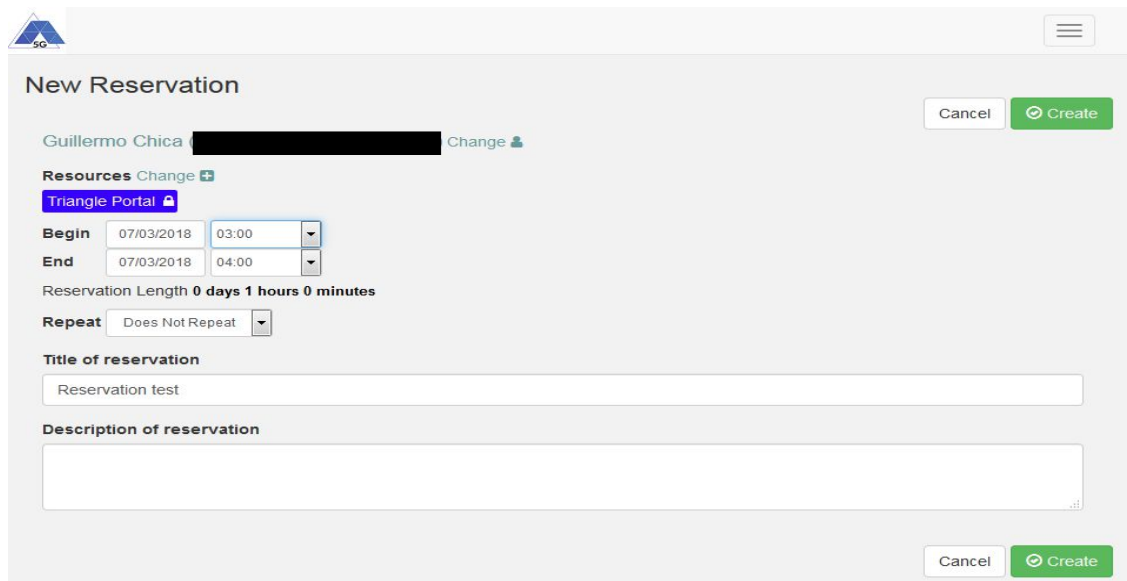
Booked Scheduler allows a user to define a resource element that will be shared by different people. This resource element will be associated with a resource schedule that will contain the resource reservations. In our case, the resource will be the TRIANGLE Portal Testbed. The capacity of this resource is one person, so only one researcher can use the Testbed at a given time. The resource schedule is divided in 1-hour slots.

The same email account used to access the Portal Testbed is used to access the web calendar service to make reservations.

There are several ways to make a reservation. It can be done using any submenu of the “Schedule” tab.

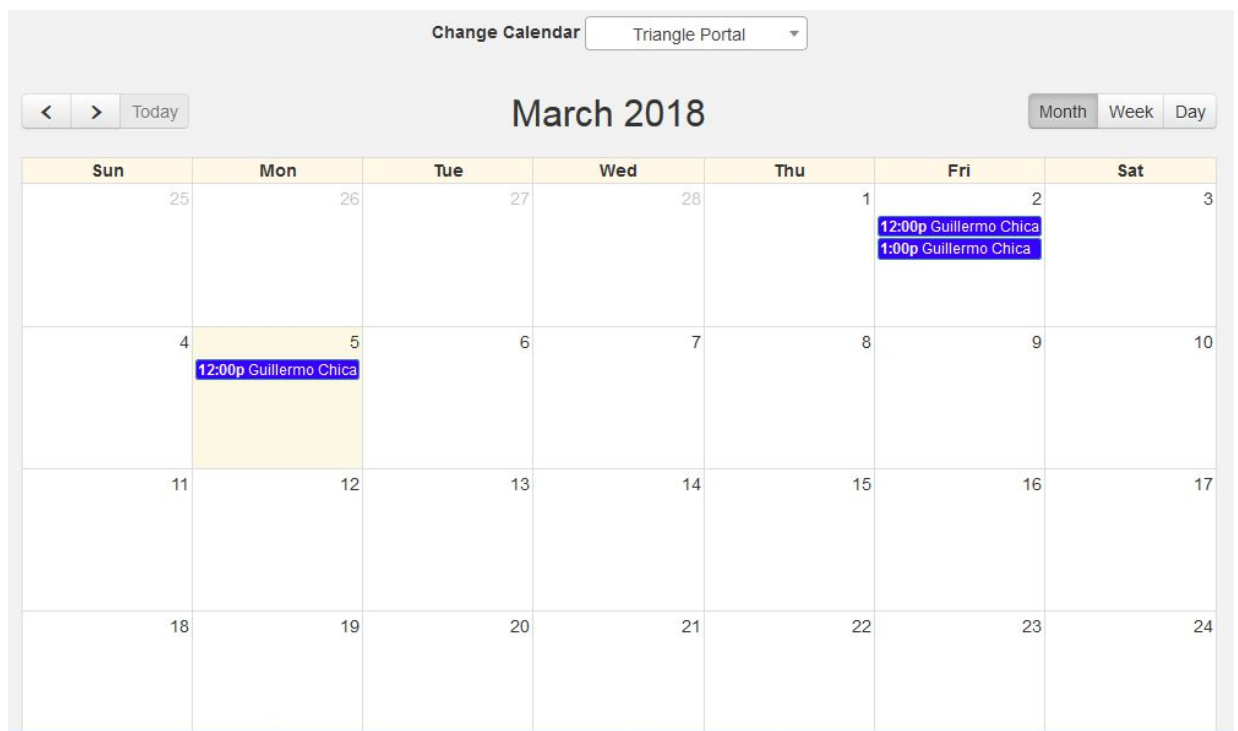
- From Bookings: Clicking on any slot from the timetable to start making a new reservation.
- From My Calendar: Clicking on any day of the calendar to start a new reservation.
- From Resource Calendar: The same as in “My calendar”.
- From Find a Time: Search a slot time and click one of the results.

Any of these methods will redirect to the “New Reservation” page. In this page, one can select the time start and end of the reservation and optionally give it a title and add comments. After clicking “Create”, the system will confirm the reservation if there are no conflicts.



The 'New Reservation' form includes a user selection dropdown (Guillermo Chica), a resource dropdown (Triangle Portal), and time selection fields for 'Begin' (07/03/2018 03:00) and 'End' (07/03/2018 04:00). It also shows a reservation length of 0 days 1 hours 0 minutes and a repeat option set to 'Does Not Repeat'. There are input fields for the title and description of the reservation, and 'Cancel' and 'Create' buttons at the bottom right.

Figure 40 The New Reservation Page allows a user to select a slot time to use the Portal Testbed



The calendar interface shows a monthly view for March 2018. It features a 'Change Calendar' dropdown set to 'Triangle Portal' and navigation buttons for previous/next month, today, and view toggles (Month, Week, Day). The calendar grid shows dates from 25 to 24. Reservations are highlighted in blue boxes: '12:00p Guillermo Chica' and '1:00p Guillermo Chica' on Friday, March 2nd, and '12:00p Guillermo Chica' on Monday, March 5th.

Figure 41 Calendar interface. It shows the reservations made. Clicking on any day will take you to the New Reservation Page.

### 11.1.2 Testbed owner updating

The web calendar service itself only provides the calendar interface to create reservations, but these reservations alone do not have impact on the TRIANGLE Portal Testbed. In order to



automatically update the testbed owner according to the reservations made in the web calendar a couple of scripts have been designed.

The first script, written in Python, will use the Booked Scheduler API to ask if there is a reservation for the current hour, and if there is one, to get the email of the person who reserved.

With this email, a second script, written in Shell, will update the Portal testbed owner using the Portal Ruby on Rails console commands.

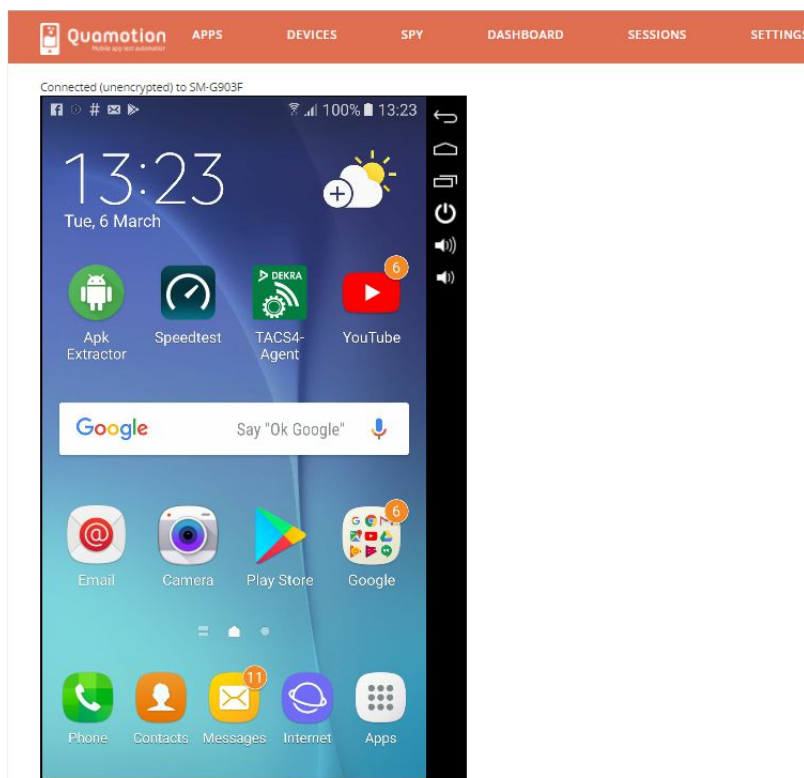
These scripts are executed at the beginning of every hour, since the web calendar is structured in 1-hour slots. This way, we provide automatic access to the TRIANGLE Portal Testbed for a researcher when he or she has booked it via the web calendar service.

## 11.2 Remote screen

Following a demand from the researchers that have been using the testbed, we now provide a remote screen option for the TRIANGLE Testbed Portal. This remote screen shows a live video of what is happening in the physical screen of the device under test using VNC technology.

To achieve this, we take advantage of the screenshotfeed feature present in the Quamotion Frontend[<http://docs.quamotion.mobi/en/latest/frontend/frontend-devices.html#open-the-screenshotfeed-for-a-device>], adding a basic authentication system to increase the security of Quamotion.

This way, the clients will have access to the full Quamotion Frontend when they reserve the TRIANGLE Portal Testbed. Only the researcher that has made a reservation for a certain time slot will have access to the Frontend during that timeslot. In the Frontend, under the “Devices” tab, they will be able to open a new Remote screen session for the device. Additionally, they will be able to view the status of the sessions started during their experiments (in the “Sessions” tab), which will give them more information about the execution of their experiment, something that has also been demanded.



**Figure 42** View of the screenshot feed feature provided by Quamotion.

### 11.2.1 Basic Authentication

As already mentioned, we have extended the Quamotion Frontend with a basic authentication system as a protection mechanism. This way, when a user tries to access the Frontend a prompt will ask for username and password, which has been previously provided by the testbed managers.

To increase the security, first we use an Nginx web server with the reverse proxy feature. This allows us to hide the existence of the Quamotion Frontend server, closing its port to the public, leaving open only the typical port 80. The Quamotion resources are returned to the client as if they originated from the Nginx web server.

Additionally, we use the Nginx Basic Authentication feature, forcing clients to use their email (the same one they use in the Testbed and Booking service) and a password to access the Quamotion Frontend. This authentication system uses a password file that stores a list of usernames and their unencrypted password. We, using user's email as seed, generate the password. This password is sent to their owners and is permanent.

The password file will only contain the username and password of the client that has reserved the testbed at the present time. For example, if there are no reservations at a given time, the password file is empty, so nobody can try to access. If there is a reservation at a given time, only the username and password of the client who did the reservation is stored in the password file at that given time, so only him or her can try to access.

The editing of the password file is achieved with a Python script that consumes the Booked API to know if there is a reservation every hour. If there is not, it cleans the password file, deleting any info that there could be inside. If there is a reservation, it gets the email of the person who



did it, generates the password, as described before, and updates the content of the password file with this information.

### 11.3 Power shell support

Docker is a software tool that let us to package an application and its dependencies inside a virtual container to be launched on demand, gaining in flexibility, portability and stability. This new deployment allows us to execute all WebDriver actions using PowerShell.

Our Quamotion Docker deployment consists in two Docker containers provided by Quamotion, running in a Windows 10 host:

- WebDriver container: This container runs the latest WebDriver version.
- Ubuntu runner container: This container is an Ubuntu system used to run PowerShell scripts. These scripts are the user flows executed in the device by WebDriver. The user flows are downloaded in this container and then executed. This way, we have a secure environment to run any user flow without affecting the host machine.

We use several PowerShell scripts to deploy and interact with the Docker containers. The first script runs the adb server in the Windows 10 host. This adb server is configured so that the containers and the host can connect and adb client to it. Then, a second script launches and configures the two containers. A third script downloads two scripts in the Ubuntu container: the runner script and the user flow generated by the researcher. The runner script creates a new WebDriver session, executes the user flow script downloaded and end the session after it. Finally, a script tears down all the containers and stop the adb server.

**Table 17 List of PowerShell scripts used in the Quamotion Docker deployment**

Script	Action	Executed in
<i>StartAdb.ps1</i>	Run adb.exe	Windows 10 host
<i>LaunchContainers.ps1</i>	Launch WebDriver and Ubuntu Containers. Add application and license to WebDriver.	Windows 10 host
<i>StopContainers.ps1</i>	Tear down the containers and stop the adb server.	Windows 10 host.
<i>RunApplicationSession.ps1</i>	Download Runner.ps1 and Userflow.ps1 in the Ubuntu container, then execute Runner.ps1.	Windows 10 host
<i>Runner.ps1</i>	Start a new WebDriver session, execute Userflow.ps1 and then end the session.	Ubuntu container
<i>Userflow.ps1</i>	Perform actions in the device.	Ubuntu container
<i>qmDocker.psm1</i>	Assemble several functions to be used by the rest of the scripts	Windows 10 host and Ubuntu container.

As the Runner.ps1 and Userflow.ps1 scripts are executed in the Ubuntu container, they need to be available on the Internet to be downloaded in it. The RunApplicationSession.ps1 script takes the URL of each script to perform the download in the Ubuntu container. Since the user



flow is uploaded to the Portal by the researcher, its URL will be served by orcomposutor to RunApplicationSession.ps1. Runner.ps1 will be always the same, so it can be placed in the Portal as static content and its URL will be passed to RunApplicationSession.ps1 as well.

The user flow script is composed by the researchers using the Spy feature of WebDriver, as before. However, the script will be now exported in PowerShell, instead of in JSON format. This is not a dramatic change since WebDriver is already able to execute user flows scripts in PowerShell.

All the scripts executed in the Windows 10 host are integrated in our TAP test plan master template, using the Run Process step. These steps substitute the previous ones used to create a new session, replay the user flow and stop the session.

**Table 18 List of PowerShell scripts used in the Quamotion Docker deployment**

Step	Action
<i>Stop Containers</i>	Execute the StopContainers.ps1 script.
<i>Start adb</i>	Execute the StartAdb.ps1 script
<i>Launch Containers</i>	Run the LaunchContainers.ps1 script.
<i>Run Session</i>	Execute the RunApplicationSession.ps1 script.

This new approach will not have an impact in the way that researchers uses the Triangle Portal. However, it allows us to start every new experiment from a clean state, since the containers can be deployed and torn down every time.

## 11.4 iOS support

The TAP plugins developed in the Release 2 for controlling devices based on iOS has been integrated into the master template which drives the execution of tests launched through the Portal. In this Release, iOS devices can be selected in the Portal during the Creation of the testing campaign.

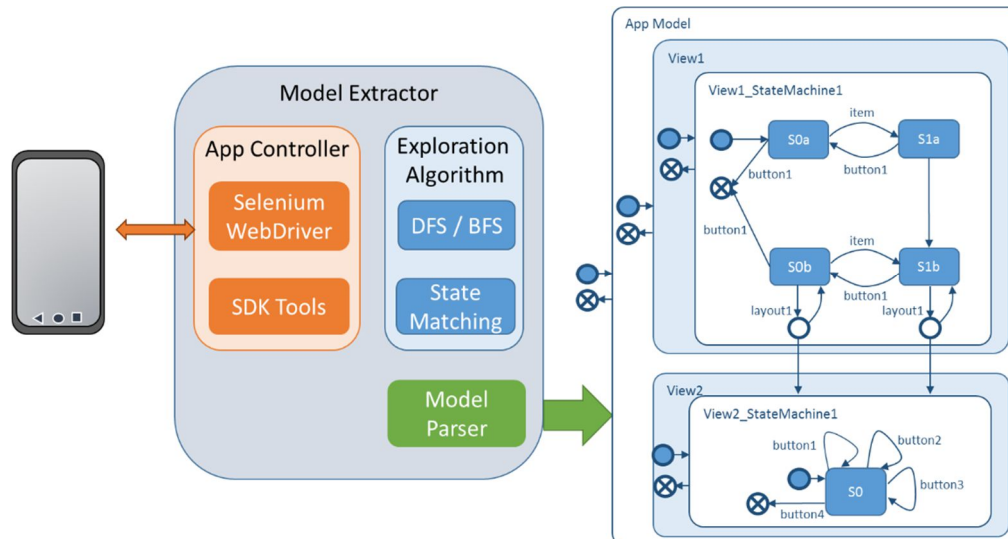
## 11.5 Model-based testing: Automatic App model extraction

In deliverable D3.2 “Progress report on the testing framework Release 2” we presented a model-based testing approach that given a model of the application under test, produces a set of app user flows that can be used during the test case execution. During the last months, we have enhanced this approach defining methodology and implementing a tool to assist the app developer to generate the application model. Although the approach is independent of the device operating system, some tasks are explained targeting Android devices.

Figure 43 shows an overview of the approach, which is based on three main elements: (i) the app controller, (ii) the exploration algorithm, and (iii) the model parser. Given the application binaries, the app controller installs and controls the execution of the target app. The app controller can perform user events, and capture the hierarchy of visual elements. The exploration algorithm decides the order in which events are performed and determines whether the app is in a visited or new state after performing such events. Finally, the model parser



transforms the states and transitions obtained from the exploration into the app model. The following sections explain each element in more detail.



**Figure 43 Model extraction overview**

### 11.5.1 App controller

The app controller interacts with the device where the application is running. The app controller is responsible for the following tasks: 1) Install, launch and close the application, 2) obtain the hierarchy of visual elements of the active view, 3) determine the list of visual elements that accept user events, 4) perform user events on specific visual elements (e.g. click, long click or scroll), and 5) fire system events (e.g. open/close keyboard or play/pause media).

This element is closely related to the device's operating system, and thus its implementation may change depending on that. Currently, we have implemented an app controller for Android devices, which is based on two testing frameworks:

Quamotion WebDriver is a test automation framework that automates iOS and Android apps on real devices. WebDriver is an open protocol for test automation originally designed for web applications based on exchange of JSON messages.

Android UIAutomator [1] is a UI testing framework included in the Android SDK. In our approach, the main task of UIAutomator is to extract the Document Object Model (DOM) of the app; i.e., to obtain the hierarchy of visual elements. For each visible element, the DOM includes a list of attributes: the resource identifier, the class, and the user events accepted. This information is especially suitable to determine if the UI has changed after performing a user event, and to obtain the list of visual elements and events that must be explored.

Mobile applications have complex UIs with multiple activities, fragments, overlays, views, etc., which accept different types of user events. The exhaustive exploration of all visual elements that react to user events can lead, in the worst case, to large models with a worse performance in the generation phase of the app user flow. To manage the size of the application model, the app controller includes the following configurable options:

- Types of events considered in the exploration. For instance, if scrolling a layout produces the same effect as clicking on a tab menu, we can ignore the scrollable views and perform only click events.



- Number of list items explored. In some apps, the effect of performing an event on a list item is the same. For instance, in a music player with a list of playable songs, clicking on each song will start playback. Thus, exploring just some of the items is enough to extract the model.
- Ignored elements. Some events in specific elements could have non-desired effects during the model extraction or the test execution. For instance, elements that restore the account password or open the configuration settings. In this way, they are systematically excluded from exploration.
- Predefined text for specific EditText fields. This is relevant in those apps whose behaviour depends on the information provided in a form, for instance apps that require logging. Observe that most of the configurable options are related to task 3, obtaining the list of visual elements that will be explored. In this way, the exploration algorithm examines a reduced number of states. In contrast, the configuration of input text for EditText fields, is required to correctly explore all desirable app behaviours.

### 11.5.2 Exploration algorithm

The exploration algorithm defines a strategy to execute user events and traverse the different app states. A state represents the app UI after performing a user event on a specific element, that is; a state is a 3-tuple (***xPath***, ***event***, ***dom***), where ***xPath*** identifies the element in the source DOM, ***event*** is the event performed on the element, and ***dom*** is the DOM after the event. Transitions between states symbolise the execution of the event on the element stored in the target state.

We have an exploration algorithm with a depth-first search strategy. The main data structures are the set of ***visited*** states, the ***path*** (list) of states that leads to the current one, and the stack of ***unvisited*** states. While there are unvisited states, the algorithm extracts one and requests that the app controller performs the user event on the visual element, and stores the resulting DOM. Then, the algorithm checks whether a state in visited has an equivalent DOM. If so, the state is considered visited, and the app model is updated with a new transition from the parent state to the visited state and backtracks. Otherwise, the state is new, and it is included in visited. In addition, the app model is updated with the new state and the transition from its parent. If the node has successors, that is visual elements that can handle user events, they are included in the stack of unvisited states. Otherwise, the algorithm backtracks to a previous state.

#### 11.5.2.1 Matching criterion

The matching criterion defines when two DOMs can be considered equivalent. If we consider that two DOMs are equivalent when they are completely equal, the number of different states explored can be very large. Thus, the objective of the matching criterion is to reduce the number of different states, abstracting away specific content of the DOMs. The criterion is defined at different levels as follows:

- Two DOMs are equivalent if they are in the same activity, have the same hierarchy (hierarchy relation and number of nodes) and their nodes are equivalent.
- Two nodes are equivalent if the following attributes are equal: resource-id, class, package, checkable, clickable, enabled, scrollable, long-clickable.
- If the node is editable, the text attribute must be also equal. A node is editable if its class is `android.widget.EditText` or inherits from this class. This rule is required to generate models of forms.

### 11.5.3 Backtracking

The exploration algorithm must backtrack when the explored state has been previously visited or when it has no successors. The backtracking process consists in finding a state in **path** (from last to first) that has at least one unexplored successor state, i.e.; there is a successor state in **unvisited**. The backtracking process leaves this state in the last position of **path**. After that, the algorithm requests that the app controller closes the app, and performs the list of user events (on their corresponding elements) included in **path**. When the backtracking process ends, the app will be ready to accept the user event of the next unexplored state.

### 11.5.4 Model parse

The model parser produces the app model that will be used to generate the app user flows. The model parser acts when new states and/or transitions are added. The app model is described with the modelling language presented in [2], which was also introduced in deliverable D3.2. The language is based on nested state machines. The higher-level state machine represents the app. It can contain one or several state machines associated with the different activities of the app. At the lower level, the state machines represent the interaction of the user with the app. Transitions represent the user events performed in specific visual elements (buttons, layouts, etc.), and the source and target states symbolise the UI before and after the user event. There is a special type of state, called the connection state, used to transit between state machines. Connection states have two outgoing unlabelled transitions: one that points to a state of the same state machine (returning state), and another to the target state machine. When a connection state is reached, the app executes the target state machine, and when the target state machine reaches the final state, the app comes back to the returning state of the source state machine. Figure 44 shows the model of the Universal Music Player app.

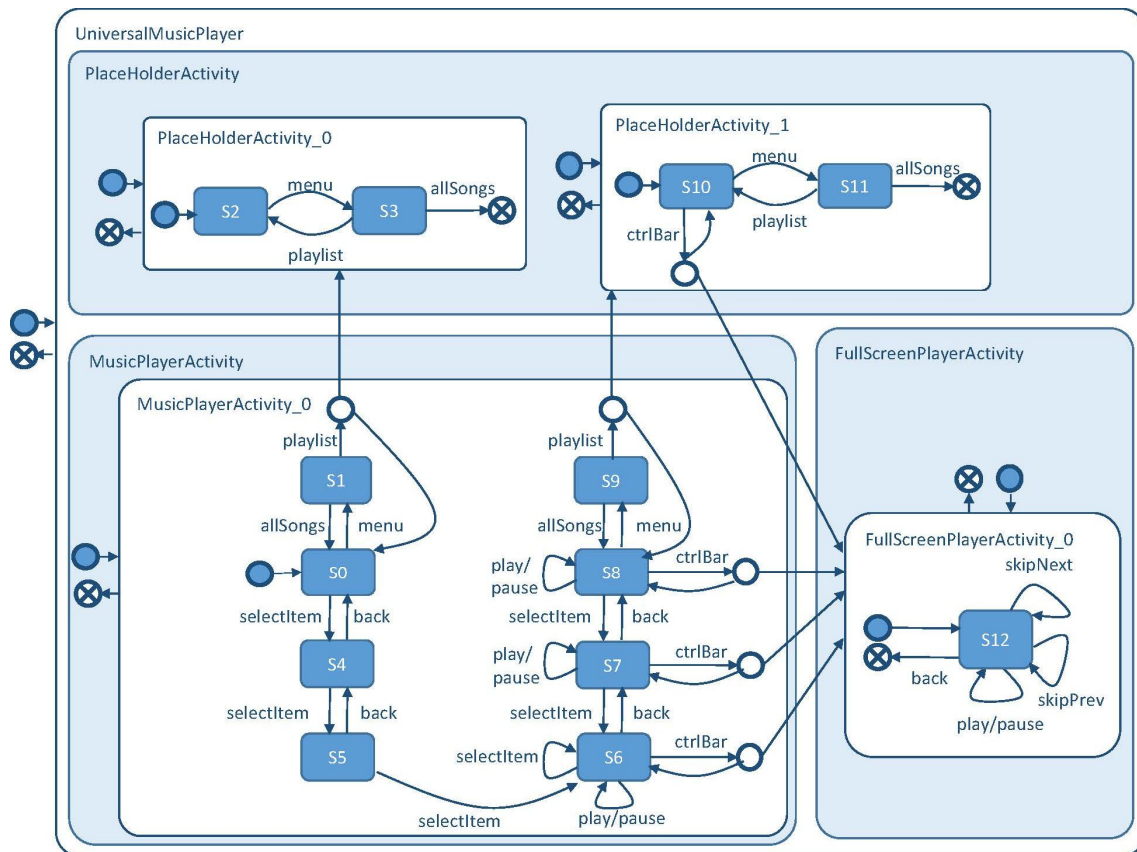


Figure 44. Universal Music Player model



In Figure 44, the app model describes the desired/undesired behaviours that the developer wishes to test. However, the app model depicts the possible interactions of the user. Although the modelling language is very expressive, the current version of the model parser has the following restrictions:

- Transitions between state machines of different views are allowed, but not transitions between state machines of the same view.
- Transitions are not labelled with temporal constraints.
- Transitions to other apps are not included in the model.
- Initial states do not represent the state of the UI. Thus, transitions from initial states are automatically fired, without any user event.

### 11.5.5 Evaluation

To evaluate the model extraction process, we have chosen five different apps from Google Play Store and Android SDK: iDo Calculator, Kolab Notes, Topeka, Universal Music Player, and WordPress. The model extraction is a configurable process, which allows us to select the maximum depth explored (number of consecutive user events), number of list items explored, and list of elements excluded. In addition, if the app requires a form to be filled in, we can provide specific text input. Table 15 shows the different configurations for the evaluated apps, and Table 20, some results.

App	max depth	list items	events	el. excluded
iDo Calc.	5	1	click	0
UAMP	10	1	click	0
Kolab Notes	8	2	click	2
Topeka	15	1	click	0
Word Press	8	1	click	7

**Table 19 Model extraction - Configuration**

iDo Calculator<sup>1</sup> is a calculator with simple and scientific modes. Text input is performed by clicking independent buttons for each digit and arithmetic operation. The resulting app model considers the click on each calculator button as a different user event. Although all these events leave the app in the same state, the number of transitions is too high to produce the app user flows later. In this situation, we recommend manually modifying the model to abstract the buttons, for example by defining two types of buttons, numeric and arithmetic, and reducing the number of transitions.

Universal Music Player (UAMP)<sup>2</sup> is a sample app included in the Android SDK. It presents a list of songs, classified by Genre that can be reproduced. From the point of view of the app model, playing any of the songs has the same effect. Thus, we have configured the model extraction to explore just the first item of each list. It is worth mentioning a limitation of the UIAutomator dump process; it cannot obtain the DOM when the UI is changing dynamically, for instance if a

<sup>1</sup> Available at <https://play.google.com/store/apps/details?id=com.ibox.calculators>

<sup>2</sup> Available at <https://github.com/googlesamples/android-UniversalMusicPlayer>



video/song is playing. Thus, the application controller has to pause media playing before obtaining the DOM.

Kolab Notes<sup>3</sup> is an app for taking notes that can be local to, or shared by different devices using an account. The model only considers the local mode. In addition, we have excluded two elements from the exploration, because these elements show a colour picker that is currently difficult to handle. With respect to the text input, we fill in the note title and content with the same text in all executions. A note can be deleted, edited, etc., and these options are shown in a list, thus, we have set the number of list items explored to 2.

Topeka<sup>4</sup> is a Google sample for playing quizzes. Quiz questions are random, some of them are answered by choosing from a possible four answers, and others by writing text. Thus, it is difficult to systematically get the right answer. In addition, this issue can produce slightly different models and complicates the automatic execution of app user flows. Therefore, the app developer must provide us with a list of questions that are asked in the same order.

WordPress<sup>5</sup> is an app for visualising and managing WordPress sites. The app has an initial login form, thus we configure the user name and password with specific input text. In addition, the app provides links to recover user name and password, create a new account or read the Terms of Service. We have excluded these links from the exploration. Another peculiarity of this app, is that it suggests sites to visit. The list of sites changes dynamically, and furthermore, the sites can include different clickable elements. This situation is similar to the dynamic questions in Topeka. However, in order to produce a first version of the model, we have limited the number of sites explored by limiting the maximum depth.

**Table 20. Model extraction - Results**

App	Activities	State machines	States	Transitions	Time (min)	App launches
iDo Calc.	2	2	9	93	49	23
UAMP	3	4	13	41	14	22
Kolab Notes	2	2	16	69	32	45
Topeka	3	3	16	51	33	38
Word Press	14	16	31	77	54	39

---

<sup>3</sup> Available at <https://play.google.com/store/apps/details?id=org.kore.kolabnotes.android>

<sup>4</sup> Available at <https://github.com/googlesamples/android-topeka>

<sup>5</sup> Available at <https://play.google.com/store/apps/details?id=org.wordpress.android>



## 12 Internal test experiment

The results obtained during the performance testing of the testbed have been included in D3.2.



## 13 TRIANGLE testbed Release 4 specifications

Release 4 is expected to be available by the 30<sup>th</sup> of September 2018. The expected new features for this release are grouped as follows:

- **User point of view**
  - TRIANGLE report in PDF format. This report will provide a top-down view to track the results in each of the domains and test cases executed.
  - Extend the number of test cases supported including new domains:
    - Reliability
    - Network resources usage
    - Network adaptation
  - New network scenarios
- **Capabilities**
  - Usage of model based testing for the automatic generation of app models. The app user flows will be generated based on this model.
  - New YAML file to provide extra information for the KPIs computation test steps and the ETL framework to provide more accurate measurements. This file will include, among others, the results obtained during the calibration of the testbed.
  - In the scope of Virtual Reality, Gaming and Augmented Reality measurement capabilities, the robotic arm platform integrated in Release 3 (section 4.2.4) supports only Android devices. In Release 4, DEKRA in tight the integration with Quamotion upgrading the platform to support iOS devices. TRIANGLE testbed will potentially have the capability of testing VR and Gaming applications on both Android and iOS devices. Full support of Augmented Reality use case will remain outside the scope of Release 4 due to the foreseen complexity of mocking the device camera.
  - Error Handling
    - Re-initialize the testbed components.
- **Testbed Access**
  - Improve utilization of the testbed through the inclusion of a mechanism to schedule efficiently the execution of the campaigns.





## 14 References

1. Android: Testing UI for Multiple Apps. <https://developer.android.com/training/testing/ui-testing/uiautomator-testing.html>
2. Espada, A.R., Gallardo, M.M., Salmeron, A., Merino, P.: Using Model Checking to Generate Test Cases for Android Applications. In: Pakulin, N., K. Petrenko, A., Schlinglo, B.H. (eds.) Proc. 10th Workshop on Model Based Testing. EPTCS, vol. 180, pp. 7{21. Open Publishing Association (2015)
3. OpenStackClient," [Online]. Available: <https://docs.openstack.org/python-openstackclient/latest/>.
4. "OpenStack API Documentation," [Online]. Available: <https://developer.openstack.org/api-guide/quick-start/>.
5. "Authentication and API request workflow," [Online]. Available: <https://developer.openstack.org/api-guide/quick-start/api-quick-start.html#authentication-and-api-request-workflow>.
6. "ETSI OSM," [Online]. Available: <https://osm.etsi.org/>.
7. C. B. (ed.), "OSM White Paper - Release TWO Technical Overview," April 2017. [Online]. Available: <https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseTWO-FINAL.pdf>.
8. "OSM Release TWO," [Online]. Available: [https://osm.etsi.org/wikipub/index.php/OSM\\_Release\\_TWO](https://osm.etsi.org/wikipub/index.php/OSM_Release_TWO).
9. "RO Northbound Interface," [Online]. Available: [https://osm.etsi.org/wikipub/index.php/RO\\_Northbound\\_Interface](https://osm.etsi.org/wikipub/index.php/RO_Northbound_Interface).
10. "SO REST API (OSM RELEASE ONE)," [Online]. Available: <https://osm.etsi.org/wikipub/images/2/24/Osm-r1-so-rest-api-guide.pdf>.
11. "Logs and troubleshooting (Release TWO)," [Online]. Available: [https://osm.etsi.org/wikipub/index.php/Logs\\_and\\_troubleshooting\\_\(Release\\_TWO\)](https://osm.etsi.org/wikipub/index.php/Logs_and_troubleshooting_(Release_TWO)).
12. "OpenStack images," [Online]. Available: <https://docs.openstack.org/image-guide/obtain-images.html>.
13. "Reference VNF and NS Descriptors (Release TWO)," [Online]. Available: [https://osm.etsi.org/wikipub/index.php/Reference\\_VNF\\_and\\_NS\\_Descriptors\\_\(Release\\_TWO\)](https://osm.etsi.org/wikipub/index.php/Reference_VNF_and_NS_Descriptors_(Release_TWO)).





## **15 Annex 1: Test cases supported in Release 3.**

### **15.1 Apps User Experience (AUE)**

#### **Common Services**

- Open the AUT (Open the App)
- Menu Navigation

#### **Content Distribution Streaming Services**

- Non Interactive Playback
- Play and Pause
- Stop and Replay
- Search and Seek
- Rewind
- Playlist Skip Forward and Backward
- Download content for offline playing
- Fast Forward

#### **Live Streaming Services**

- Play Live Video from User
- Broadcast Live Video

#### **Social Networking**

- Picture Posting
- Video Posting
- Comment Posting
- File Posting
- Show Picture
- Play Video
- File Downloading
- Play Live Video from User
- Search Object

#### **High Speed Internet**

- Downloading files sequentially
- Uploading files sequentially
- Pause and Resume Download Transfer
- Pause and Resume Upload Transfer



### **Virtual Reality**

- Virtual Experience Loading
- Virtual Scene Loading

### **Augmented Reality**

- Load Augmentation Layer on Physical Marker
- Load Augmentation Layer at Location

## **15.2 Mobile Devices Energy Consumption (DEC)**

### **Common Services**

- Device ON
- Device ON with screen OFF
- Background state

### **Content Distribution Streaming Services**

- Non Interactive Playback
- Play and Pause
- Rewind
- Download content for offline playing
- Non Interactive Playback with screen off
- Fast Forward

### **Live Streaming Services**

- Play Live Video from User
- Broadcast Live Video
- Broadcast live video with screen off

### **Social Networking**

- Picture Posting
- Video Posting
- File Posting
- Show Picture
- Play Video
- File Downloading

### **High Speed Internet**

- Downloading files sequentially
- Uploading files sequentially
- Downloading several files simultaneously



- Uploading Several Files simultaneously
- Downloading a file with screen off
- Uploading a file with screen off

### **Virtual Reality**

- Virtual Experience Loading
- Virtual Scene Loading

### **Augmented Reality**

- Load augmentation layer on a physical marker
- Load Augmentation Layer at Location
- Augmented reality session

### **Gaming**

- Start Game session
- Short Game session
- Long Game session

## **15.3 Apps Energy Consumption (AEC)**

### **Common Services**

- Device ON
- Open the app
- Background state

### **Content Distribution Streaming Services**

- Non Interactive Playback
- Play and Pause
- Rewind
- Download content for offline playing
- Non Interactive Playback with screen off
- Fast Forward

### **Live Streaming Services**

- Play Live Video from User
- Broadcast Live Video
- Broadcast live video with screen off

### **Social Networking**

- Picture Posting
- Video Posting



- File Posting
- Show Picture
- Play Video
- File Downloading

### **High Speed Internet**

- Downloading files sequentially
- Uploading files sequentially
- Downloading several files simultaneously
- Uploading Several Files simultaneously
- Downloading a file with screen off
- Uploading a file with screen off

### **Virtual Reality**

- Virtual Experience Loading
- Virtual Scene Loading

### **Augmented Reality**

- Load augmentation layer on a physical marker
- Load Augmentation Layer at Location
- Augmented reality session

### **Gaming**

- Start Game session
- Short Game session
- Long Game session

## **15.4 Mobile devices User Experience with Reference Apps Test Specification (DRA)**

### **Content Distribution Streaming Services**

- Non Interactive Playback

### **Social Networking**

- Picture Posting
- Video Posting
- Comment Posting

### **High Speed Internet**

- Downloading files sequentially
- Uploading files sequentially



## 15.5 Applications Device Resources Usage (RES)

### Common Services

- Device ON
- Open the App
- Background state

### Content Distribution Streaming Services

- Non Interactive Playback
- Play and Pause
- Rewind
- Download content for offline playing
- Non Interactive Playback with screen off
- Fast Forward

### Live Streaming Services

- Play Live Video from User
- Broadcast Live Video
- Broadcast live video with screen off

### Social Networking

- Picture Posting
- Video Posting
- File Posting
- Show Picture
- Play Video
- File Downloading

### High Speed Internet

- Downloading files sequentially
- Uploading files sequentially
- Downloading several files simultaneously
- Uploading Several Files simultaneously
- Downloading a file with screen off
- Uploading a file with screen off

### Virtual Reality

- Virtual Experience Loading
- Virtual Scene Loading



### **Augmented Reality**

- Load augmentation layer on a physical marker
- Load Augmentation Layer at Location
- Augmented reality session

### **Gaming**

- Start Game session
- Short Game session
- Long Game session



## 16 Annex 2: Measurements points (Instrumentation library)

### 16.1 Common Services

#### Login

- **App Initialization Start - Login Required**  
`eu.triangle_project.appinstr.co.Login.appInitializationStartLoginRequired()`  
Generated message:  
`Co\tLogin\tApp Initialization Start - Login Required`
- **App Initialization Start - Login Not Required**  
`eu.triangle_project.appinstr.co.Login.appInitializationStartLoginNotRequired()`  
Generated message:  
`Co\tLogin\tApp Initialization Start - Login Not Required`
- **App Started**  
`eu.triangle_project.appinstr.co.Login.appStarted()`  
Generated message:  
`Co\tLogin\tApp Started`

#### Menu Navigation

- **Start Menu Navigation**  
`eu.triangle_project.appinstr.co.MenuNavigation.startMenuNavigation()`  
Generated message:  
`Co\tNavigation\tStart Menu Navigation`
- **Menu Navigation - App Ready**  
`eu.triangle_project.appinstr.co.MenuNavigation.menuNavigationAppReady(<success>)`  
Generated message:  
`Co\tNavigation\tMenu Navigation - App Ready\t<boolean success>`

### 16.2 Content Distribution Streaming Services

#### Media File Playback

- **Media File Playback - Start**  
`eu.triangle_project.appinstr.cs.MediaFilePlayback.mediaFilePlaybackStart()`  
Generated message:  
`Cs\tPlayback\tMedia File Playback - Start`
- **Media File Playback - End**  
`eu.triangle_project.appinstr.cs.MediaFilePlayback.mediaFilePlaybackEnd()`  
Generated message:  
`Cs\tPlayback\tMedia File Playback - End`
- **Media File Playback - First Picture**  
`eu.triangle_project.appinstr.cs.MediaFilePlayback.mediaFilePlaybackFirstPicture()`  
Generated message:



Cs\tPlayback\tMedia File Playback - First Picture

- **Media File Playback - Video Resolution**

eu.triangle\_project.appinstr.cs.MediaFilePlayback.mediaFilePlaybackVideoResolution(<resolution\_x>, <resolution\_y>)

Generated message:

Cs\tPlayback\tMedia File Playback - Video Resolution\t<int resolution\_x>\t<int resolution\_y>

- **Media File Playback - Content Stall Start**

eu.triangle\_project.appinstr.cs.MediaFilePlayback.mediaFilePlaybackContentStallStart()

Generated message:

Cs\tPlayback\tMedia File Playback - Content Stall Start

- **Media File Playback - Content Stall End**

eu.triangle\_project.appinstr.cs.MediaFilePlayback.mediaFilePlaybackContentStallEnd()

Generated message:

Cs\tPlayback\tMedia File Playback - Content Stall End

## Play and Pause

- **Media File Playback - Pause**

eu.triangle\_project.appinstr.cs.PlayAndPause.mediaFilePlaybackPause(<success>)

Generated message:

Cs\tPlayPause\tMedia File Playback - Pause\t<boolean success>

- **Media File Playback - Resume**

eu.triangle\_project.appinstr.cs.PlayAndPause.mediaFilePlaybackResume(<success>)

Generated message:

Cs\tPlayPause\tMedia File Playback - Resume\t<boolean success>

## Stop and Replay

- **Media File Playback - Stop**

eu.triangle\_project.appinstr.cs.StopAndReplay.mediaFilePlaybackStop(<success>)

Generated message:

Cs\tStopReplay\tMedia File Playback - Stop\t<boolean success>

## Search and Seek

- **Media File Playback - Search**

eu.triangle\_project.appinstr.cs.SearchAndSeek.mediaFilePlaybackSearch(<success>)

Generated message:

Cs\tSearchSeek\tMedia File Playback - Search\t<boolean success>

- **Media File Playback - Seek**

eu.triangle\_project.appinstr.cs.SearchAndSeek.mediaFilePlaybackSeek(<success>)

Generated message:





Cs\tSearchSeek\tMedia File Playback - Seek\t<boolean success>

- **Media File Playback - First Search Result**

eu.triangle\_project.appinstr.cs.SearchAndSeek.mediaFilePlaybackFirstSearchResult()

Generated message:

Cs\tSearchSeek\tMedia File Playback - First Search Result

## Rewind and Fast Forward

- **Media File Playback - Rewind**

eu.triangle\_project.appinstr.cs.RewindandFastForward.mediaFilePlaybackRewind(<success>)

Generated message:

Cs\tRewindFF\tMedia File Playback - Rewind\t<boolean success>

- **Media File Playback - Fast Forward**

eu.triangle\_project.appinstr.cs.RewindandFastForward.mediaFilePlaybackFastForward(<success>)

Generated message:

Cs\tRewindFF\tMedia File Playback - Fast Forward\t<boolean success>

## Playlist Skip Forward and Backwards

- **Playlist - Skip Forward**

eu.triangle\_project.appinstr.cs.PlaylistSkipForwardandBackward.playlistSkipForward(<success>)

Generated message:

Cs\tSkipFwBw\tPlaylist - Skip Forward\t<boolean success>

- **Playlist - Skip Backwards**

eu.triangle\_project.appinstr.cs.PlaylistSkipForwardandBackward.playlistSkipBackwards(<success>)

Generated message:

Cs\tSkipFwBw\tPlaylist - Skip Backwards\t<boolean success>

## Download Media Content for Offline Playing

- **Media Content Download - Start**

eu.triangle\_project.appinstr.cs.DownloadMediaContentForOfflinePlaying.mediaContentDownloadStart()

Generated message:

Cs\tDownloadMedia\tMedia Content Download - Start

- **Media Content Download - End**

eu.triangle\_project.appinstr.cs.DownloadMediaContentForOfflinePlaying.mediaContentDownloadEnd(<success>)

Generated message: Cs\tDownloadMedia\tMedia Content Download - End\t<boolean success>



## 16.3 Live Streaming Services

### Live Video Playback

- **Live Video Playback - Start**  
`eu.triangle_project.appinstr.ls.LiveVideoPlayback.liveVideoPlaybackStart()`  
Generated message:  
`Ls\tLivePlayback\tLive Video Playback - Start`
- **Live Video Playback - End**  
`eu.triangle_project.appinstr.ls.LiveVideoPlayback.liveVideoPlaybackEnd(<success>)`  
Generated message:  
`Ls\tLivePlayback\tLive Video Playback - End\t<boolean success>`
- **Live Video Playback - First Picture**  
`eu.triangle_project.appinstr.ls.LiveVideoPlayback.liveVideoPlaybackFirstPicture()`  
Generated message:  
`Ls\tLivePlayback\tLive Video Playback - First Picture`
- **Video Resolution**  
`eu.triangle_project.appinstr.ls.LiveVideoPlayback.videoResolution(<resolution_x>, <resolution_y>)`  
Generated message:  
`Ls\tLivePlayback\tVideo Resolution\t<int resolution_x>\t<int resolution_y>`
- **Live Video Playback - Stall Start**  
`eu.triangle_project.appinstr.ls.LiveVideoPlayback.liveVideoPlaybackStallStart()`  
Generated message:  
`Ls\tLivePlayback\tLive Video Playback - Stall Start`
- **Live Video Playback - Stall End**  
`eu.triangle_project.appinstr.ls.LiveVideoPlayback.liveVideoPlaybackStallEnd()`  
Generated message:  
`Ls\tLivePlayback\tLive Video Playback - Stall End`

### Broadcast Live Video

- **Broadcast Live Video - Start**  
`eu.triangle_project.appinstr.ls.BroadcastLiveVideo.broadcastLiveVideoStart()`  
Generated message:  
`Ls\tLiveBroadcast\tBroadcast Live Video - Start`
- **Broadcast Live Video - End**  
`eu.triangle_project.appinstr.ls.BroadcastLiveVideo.broadcastLiveVideoEnd(<success>)`  
Generated message:  
`Ls\tLiveBroadcast\tBroadcast Live Video - End\t<boolean success>`
- **Broadcast Live Video - First Picture**  
`eu.triangle_project.appinstr.ls.BroadcastLiveVideo.broadcastLiveVideoFirstPicture()`



ure()

Generated message:

Ls\tLiveBroadcast\tBroadcast Live Video - First Picture

- Video Resolution

eu.triangle\_project.appinstr.ls.BroadcastLiveVideo.videoResolution(<resolution\_x>, <resolution\_y>)

Generated message:

Ls\tLiveBroadcast\tVideo Resolution\t<int resolution\_x>\t<int resolution\_y>

- Broadcast - Stall Start

eu.triangle\_project.appinstr.ls.BroadcastLiveVideo.broadcastStallStart()

Generated message:

Ls\tLiveBroadcast\tBroadcast - Stall Start

- Broadcast - Stall End

eu.triangle\_project.appinstr.ls.BroadcastLiveVideo.broadcastStallEnd()

Generated message:

Ls\tLiveBroadcast\tBroadcast - Stall End

## 16.4 Social Networking

### Post Image

- Post Image - Start

eu.triangle\_project.appinstr.sn.PostImage.postImageStart()

Generated message:

Sn\tPostImage\tPost Image - Start

- Post Image - End

eu.triangle\_project.appinstr.sn.PostImage.postImageEnd(<success>)

Generated message:

Sn\tPostImage\tPost Image - End\t<boolean success>

### Post Video

- Post Video - Start

eu.triangle\_project.appinstr.sn.PostVideo.postVideoStart()

Generated message:

Sn\tPostVideo\tPost Video - Start

- Post Video - End

eu.triangle\_project.appinstr.sn.PostVideo.postVideoEnd(<success>)

Generated message:

Sn\tPostVideo\tPost Video - End\t<boolean success>

### Post Text

- Post Text - Start

eu.triangle\_project.appinstr.sn.PostText.postTextStart()

Generated message:



Sn\tPostText\tPost Text - Start

- **Post Text - End**

eu.triangle\_project.appinstr.sn.PostText.postTextEnd(<success>)

Generated message:

Sn\tPostText\tPost Text - End\t<boolean success>

## Post File

- **Post File - Start**

eu.triangle\_project.appinstr.sn.PostFile.postFileStart()

Generated message:

Sn\tPostFile\tPost File - Start

- **Post File - End**

eu.triangle\_project.appinstr.sn.PostFile.postFileEnd(<success>)

Generated message:

Sn\tPostFile\tPost File - End\t<boolean success>

## Show Image

- **Social Networking - Image Download Start**

eu.triangle\_project.appinstr.sn.ShowImage.socialNetworkingImageDownloadStart()

Generated message:

Sn\tShowImage\tSocial Networking - Image Download Start

- **Social Networking - Image Download End**

eu.triangle\_project.appinstr.sn.ShowImage.socialNetworkingImageDownloadEnd(<success>)

Generated message:

Sn\tShowImage\tSocial Networking - Image Download End\t<boolean success>

## Play Video

- **Social Networking - Play Video Start**

eu.triangle\_project.appinstr.sn.PlayVideo.socialNetworkingPlayVideoStart()

Generated message:

Sn\tPlayVideo\tSocial Networking - Play Video Start

- **Social Networking - Play Video End**

eu.triangle\_project.appinstr.sn.PlayVideo.socialNetworkingPlayVideoEnd(<success>)

Generated message:

Sn\tPlayVideo\tSocial Networking - Play Video End\t<boolean success>

- **Social Networking - Video First Picture**

eu.triangle\_project.appinstr.sn.PlayVideo.socialNetworkingVideoFirstPicture()

Generated message:

Sn\tPlayVideo\tSocial Networking - Video First Picture



- **Social Networking - Video Resolution**

```
eu.triangle_project.appinstr.sn.PlayVideo.socialNetworkingVideoResolution(<resolution_x>, <resolution_y>)
```

**Generated message:**

```
Sn\tPlayVideo\tSocial Networking - Video Resolution\t<int resolution_x>\t<int resolution_y>
```

- **Social Networking - Video Stall Start**

```
eu.triangle_project.appinstr.sn.PlayVideo.socialNetworkingVideoStallStart()
```

**Generated message:**

```
Sn\tPlayVideo\tSocial Networking - Video Stall Start
```

- **Social Networking - Video Stall End**

```
eu.triangle_project.appinstr.sn.PlayVideo.socialNetworkingVideoStallEnd()
```

**Generated message:**

```
Sn\tPlayVideo\tSocial Networking - Video Stall End
```

## File Downloading

- **Social Networking - File Download Start**

```
eu.triangle_project.appinstr.sn.FileDownloading.socialNetworkingFileDownloadStart()
```

**Generated message:**

```
Sn\tFileDownload\tSocial Networking - File Download Start
```

- **Social Networking - File Download End**

```
eu.triangle_project.appinstr.sn.FileDownloading.socialNetworkingFileDownloadEnd(<success>)
```

**Generated message:**

```
Sn\tFileDownload\tSocial Networking - File Download End\t<boolean success>
```

## Play Live Video from User

- **Social Networking - Live Streaming Start**

```
eu.triangle_project.appinstr.sn.PlayLiveVideoFromUser.socialNetworkingLiveStreamingStart()
```

**Generated message:**

```
Sn\tPlayFromUser\tSocial Networking - Live Streaming Start
```

- **Social Networking - Live Streaming End**

```
eu.triangle_project.appinstr.sn.PlayLiveVideoFromUser.socialNetworkingLiveStreamingEnd(<success>)
```

**Generated message:**

```
Sn\tPlayFromUser\tSocial Networking - Live Streaming End\t<boolean success>
```

- **Social Networking - Live Streaming First Frame**

```
eu.triangle_project.appinstr.sn.PlayLiveVideoFromUser.socialNetworkingLiveStreamingFirstFrame()
```

**Generated message:**

```
Sn\tPlayFromUser\tSocial Networking - Live Streaming First Frame
```

- **Social Networking - Live Streaming Resolution**

```
eu.triangle_project.appinstr.sn.PlayLiveVideoFromUser.socialNetworkingLiveStreamingResolution(<resolution x>, <resolution y>)
```



**Generated message:**

Sn\tPlayFromUser\tSocial Networking - Live Streaming Resolution\t<int resolution\_x>\t<int resolution\_y>

- **Social Networking - Live Streaming Stall Start**

eu.triangle\_project.appinstr.sn.PlayLiveVideoFromUser.socialNetworkingLiveStreamingStallStart()

**Generated message:**

Sn\tPlayFromUser\tSocial Networking - Live Streaming Stall Start

- **Social Networking - Live Streaming Stall End**

eu.triangle\_project.appinstr.sn.PlayLiveVideoFromUser.socialNetworkingLiveStreamingStallEnd()

**Generated message:**

Sn\tPlayFromUser\tSocial Networking - Live Streaming Stall End

## Search Object

- **Social Networking - Search Start**

eu.triangle\_project.appinstr.sn.SearchObject.socialNetworkingSearchStart(<success>)

**Generated message:**

Sn\tSearch\tSocial Networking - Search Start\t<boolean success>

- **Social Networking - Search First Result**

eu.triangle\_project.appinstr.sn.SearchObject.socialNetworkingSearchFirstResult()

**Generated message:**

Sn\tSearch\tSocial Networking - Search First Result

## 16.5 High Speed Internet

### File Download

- **File Download - Start**

eu.triangle\_project.appinstr.hs.FileDownload.fileDownloadStart(<transfer\_id>)

**Generated message:**

Hs\tDownload\tFile Download - Start\t<int transfer\_id>

- **File Download - End**

eu.triangle\_project.appinstr.hs.FileDownload.fileDownloadEnd(<transfer\_id>, <success>)

**Generated message:**

Hs\tDownload\tFile Download - End\t<int transfer\_id>\t<boolean success>

### File Upload

- **File Upload - Start**

eu.triangle\_project.appinstr.hs.FileUpload.fileUploadStart(<success>)

**Generated message:**

Hs\tUpload\tFile Upload - Start\t<boolean success>



- File Upload - End

```
eu.triangle_project.appinstr.hs.FileUpload.fileUploadEnd(<transfer_id>,  
<success>)
```

Generated message:

```
Hs\tUpload\tFile Upload - End\t<int transfer_id>\t<boolean success>
```

## Pause and Resume Download

- File Download - Pause

```
eu.triangle_project.appinstr.hs.PauseandResumeDownload.fileDownloadPause(<success>)
```

Generated message:

```
Hs\tDownloadPause\tFile Download - Pause\t<boolean success>
```

- File Download - Resume

```
eu.triangle_project.appinstr.hs.PauseandResumeDownload.fileDownloadResume(<success>)
```

Generated message:

```
Hs\tDownloadPause\tFile Download - Resume\t<boolean success>
```

## Pause and Resume Upload

- File Upload - Pause

```
eu.triangle_project.appinstr.hs.PauseandResumeUpload.fileUploadPause(<success>)
```

Generated message:

```
Hs\tUploadPause\tFile Upload - Pause\t<boolean success>
```

- File Upload - Resume

```
eu.triangle_project.appinstr.hs.PauseandResumeUpload.fileUploadResume(<success>)
```

Generated message:

```
Hs\tUploadPause\tFile Upload - Resume\t<boolean success>
```

## 16.6 Virtual Reality

### Virtual Reality Session

- Scenario Selected

```
eu.triangle_project.appinstr.vr.VirtualRealitySession.scenarioSelected()
```

Generated message:

```
Vr\tVrSession\tScenario Selected
```

- 3D Visual Context Loaded

```
eu.triangle_project.appinstr.vr.VirtualRealitySession._3DVisualContextLoaded()
```

Generated message:

```
Vr\tVrSession\t3D Visual Context Loaded
```

- Immersion Session Started

```
eu.triangle_project.appinstr.vr.VirtualRealitySession.immersionSessionStarted()
```

Generated message:



Vr\tVrSession\tImmersion Session Started

- Immersion Session Ended

eu.triangle\_project.appinstr.vr.VirtualRealitySession.immersionSessionEnded(<success>)

Generated message:

Vr\tVrSession\tImmersion Session Ended\t<boolean success>

- Immersion Session Resolution

eu.triangle\_project.appinstr.vr.VirtualRealitySession.immersionSessionResolution()

Generated message:

Vr\tVrSession\tImmersion Session Resolution

## 16.7 Augmented Reality

### Augmented Reality Session

- Aim To Physical Marker

eu.triangle\_project.appinstr.ar.AugmentedRealitySession.aimToPhysicalMarker()

Generated message:

Ar\tArSession\tAim To Physical Marker

- Aim at Location

eu.triangle\_project.appinstr.ar.AugmentedRealitySession.aimAtLocation()

Generated message:

Ar\tArSession\tAim at Location

- Virtual Layer Displayed

eu.triangle\_project.appinstr.ar.AugmentedRealitySession.virtualLayerDisplayed()

Generated message:

Ar\tArSession\tVirtual Layer Displayed

- Augmentation Session Started

eu.triangle\_project.appinstr.ar.AugmentedRealitySession.augmentationSessionStarted()

Generated message:

Ar\tArSession\tAugmentation Session Started

- Augmentation Session Ended

eu.triangle\_project.appinstr.ar.AugmentedRealitySession.augmentationSessionEnded(<success>)

Generated message:

Ar\tArSession\tAugmentation Session Ended\t<boolean success>

- Clear Augmentation Layer - Start

eu.triangle\_project.appinstr.ar.AugmentedRealitySession.clearAugmentationLayerStart()

Generated message:

Ar\tArSession\tClear Augmentation Layer - Start





- Clear Augmentation Layer - End

```
eu.triangle_project.appinstr.ar.AugmentedRealitySession.clearAugmentationLayerEnd(<success>)
```

Generated message:

```
Ar\tArSession\tClear Augmentation Layer - End\t<boolean success>
```

## 16.8 Gaming

### Game Session

- Game Session Start

```
eu.triangle_project.appinstr.ga.GameSession.gameSessionStart()
```

Generated message:

```
Ga\tGameSession\tGame Session Start
```

- Game Started

```
eu.triangle_project.appinstr.ga.GameSession.gameStarted()
```

Generated message:

```
Ga\tGameSession\tGame Started
```

- Game Session End

```
eu.triangle_project.appinstr.ga.GameSession.gameSessionEnd(<success>)
```

Generated message:

```
Ga\tGameSession\tGame Session End\t<boolean success>
```

- Game Content Stall Start

```
eu.triangle_project.appinstr.ga.GameSession.gameContentStallStart()
```

Generated message:

```
Ga\tGameSession\tGame Content Stall Start
```

- Game Content Stall End

```
eu.triangle_project.appinstr.ga.GameSession.gameContentStallEnd()
```

Generated message:

```
Ga\tGameSession\tGame Content Stall End
```

- Game Video Resolution

```
eu.triangle_project.appinstr.ga.GameSession.gameVideoResolution()
```

Generated message:

```
Ga\tGameSession\tGame Video Resolution
```

### Pause and Resume

- Game Pause

```
eu.triangle_project.appinstr.ga.PauseandResume.gamePause(<success>)
```

Generated message:

```
Ga\tGamePause\tGame Pause\t<boolean success>
```

- Game Resume

```
eu.triangle_project.appinstr.ga.PauseandResume.gameResume(<success>)
```

Generated message:



Ga\tGamePause\tGame Resume\t<boolean success>

## Saved Game Session

- **Saved Game Load Start**

eu.triangle\_project.appinstr.ga.StartSavedGameSession.savedGameLoadStart()

**Generated message:**

Ga\tSavedGame\tSaved Game Load Start



## 17 Annex 3: Robotic Arm Remote Control Interface

This annex contains the details about the remote-control interface exposed by the robotic arm platform. This functionality was implemented in Release 2 of the testbed and it is further enhanced in Release 3 via a customized TAP plugin. This platform aims at the measurement capabilities derived from the Virtual Reality, Gaming and Augmented Reality use cases.

### SET WRITE MODE

Sets the capture mode.

- **ON:** It saves all the captured screenshots to disk. Use this mode to obtain the target files for testing.
- **OFF:** It does not save the captured screenshots to disk. Use this mode for testing.

#### Syntax

```
APP:SETWRITEMODE mode
```

#### Parameters

Name	Type	Possible values
Mode	string	{on, off}

### RESET

Check that the robotic arm is operational and then it sets yaw, pitch and roll to 0.

#### Syntax

```
ARM:RESET
```

#### Parameters

None.



## ROLL

Roll the arm to a given position at a given speed.

### Syntax

```
ARM:ROLL value, speed
```

### Parameters

Name	Type	Possible values
value	integer	-90, 90
speed	enum	VERY_LOW, LOW, MEDIUM, HIGH

## PITCH

Pitch the arm to a given position at a given speed.

### Syntax

```
ARM:PICTH value, speed
```

### Parameters

Name	Type	Possible values
value	integer	-90, 90
speed	enum	VERY_LOW, LOW, MEDIUM, HIGH

## YAW

Yaw the arm to a given position at a given speed.

### Syntax

```
ARM:YAW value, speed
```

### Parameters

Name	Type	Possible values
value	integer	-180, 180



speed	enum	VERY_LOW, LOW, MEDIUM, HIGH
-------	------	-----------------------------

## START SCREEN CAPTURE

Starts screen mirroring engine.

Invoke this function before any other function from Device Interface group.

### Syntax

**OBJECT : START** orientation

### Parameters

Name	Type	Possible values
orientation	integer	{VERTICAL, HORIZONTAL}

## STOP SCREEN CAPTURE

Stops screen mirroring engine.

### Syntax

**OBJECT : STOP**

### Parameters

None

## TAP AT LENS CENTER

Taps at the lens (VR view) center.

This function does not provoke any movement of the arm.

### Syntax

**OBJECT : TAPATLENSCENTER**

### Parameters

None.



## FIND AND TAP AT OBJECT

Finds one object and tap at its center.

This function does not provoke any movement of the arm.

### Syntax

```
OBJECT:FINDANDTAP delay, number, objects
```

### Parameters

Name	Type	Possible values
delay	double	{0, 60}
number	integer	{1, 10}
objects	string	Target image file name. Size of "number".

## FIND AND SWIPE AT OBJECT

Finds one object from the device screen and swipe at its center.

This function does not provoke any movement of the arm.

### Syntax

```
OBJECT:FINDANDSWIPE delay, number, objects
```

### Parameters

Name	Type	Possible values
Delay	double	{0, 60}
Number	integer	{1, 10}
objects	string	Target image file name. Size of "number".



## MOVE FIND AND TAP AT OBJECT

Moves the robotic arm until the device screen shows one object, and then it taps at the object center.

### Syntax

```
OBJECT:MOVEFINDANDTAP delay, number, objects
```

### Parameters

Name	Type	Possible values
delay	double	{0, 60}
number	integer	{1, 10}
objects	string	Target image file name. Size of "number".

## MOVE FIND AND AIM AT OBJECT

Moves the robotic arm until the device screen shows one object, and then it moves the robotic arm until the object gets at the center of the lens aim.

### Syntax

```
OBJECT:MOVEFINDANDAIM delay, number, objects
```

### Parameters

Name	Type	Possible values
delay	double	{0, 60}
number	integer	{1, 10}
objects	string	Target image file name. Size of "number".



## FIND AND MEASURE

Measures the time until finds one object in the device screen. This function implements the performance indicator “time to load an object”.

This function does not provoke any movement of the arm.

### *Syntax*

```
OBJECT:FINDANDMEASURE number, objects
```

### *Parameters*

Name	Type	Possible values
number	integer	{1, 10}
objects	string	Target image file name. Size of “number”.





## 18 Annex 4: OpenStack API access

In the URLs, the service names need to be replaced with IP addresses from section 8.1.6.

*Identity API:* http://<Keystone>:5000/v3

The Identity service provided by the Keystone package provides authentication tokens that are used by the other REST APIs to authenticate and authorize the client. A client first needs to authenticate itself and the Identity Service and request an authentication token, this authentication token can then be used for a period of time to make follow-up requests to the other REST APIs. The exact process is described in [5], though summarizes to the following actions

1. Make an HTTP POST request to the Identity API service containing the headers and data from Table 21, with variables filled in from Table 10:

**Table 21: Identity Service API Request Token**

```
Content-Type: application/json
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "domain": {
            "name": "$OS_USER_DOMAIN_NAME"
          },
          "name": "$OS_USERNAME",
          "password": "$OS_PASSWORD"
        }
      }
    },
    "scope": {
      "project": {
        "domain": {
          "name": "$OS_PROJECT_DOMAIN_NAME"
        },
        "name": "$OS_PROJECT_NAME"
      }
    }
  }
}
```

2. In response, if authentication succeeds, the Identity API will respond similar to Table 22, where HTTP 201 created indicates a successful authentication, the HTTP Header *X-Subject-Token* indicated the token, and response DATA provides further information about the session and user account.

**Table 22: Identity API Service Authentication Token Response**

HTTP 201 Created
------------------



```
X-Subject-Token: d55fa31bec30448386e34e082aa6e0fe
{
  "token": {
    "is_domain": false,
    "methods": [
      "password"
    ],
    "roles": [
      {
        "id": "9fe2ff9ee4384b1894a90878d3e92bab",
        "name": "_member_"
      },
      {
        "id": "86599d047b944cf4ac64dc5b676518a0",
        "name": "Admin"
      }
    ],
    "expires_at": "2017-12-13T15:24:39.000000Z",
    "project": {
      "domain": {
        "id": "default",
        "name": "Default"
      },
      "id": "086a8aacd0bc4aaebdb87ba3f8e8a556",
      "name": "admin"
    },
    "user": {
      "password_expires_at": null,
      "domain": {
        "id": "default",
        "name": "Default"
      },
      "id": "056b7395f54c486a85e130f4c69bdc74",
      "name": "admin"
    },
    "audit_ids": [
      "aZRwAlMBRGa_uhjOugT__Q"
    ],
    "issued_at": "2017-12-13T14:24:39.000000Z"
  }
}
```



3. Store the token returned in HTTP Header *X-Subject-Token* into the variable `$OS_TOKEN` and include it in the HTTP Header *X-Auth-Token* in future requests to the other REST APIs. For example, using *curl* you can request an overview of all other available APIs from the Identity service: ``curl -v -H "X-Auth-Token: $OS_TOKEN" $OS_AUTH_URL/auth/catalog | python -m json.tool``. Additionally, you may need the project-id of the OpenStack project you're authenticated to which is stored in the JSON-encapsulated response in the property *token.project.id* and store it into `$OS_PROJECT_ID`.

Using this token, the following APIs can be accessed (replace service names with IP addresses from Section 8.1.6 and use the appropriate `$OS_PROJECT_ID` from a project you are authorized on):

Name	Service Type	URL
Cinder V3	Volume	<code>http://&lt;Cinder-API&gt;:8776/v3/&lt;\$OS_PROJECT_ID&gt;</code>
Cinder V2	Volume	<code>http://&lt;Cinder-API&gt;:8776/v2/&lt;\$OS_PROJECT_ID&gt;</code>
Glance	Image	<code>http://&lt;Glance&gt;:9292</code>
Nova	Compute	<code>http://&lt;Nova-Cloud-Controller&gt;:8774/v2/&lt;\$OS_PROJECT_ID&gt;</code>
Keystone	Identity	<code>http://&lt;Keystone&gt;:5000/v2.0</code>
Neutron	Network	<code>http://&lt;Neutron-API&gt;:9696</code>
Placement (Nova)	Compute Placement	<code>http://&lt;Nova-Cloud-Controller&gt;:8778</code>
Heat	Orchestration <sup>6</sup>	<code>http://&lt;Heat&gt;:8004/v1/&lt;\$OS_PROJECT_ID&gt;</code>
Heat-CFN	CloudFormation-compatible Orchestration	<code>http://&lt;Heat&gt;:8000/v1</code>

The OpenStack API Documentation (OpenStack API Documentation, sd) give a full description of all APIs and their exact commands.

<sup>6</sup> Note: a different solution, i.e., Open Source Mano, is used as the orchestrator.



## 19 Annex 5: Sample work-flow

In this annex we will present a sample workflow which can be used by the experimenter using MANO integrated with TAP. We will show steps needed to create a basic TRIANGLE topology and perform a basic connectivity test. We assume the experimenter is using the infrastructure (“Stable Cloud5”) described in the previous sections of this document.

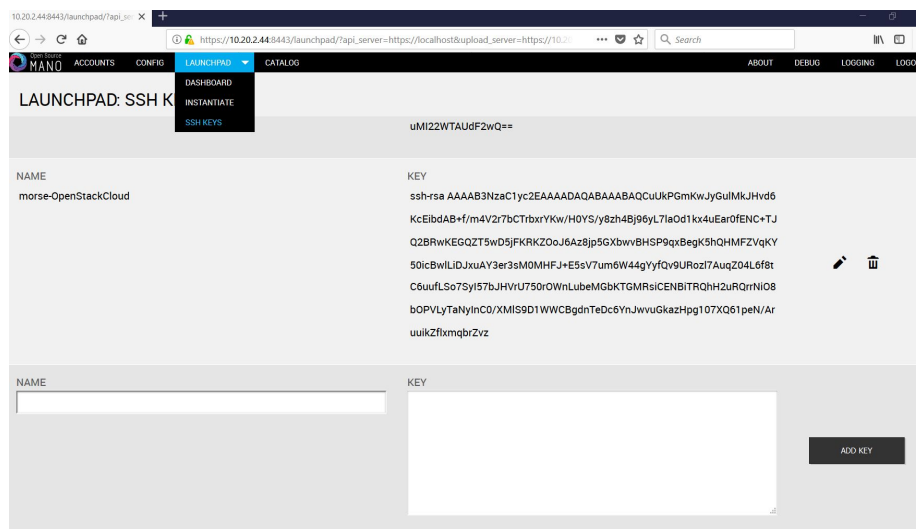
### 19.1 Bootstrapping environment

We will describe the one-time-effort actions required to bootstrap the environment for a particular experiment.

#### 19.1.1 Key deployment

Most of the VM images which are to be used in OpenStack environment use the *cloud-init* mechanism to preconfigure the images, among others, ssh keys are injected to the instance for authentication. Frequently, password logins are disabled due to security reasons and the only way to access an instance is to have public-private key pairs deployed. MANO allows for automation of this process but first needs to have the public key(s) stored.

1. Open MANO GUI at <https://10.20.2.44:8443/launchpad> and log in (default credentials are admin/admin, though these may be changed)
2. Navigate to *Launchpad*->*SSH keys* menu
3. Insert a public key and give it a name which will be used in further steps to identify the specific user’s identity or machine’s deployment key.



#### 19.1.2 Image deployment

Virtual Machines instantiated by MANO are based on the images managed by OpenStack Glance service. The delivered Stable Cloud5 is pre-populated with common images, however, the experimenters may need to upload their specific images.



1. Download (or prepare) an image source file. The sources for popular images are listed here: 12.
2. Log in to the OpenStack Dashboard at <http://10.20.2.43> ; the default credentials are admin/admin though may be changed.
3. Navigate to Project -> Compute -> Images and press “Create Image” button, see Figure 45.
4. Give an image a name (it will be used to identify it by MANO later on), choose the source file (downloaded or prepared), configure the right format and give RAM and disk parameters (observe, that RAM is given in MB while the disk size is given in GB) and press the “Create Image” button, see Figure 46

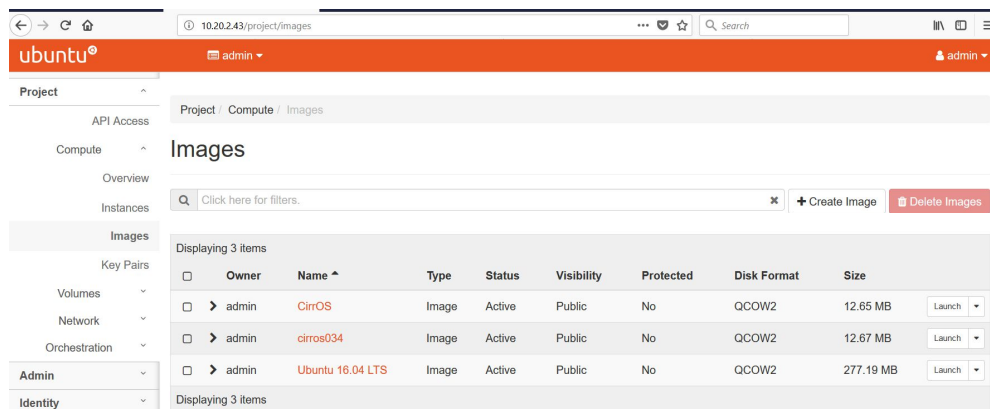


Figure 45: Image creation (part 1)

## 19.2 Preparing descriptors

MANO uses Descriptor packages for Virtual Network Functions (VNFDs) and Network Services (NSDs). In their most basic form, these are *yaml* configuration files archived and compressed through *tar* and *gzip*, possibly accompanied by some other auxiliary files (e.g., icons). A collection of sample descriptors can be found here: [13]. We will now demonstrate how to prepare a basic TRIANGLE topology which constitutes of three client machines and one server, connected to the same network.

### 19.2.1 Preparing Virtual Network Function Descriptor

1. Open MANO GUI at <https://10.20.2.44:8443/> and log in (default credentials are admin/admin)
2. Navigate to the *Catalog* menu and press “+ Add VNFD” button. A new VNF and VDU (Virtual Deployment Unit) appear, see Figure 47
3. Click on the VNF in the main part of the screen and change its name and (after pressing “more” in the right part of the screen) its ID, see Figure 48. After pressing “Update” a new VNFD (with its newly chosen name) will appear on the right hand side due to selecting a new ID. Make sure you continue with configuring this specific VNFD. An old VNFD (vnfd-1) can be deleted.
4. Scroll down to “Connection Point” (“more” must be pressed earlier) and rename it to “eth0” (Figure 49)
5. Click on VDU in the middle of a screen. Adjust its Name, VM flavor (make sure the values here meet the minimal requirements for a given image, see Section 19.1.2), Image name (make sure it exists in OpenStack), External Interface (changed in previous bullet) and ID, see Figure 50. Press Update to save the changes.
6. Return to the VNF config and adjust connectivity, as VDU id has just been changed (Figure 51)
7. For the reference, the *yaml* configuration file containing the resulting descriptors is provided in Table 23



Image Details

Metadata

Image Details

Specify an image to upload to the Image Service.

Image Name\*

centos7-1711

Image Description

Image Source

Source Type

File

File\*

Browse...

CentOS-7-x86\_64-Generic

Format\*

QCOW2 - QEMU Emulator

Image Requirements

Kernel

Choose an image

Ramdisk

Choose an image

Architecture

Minimum Disk (GB)

20

Minimum RAM (MB)

2048

Image Sharing

Visibility

Public

Private

Protected

Yes

No

✕ Cancel

< Back

Next >

✓ Create Image

Figure 46: Image creation (part 2)

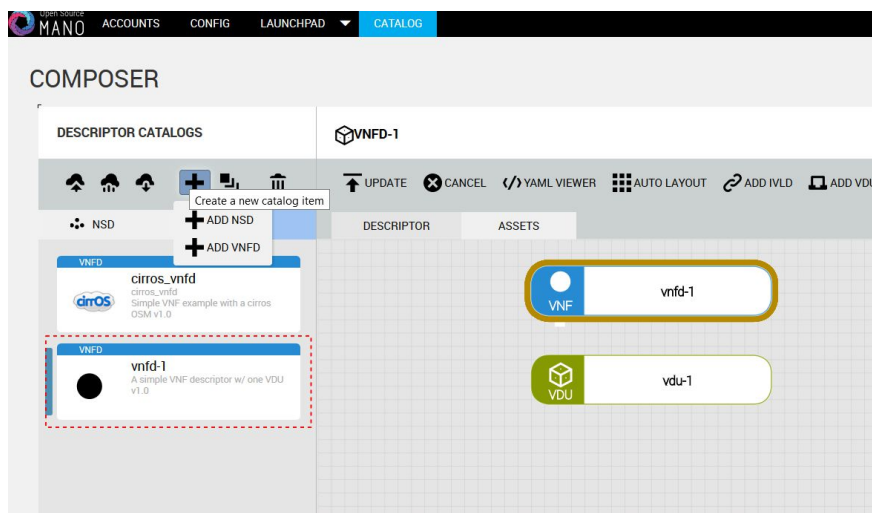


Figure 47: VNFD creation (adding VNDF 1)

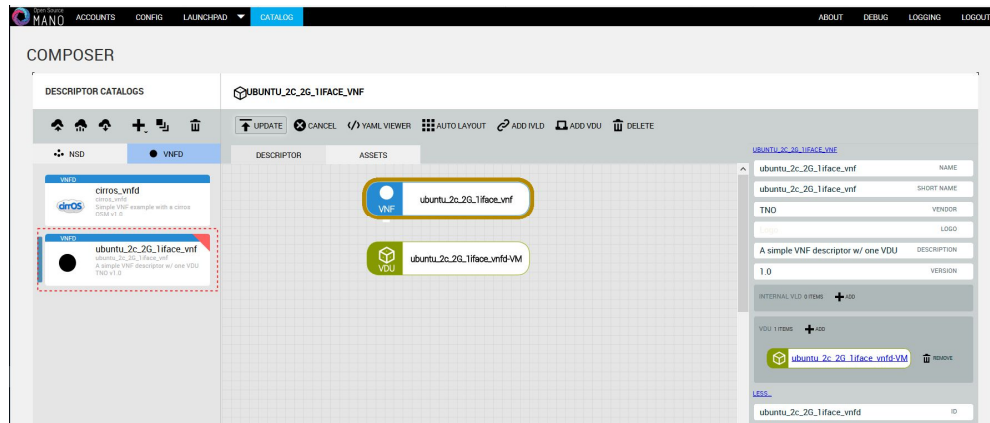


Figure 48: VNFD creation (name and ID)

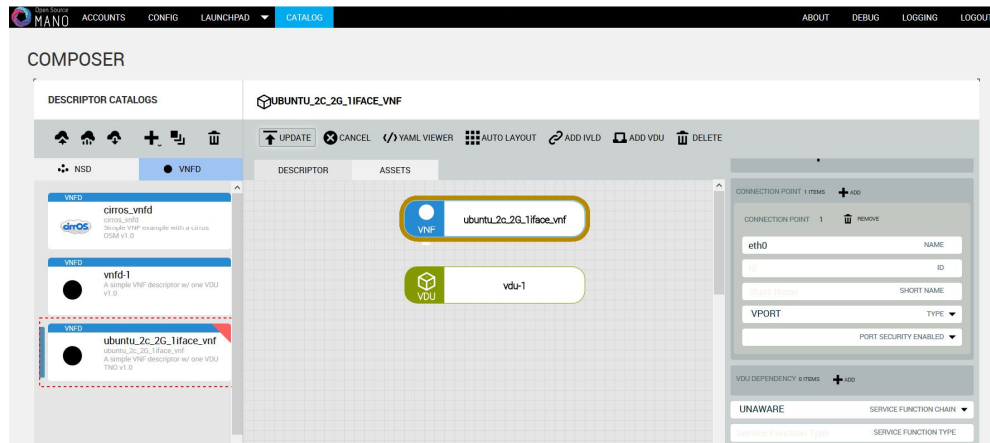


Figure 49: VNFD creation (connectivity)

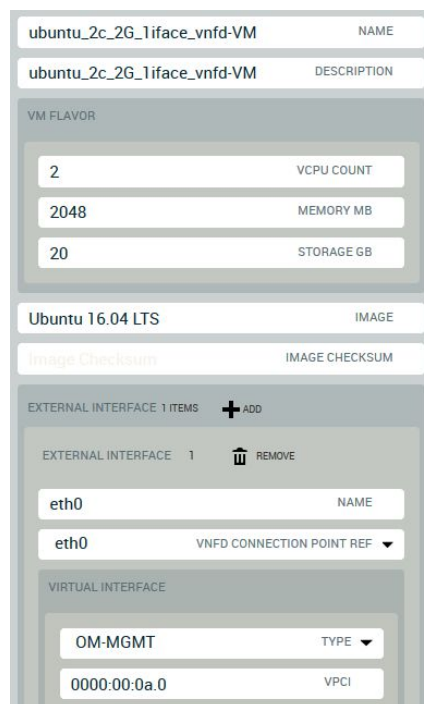


Figure 50: VNFD creation (VDU details)

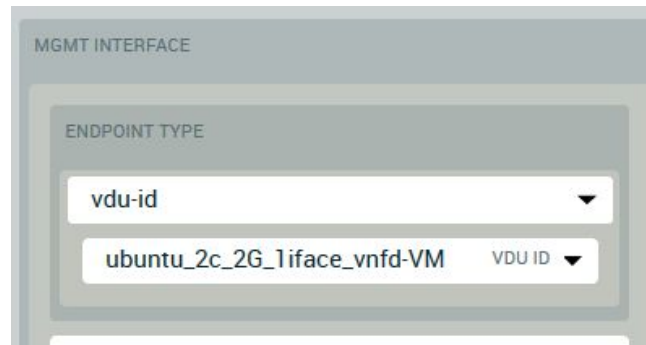


Figure 51: VNFD creation (VNF connectivity)

Table 23: Sample client VNFD

```
ubuntu_2c_2G_1iface_vnf
---
id: "ubuntu_2c_2G_1iface_vnfd"
name: "ubuntu_2c_2G_1iface_vnf"
short-name: "ubuntu_2c_2G_1iface_vnf"
vendor: "TNO"
description: "A simple VNF descriptor w/ one VDU"
version: "1.0"
mgmt-interface:
  vdu-id: "ubuntu_2c_2G_1iface_vnfd-VM"
connection-point:
  -
    name: "eth0"
    type: "VPORT"
vdu:
  -
    id: "ubuntu_2c_2G_1iface_vnfd-VM"
    name: "ubuntu_2c_2G_1iface_vnfd-VM"
    description: "ubuntu_2c_2G_1iface_vnfd-VM"
    vm-flavor:
      vcpu-count: 2
      memory-mb: 2048
```





```
storage-gb: 20
guest-epa:
  cpu-pinning-policy: "ANY"
image: "Ubuntu 16.04 LTS"
supplemental-boot-data:
  boot-data-drive: "false"
external-interface:
  -
    name: "eth0"
    vnf-d-connection-point-ref: "eth0"
    virtual-interface:
      type: "OM-MGMT"
      vpci: "0000:00:0a.0"
      bandwidth: 0
service-function-chain: "UNAWARE"
meta:
  "{ \"containerPositionMap\": { \"a66a5a22-5fc0-4c82-87fe-a4a41a90751d\": { \"top\": 30, \"left\": 260, \"right\": 510, \"bottom\": 85, \"width\": 250, \"height\": 55 }, \"a66a5a22-5fc0-4c82-87fe-a4a41a90751d/vdu-1\": { \"top\": 130, \"left\": 260, \"right\": 510, \"bottom\": 185, \"width\": 250, \"height\": 55 }, \"ubuntu_2c_2G_1iface_vnf\": { \"top\": 30, \"left\": 260, \"right\": 510, \"bottom\": 85, \"width\": 250, \"height\": 55 }, \"ubuntu_2c_2G_1iface_vnf/vdu-1\": { \"top\": 130, \"left\": 260, \"right\": 510, \"bottom\": 185, \"width\": 250, \"height\": 55 }, \"ubuntu_2c_2G_1iface_vnf/ubuntu_2c_2G_1iface_vnf\": { \"top\": 130, \"left\": 260, \"right\": 510, \"bottom\": 185, \"width\": 250, \"height\": 55 }, \"ubuntu_2c_2G_1iface_vnf/ubuntu_2c_2G_1iface_vnfd-\": { \"top\": 130, \"left\": 260, \"right\": 510, \"bottom\": 185, \"width\": 250, \"height\": 55 }, \"ubuntu_2c_2G_1iface_vnf/ubuntu_2c_2G_1iface_vnfd-VM\": { \"top\": 130, \"left\": 260, \"right\": 510, \"bottom\": 185, \"width\": 250, \"height\": 55 }, \"ubuntu_2c_2G_1iface_vnfd\": { \"top\": 30, \"left\": 260, \"right\": 510, \"bottom\": 85, \"width\": 250, \"height\": 55 }, \"ubuntu_2c_2G_1iface_vnfd/ubuntu_2c_2G_1iface_vnfd-VM\": { \"top\": 130, \"left\": 260, \"right\": 510, \"bottom\": 185, \"width\": 250, \"height\": 55 } } }
```