Programmable edge-to-cloud virtualization fabric for the 5G Media industry

D5.3 – 5G-MEDIA Web API, Tutorials and Code Samples

Work Package:	WP5 – 5G-MEDIA APIs and SDK Tools
Lead partner:	NET
Author(s):	Refik Fatih Ustok [NET], Ugur Acar [NET], Selcuk Keskin [NET], Ahmet Salih Cinkaya [NET], David Breitgand [IBM], Avi Weit [IBM], Francesco Iadanza [ENG], Francesca Moscatelli [NXW], Giacomo Bernini [NXW], David Griffin [UCL], Morteza Kheirkhah [UCL]
Delivery date (DoA):	November 30th, 2019
Actual delivery date:	November 29th, 2019
Dissemination level:	Public
Version number:	1.0 **Status:** Final
Grant Agreement N°:	761699
Project Acronym:	5G-MEDIA
Project Title:	Programmable edge-to-cloud virtualization fabric for the 5G Media industry
Instrument:	IA
Call identifier:	H2020-ICT-2016-2
Topic:	ICT-08-2017, 5G PPP Convergent Technologies, Strand 2: Flexible network applications
Start date of the project:	June 1st, 2017
Duration:	33 months

Horizon 2020 European Union Funding for Research & Innovation

Project co-funded by the European Commission under the Horizon 2020 Programme.

Revision History

Revision	Date	Who	Description
0.1	Mar. 14th, 2018	NET	TOC
0.2	May 22nd, 2019	NET	updated TOC
0.3	June 12th, 2019	NET, IBM, UCL, ENG, NXW	SDK REST APIs have been updated
0.4	July 27th, 2019	NET, IBM, UCL, ENG, NXW	SVP REST APIs have been updated
0.5	Sep. 11th,2019	NET, IBM, UCL, ENG, NXW	Initial draft
0.6	Oct. 30th, 2019	NET	First version has been released
0.7	Nov. 20th, 2019	NET	Revised after 1st internal review
0.8	Nov. 27th, 2019	NET	Revised after 2nd internal review

Quality Control

Role	Date	Who	Approved/Comment
Internal Reviewer	November 18th ,2019	ENG	
Internal Reviewer	November 18th ,2019	UPM	
2nd Internal Reviewer	November 26th ,2019	ENG	

Disclaimer

The 5G-MEDIA Consortium is the following:

Participant number	Participant organisation name	Short name	Country
01	ENGINEERING – INGEGNERIA INFORMATICA SPA	ENG	Italy
02	IBM ISRAEL - SCIENCE AND TECHNOLOGY LTD	IBM	Israel
03	SINGULARLOGIC ANONYMI ETAIREIA PLIROFORIAKON SYSTIMATON KAI EFARMOGON PLIROFORIKIS	SiLO	Greece
04	HELLENIC TELECOMMUNICATIONS ORGANIZATION S.A. - OTE AE (ORGANISMOS TILEPIKOINONION TIS ELLADOS OTE AE)	OTE	Greece
05	CORPORACION DE RADIO Y TELEVISION ESPANOLA SA	RTVE	Spain
06	UNIVERSITY COLLEGE LONDON	UCL	United Kingdom
07	TELEFONICA INVESTIGACION Y DESARROLLO SA	TID	Spain
08	UNIVERSIDAD POLITECNICA DE MADRID	UPM	Spain
09	INSTITUT FUER RUNDFUNKTECHNIK GMBH	IRT	Germany
10	NEXTWORKS	NXW	Italy
11	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS	CERTH	Greece
12	NETAS TELEKOMUNIKASYON ANONIM SIRKETI	NET	Turkey
13	INTERINNOV SAS	IINV	France
14	BITTUBES GMBH	BIT	Germany
15	NATIONAL CENTER FOR SCIENTIFIC RESEARCH - DEMOKRITOS	NCSRD	Greece

Table of Contents

Definitions and acronyms

API	Application Program Interface
CLI	Command Line Interface
FaaS	Function as a Service
NBI	Northbound interface
NS	Network Service
NSD	NS Descriptor
JSON	JavaScript Object Notation
REST	Representational state transfer
SDK	Service Development Kit
SVP	Service Virtualisation Platform
OSM	Open Source Management and Orchestration
VIM	Virtualized Infrastructure Manager
VNF	Virtual Network Function
VNFD	VNF Descriptor

Executive summary

The 5G-MEDIA project focuses on building an integrated programmable service platform that facilitates the design, development and deployment of media services, exploiting the advancements of 5G technology. To achieve this goal and improve developers' time efficiencies, a service development kit (SDK) has been developed to provide a set of open-source tools that support the rapid development of media applications using a DevOps approach. The SDK services are directly accessible to the developers and/or 3rd parties from the official REST APIs exposed by the 5G-MEDIA platform. Similarly, several services in 5G-MEDIA service virtualization platform (SVP) are also available via REST APIs. In addition to REST APIs, the 5GMEDIA platform also offers WEB APIs which provide reusable code to the developers with developed JavaScript wrappers as well as customizing the tools based on developers' needs without the hassle of having to manage the REST APIs.

The main goal of this deliverable is to provide the WEB APIs of 5G-MEDIA SDK and SVP platform including tools such as serverless framework, private and public catalogues, Open Source Management and Orchestration (OSM), monitoring, authentication and cognitive network optimizer (CNO). Furthermore, this deliverable also presents Tutorials and Code samples using the WEB APIs for the 5G-MEDIA SDK and SVP.

1. Introduction

The 5G-MEDIA project focuses on building an integrated programmable service platform that facilitates the design, development and deployment of media services, exploiting the advancements of 5G technology. To achieve this goal and improve developers' time efficiencies, 5G-MEDIA offers a Service Development Kit (SDK) environment for media applications which will hide the complexity of service development and deployment on the underlying 5G infrastructure, allowing developers to concentrate on the media application layer. Also, 5G-MEDIA delivers a service virtualisation platform (SVP) to orchestrate the deployment and scaling of the media applications, interacting automatically with the underlying network for the dynamic control of the network paths and forwarding graphs by applying machine learning-driven optimisation techniques. Several SDK and SVP services are directly accessible to the developers and/or 3rd parties from the official REST APIs exposed by the 5G-MEDIA platform and the REST APIs are reported in D5.2 REST APIs, Tutorials and Code Samples.

In addition to REST APIs, the 5GMEDIA project also offers 5G-MEDIA WEB APIs is to provide reusable code to the developers with developed JavaScript wrappers as well as customizing the tools based on developers' needs. The 5G-MEDIA WEB API wrapper is useful for JavaScript developers who wish to seamlessly integrate the 5G-MEDIA REST APIs without the hassle of having to manage the low level API calls themselves. The 5G-MEDIA WEB API is fast, simple interface which allows developers to exploit the 5G-MEDIA platform capabilities by simply adding designated JavaScript libraries in their codes.

The goal of this document is to provide the WEB APIs of 5G-MEDIA SDK and SVP platform in terms of JavaScript libraries meant to be used by the 5G-MEDIA developers. Such libraries include tools such as serverless framework, private and public catalogues, Open Source Management and Orchestration (OSM), monitoring, authentication and cognitive network optimizer (CNO). Furthermore, this deliverable also presents tutorials giving some code samples using the WEB APIs for the 5G-MEDIA SDK and SVP. In this document, all WEB APIs that are used in the 5G-MEDIA platform are listed. The JavaScript wrappers which are developed during the lifespan of 5G-MEDIA project are provided in [WEBAPI].

This document is organized as follows. In Section 2, the WEB APIs are listed for the SDK tools such as serverless framework, monitoring, vim emulator, private catalogue, validator and CNO module. In Section 3, the code samples are provided using the SDK WEB APIs in order to introduce developers how to use the WEB APIs. Following the WEB APIs of the SDK, the WEB APIs of the SVP are discussed in Section 4. Similar to Section 3, tutorials with code samples are provided for the SVP WEB APIs in Section 5. Finally, the document is concluded in Section 6.

1.1. Scope of the Deliverable

This deliverable provides only the WEB APIs of 5G-MEDIA platform including the SDK and the SVP. The REST APIs of the platform are the subject of D5.2. 5G-MEDIA REST API, Tutorials and Code Samples.

2. SDK WEB APIs Definition

2.1. Serverless Framework

The Function-as-a-Service (FaaS) allows event-driven on-demand Virtual Network Function (VNF) instantiation and execution in addition to their seamless elasticity contrary to a traditional Virtual Machine (VM) oriented approach, where virtual appliances are continuously running and thus leading to low utilization. With FaaS the VNFs are deployed where they are needed, when they are needed and for the exact duration that they are needed, which is of crucial importance for the highly dynamic 5G network environment. To the best of our knowledge, FaaS has not been applied to Network Function Virtualization (NFV) orchestration prior to 5G-MEDIA [5GMEDIASDK].

The FaaS emulation requires the Lean Openwhisk in the development environment. The developer can use and easily adapt Lean Openwhisk (OW) [Lean] with their web applications using WEB API calls as the OW capabilities in the system are available through a WEB API. As mentioned in D5.2 REST APIs, Tutorials and Code Samples, actions in OW terminology are stateless functions that run on the OW platform and encapsulate application logic to be executed in response to events. In 5G-MEDIA project, the Function-as-a-service (FaaS) based VNFs are not images in the regular sense. They are artefacts such as actions, triggers, rules and packages that together comprise the FaaS based virtual network function (VNF) image. Using the WEB API for the serverless framework, developers can easily get all created actions, rules, triggers as well as easily create, update and delete them.

The JavaScript wrappers that are created for the serverless framework are given below. The code samples showing how to utilize these wrappers are given in Section 3.1.

getActions (namespace: *string*, limit: *integer*, skip: *integer*)	Get all OW actions
Parameters	
namespace	The entity namespace
limit	Number of entities to include in the result (0-200). The default limit is 30. A value of 0 sets the limit to the maximum.
skip	Number of entities to skip in the result.
Returns	
OW Actions	An Array with JSON Objects that include all actions information such

	as their names, namespaces, versions, images, codes, limits etc.
	
	Example:
	[
	{
	"namespace":"use-case1",
	"name":"string",
	"version":"string",
	"publish":true,
	"exec":{
	"kind":"blackbox",
	"code":"string",
	"image":"string",
	"main":"string",
	"binary":true,
	"components":[
	"string"
	]
	},
	"annotations":[
	{
	"key":"string"
	}
	],
	"parameters":[
	{
	"key":"string"
	}
	],
	"limits":{
	"timeout":0,
	"memory":0,
	"logs":0,
	"concurrency":0
	},
	"updated":0
	}
	]

Throws	
Success	
Fail: Invalid namespace	

getSingleAction (namespace: *string*, name: *string,* code: *boolean)*	Get a specific action

Parameters	
namespace	The entity namespace
name:	Name of action to fetch
code:	Include action code
Returns	

OW Action Information	JSON Object that includes a specific action's information such as its name, namespace, version, image, code, limits etc.
	Example:
	{
	"namespace": "string",
	"name": "string",
	"version": "string",
	"publish": true,
	"exec": {
	"kind": "blackbox",
	"code": "string",
	"image": "string",
	"main": "string",
	"binary": true,
	"components": [
	"string"
	]
	},
	"annotations": [
	{
	"key": "string"
	}
	],
	"parameters": [
	{
	"key": "string"
	}
	],
	"limits": {
	"timeout": 0,
	"memory": 0,
	"logs": 0,
	"concurrency": 0
	},
	"updated": 0
	}

Throws
Success
Fail: Invalid namespace or action name

createAction (namespace: *string*, name: *string,* action: *object)*	Create an action
Parameters	
namespace	The entity namespace
name:	Name of action

action:	Action Entities
	Example Value:
	{
	"namespace": "string",
	"name": "string",
	"version": "string",
	"publish": true,
	"exec": {
	"kind": "blackbox",
	"code": "string",
	"image": "string",
	"main": "string",
	"binary": true,
	"components": [
	"string"
	]
	},
	"annotations": [
	{
	"key": "string"
	}
	],
	"parameters": [
	{
	"key": "string"
	}
	],
	"limits": {
	"timeout": 0,
	"memory": 0,
	"logs": 0,
	"concurrency": 0
	}
	}

Returns	

OW Action Information	JSON Object that includes the recently-created action's information such as its name, namespace, version, image, code, limits etc.
	{
	"namespace": "string",
	"name": "string",
	"version": "string",
	"publish": true,
	"exec": {
	"kind": "blackbox",
	"code": "string",
	"image": "string",
	"main": "string",
	"binary": true,
	"components": [
	"string"
	]
	},
	"annotations": [
	{
	"key": "string"
	}
	],
	"parameters": [
	{
	"key": "string"
	}
	],
	"limits": {
	"timeout": 0,
	"memory": 0,
	"logs": 0,
	"concurrency": 0
	}
	}

Throws	
Success	
Fail: Invalid namespace or action name	

deleteAction (namespace: *string*, name: *string)*	Delete an action
Parameters	

namespace	The entity namespace
name:	Name of action
Throws	
Success	
Fail: Invalid namespace or action name	

getRules (namespace: *string*, limit: *integer*, skip: *integer*)	Get all OW rules
Parameters	
namespace	The entity namespace
limit	Number of entities to include in the result (0-200). The default limit is 30. A value of 0 sets the limit to the maximum.
skip	Number of entities to skip in the result.
Returns	

OW Rules	An array with JSON objects that includes all rules information such as their names, namespaces, versions, annotations, triggers, actions etc.
	Example Value:
	[
	{
	"namespace": "string",
	"name": "string",
	"version": "string",
	"publish": true,
	"annotations": [
	{
	"key": "string"
	}
	],
	"status": "active",
	"trigger": {
	"path": "string",
	"name": "string"
	},
	"action": {
	"path": "string",

	"name": "string" } }]
Throws	
Success	
Fail: Invalid namespace	

getSingleRule (namespace: *string*, name: *string)*	Get a specific rule
Parameters	
namespace	The entity namespace
name:	Name of rule to fetch
Returns	

OW Rule Information	JSON Object that includes the information of a specific rule. Example Value:
	{
	"namespace": "string",
	"name": "string",
	"version": "string",
	"publish": true,
	"annotations": [
	{
	"key": "string"
	}
	],
	"status": "active",
	"trigger": {
	"path": "string",
	"name": "string"
	},
	"action": {
	"path": "string",
	"name": "string"
	}
	}

Throws
Success
Fail: Invalid namespace or rule name

createRule (namespace: *string*, name: *string,* rule: *object)*	Create a rule
Parameters	
namespace	The entity namespace
name:	Name of the rule

rule:	Rule Body, Example value:
	{
	"name": "string",
	"version": "string",
	"publish": true,
	"annotations": [
	{
	"key": "string"
	}
	],
	"status": "active",
	"trigger": "string",
	"action": "string"
	}

Returns	

OW Rule Information	JSON Object that includes the recently-created rule's information such as its name, namespace, version, annotations, triger, action etc.
	Example:
	{
	"name": "string",
	"version": "string",
	"publish": true,
	"annotations": [
	{
	"key": "string"
	}
	],

	"status": "active", "trigger": "string", "action": "string" }
Throws	
Success	
Fail: Invalid namespace or rule	

deleteRule (namespace: *string*, name: *string)*	Delete a rule
Parameters	
namespace	The entity namespace
name:	Name of rule
Throws	
Success	
Fail: Invalid namespace or rule name	

getTriggers (namespace: *string*, limit: *integer*, skip: *integer*)	Get all OW triggers
Parameters	
namespace	The entity namespace
limit	Number of entities to include in the result (0-200). The default limit is 30. A value of 0 sets the limit to the maximum.
skip	Number of entities to skip in the result.
Returns	
OW Triggers	An array with JSON objects that includes all triggers information such as their

	names, namespaces, versions, annotations, parameters, limits etc.
	Example Value:
	[
	{
	"namespace": "string",
	"name": "string",
	"version": "string",
	"publish": true,
	"annotations": [
	{
	"key": "string"
	}
	],
	"parameters": [
	{
	"key": "string"
	}
	],
	"limits": {},
	"rules": {},
	"updated": 0
	}
	]

Throws	
Success	
Fail: Invalid namespace	

getSingleTrigger (namespace: *string*, name: *string)*	Get a specific trigger
Parameters	
namespace	The entity namespace
name:	Name of trigger to fetch
Returns	
OW Trigger Information	JSON Object that includes the information of a specific trigger. Example Value:

	{
	"namespace": "string",
	"name": "string",
	"version": "string",
	"publish": true,
	"annotations": [
	{
	"key": "string"
	}
	],
	"parameters": [
	{
	"key": "string"
	}
	],
	"limits": {},
	"rules": {},
	"updated": 0
	}

Throws	
Success	
Fail: Invalid namespace or trigger name	

createTrigger (namespace: *string*, name: *string,* trigger: *object)*	Create a trigger
Parameters	
namespace	The entity namespace
name:	Name of the trigger

trigger:	Trigger Body, Example value:
	{
	"namespace": "string",
	"name": "string",
	"version": "string",
	"publish": true,
	"annotations": [
	{
	"key": "string"
	}
	],
	"parameters": [
	{

	"key": "string" }], "limits": {} }
Returns	

OW Trigger Information	JSON Object that includes the recently-created trigger's information such as its name, namespace, version, annotations, parameters, limits etc.
	Example value:
	{
	"namespace": "string",
	"name": "string",
	"version": "string",
	"publish": true,
	"annotations": [
	{
	"key": "string"
	}
	],
	"parameters": [
	{
	"key": "string"
	}
	],
	"limits": {}
	}

Throws	
Success	
Fail: Invalid namespace or trigger	

deleteTrigger (namespace: *string*, name: *string)*	Delete a trigger
Parameters	
namespace	The entity namespace

name:	Name of trigger
Throws	
Success	
Fail: Invalid namespace or trigger name	

2.2. Monitoring Tools

The 5G-MEDIA SDK has a set of monitoring tools available for media application developers that can gather and centralize monitored metrics into a local database. Metrics can be queried from VNFs deployed in the emulator. After the metric data is stored in the local database, further analysis can take place to debug or optimize the performance of the monitored VNF or service.

Monitoring tools consist of cAdvisor [cAdvisor] (data collector), Prometheus [PROMETHEUS] (database), and Grafana [Grafana] (dashboard) each of which are explained in detail in Deliverable *D5.1 Programming Tools [D5.1]*. These tools also provide REST API solutions which are listed in Deliverable *D5.2 REST APIs, Tutorials and Code Samples*. In this section, we give the lists of JavaScript wrappers for each monitoring tools so that developers can exploit the capabilities of these tools without the hassle of having to manage the low level API calls.

As mentioned in Deliverables *D5.1 Programming Tools* and *D5.2 REST APIs, Tutorials and Code Samples*. In this section, the JavaScript wrappers are provided for developers who want to utilize monitoring tools and easily implement in their code.

createDashboard (dashboard: *object*)	Create/ Update a dashboard in Grafana
Parameters	

dashboard	The complete dashboard model (JSON) Example Model:
	{
	"dashboard": {
	"id": null,
	"uid": null,
	"title": "Production Overview",
	"tags": ["templated"],
	"timezone": "browser",
	"schemaVersion": 16,
	"version": 0
	},
	"folderId": 0,

	"overwrite": false }
Throws	
Success	
Fail: Invalid dashboard object	

getMonitoringData(query: *string*)	Get data by query via Prometheus
Parameters	
query	Prometheus expression query string
Returns	

Monitoring Data	Monitoring Data in JSON object
	Example Value:
	{
	"status": "success",
	"data": {
	"resultType": "vector",
	"result": [
	{
	"metric": {
	"__name__": "up",
	"job": "prometheus",
	"instance": "localhost:9090"
	},
	"value": [
	1435781451.781,
	"1"
	]
	},
	{
	"metric": {
	"__name__": "up",
	"job": "node",
	"instance": "localhost:9100"
	},
	"value": [
	1435781451.781,
	"0"
	]

	}] } }
Throws	
Success	
Fail: Invalid query	

getMonitoringContainer(name: *string*)	Get information of a selected container via cAdvisor
Parameters	
name	Name of the container to have the information. “all” for all containers
Returns	

Container Data	Container Data in JSON object including container information and list of events using cAdvisor
	Example Value :
	{
	"/docker/86c1d6c61a49f7bb5be2296bd0648e0e8d0aed6c8e4bda6ae351f5120b558052": {
	"id": "86c1d6c61a49f7bb5be2296bd0648e0e8d0aed6c8e4bda6ae351f5120b558052",
	"name": "/docker/86c1d6c61a49f7bb5be2296bd0648e0e8d0aed6c8e4bda6ae351f5120b558052",
	"aliases": [
	"redis",
	
	"86c1d6c61a49f7bb5be2296bd0648e0e8d0aed6c8e4bda6ae351f5120b558052"
	],
	"namespace": "docker",
	"spec": {
	"creation_time": "2019-08-29T13:27:22.431584149Z",
	"has_cpu": true,
	"cpu": {
	"limit": 1024,
	"max_limit": 0,
	"mask": "0-1",

	"period": 100000
	},
	"has_memory": true,
	"memory": {
	"limit": 9223372036854772000,
	"reservation": 9223372036854772000
	},
	"has_network": true,
	"has_filesystem": true,
	"has_diskio": true,
	"has_custom_metrics": false,
	"image": "redis:4.0"
	},
	"stats": [...]
	}

Throws	
Success	
Fail: Invalid container name	

2.3. Vim-Emulator

The 5G-MEDIA project offers a multi-vim Emulator environment which facilitates local prototyping and testing of network services in realistic end-to-end multi-PoP scenarios. As explained in the Serverless Framework section, the 5G-MEDIA utilizes Lean OW [Lean] for emulation FaaS-type application. The 5G-MEDIA Emulator also utilizes vim-emu for non-FaaS applications. The design of vim-emu is based on a tool called Containernet which extends the well-known Mininet emulation framework and allows us to use standard Docker containers as VNFs within the emulated network. It also allows adding and removing containers from the emulated network at runtime which is not possible in Mininet. More information on vim-emu can be found in [D5.1 Programming Tools]

vim-emu also known as son-emu WEB API is mainly used to control a deployed service in son-emu/vim-emu. This section gives an overview of implemented WEB API which provides developers some of the capabilities of vim-emu without dealing with REST API such as listing data centers, get a specific VNF instance information or deleting a specific VNF instance as well as creating or deleting the action that monitors the counters of a VNF interface.

listDatacenters()	List all data centers

Returns	

List of all data centers	The complete list of data centers
	Example Value:
	[
	{
	"internalname": "dc2",
	"label": "datacenter2",
	"metadata": {},
	"n_running_containers": 0,
	"switch": "dc2.s1"
	},
	{
	"internalname": "dc3",
	"label": "long_data_center_name3",
	"metadata": {},
	"n_running_containers": 0,
	"switch": "dc3.s1"
	}
	]

Throws	
Success	
Fail: Invalid dashboard object	

getVNFinfo(name: *string*)	Get a specific VNF instance information
Parameters	
name	label of the VNFs to list the computation metrics
Returns	
VNF Instance Information	Instance information in JSON Object Example Value: { "vnf1": { "cpu_period": null, "cpu_quota": -1, "cpu_shares": null, "cpuset": null, "datacenter": "cvim1",

	"docker_network": "172.17.0.2",
	"flavor_name": "tiny",
	"id":
	"48c52260319bf3142917e0df8301ed38f4ed13f939164448618db027f81e5c2b",
	"image": "ubuntu",
	"mem_limit": null,
	"memswap_limit": null,
	"name": "vnf1",
	"network": [
	{
	"dc_portname": "dc1.s1-eth1",
	"intf_name": "vnf1-eth0",
	"ip": "10.0.0.2",
	"mac": "7a:ee:4d:0d:3d:fa",
	"status": "MISSING",
	"up": false
	}
	],
	"short_id": "48c52260319b",
	"state": {
	"Dead": false,
	"Error": "",
	"ExitCode": 0,
	"FinishedAt": "0001-01-01T00:00:00Z",
	"OOMKilled": false,
	"Paused": false,
	"Pid": 15200,
	"Restarting": false,
	"Running": true,
	"StartedAt": "2017-03-23T15:30:43.873147835Z",
	"Status": "running"
	}
	}

Throws	
Success	
Fail: Invalid vnf name	

deleteVNF(name: *string*)	Delete a specific VNF instance
Parameters	

name	label of the VNFs to delete
Throws	
Success	
Fail: Invalid vnf name	

createMonitoring(name: *string,* interface: *string,* metric: *string*)	Create the action that monitors the counters of a VNF interface.
Parameters	
name	Name of the VNFs to monitor
interface	Name of the VNF interface to be monitored
metric	One of the following metrics: tx_bytes: Transmitted bytes rx_bytes: Received bytes tx_packets: Transmitted packets rx_packets: Received Packets
Returns	
Message	message string indicating if the monitor action is successful or not
Throws	
Success	
Fail: Invalid vnf name	

deleteMonitoring(name: *string,* interface: *string,* metric: *string*)	delete the action that monitors the counters of a VNF interface.

Parameters	
name	Name of the VNFs to monitor
interface	Name of the VNF interface to be monitored
metric	One of the following metrics: tx_bytes: Transmitted bytes rx_bytes: Received bytes tx_packets: Transmitted packets rx_packets: Received Packets
Throws	
Success	
Fail: Invalid vnf name	

2.4. Private Catalogue

Private catalogue allows developers to design and validate applications, and in the core of the SVP as a public catalogue, stores all available applications and NSs descriptors for all platform users. The catalogue also automatically onboards the network services and virtual network functions to Open Source Management and Orchestration (OSM) via plugin. The detailed information and workflow of catalogue is provided in Deliverable *D4.1 5G-MEDIA Catalogue APIs and Network Apps* [D4.1]. This section gives an overview of implemented WEB API which provides developers some of the capabilities of catalogue without dealing with REST API such as creating NSD resource, getting NSD lists or a specific NSD, creating VNFD resources, getting VNFD lists or information of a specific VNFD that is onboarded in the catalogue.

createNSD(CreateNsdInfoRequest: *JSON Object*)	Create NSD Resource
Parameters	
CreateNsdInfoRequest	JSON Object Example Value: { "nsdId": "85ee1962-f65b-4965-a73f-96b5e1d9d068", "nsdName": "faas_pingpong_nsd",

	"nsdVersion": "1.1",
	"nsdDesigner": "IBM",
	"nsdInvariantId": "85ee1962-f65b-4965-a73f-96b5e1d9d068",
	"vnfPkgIds": [
	"44aa96b6-2c55-4d20-a7c0-faca82d492d2",
	"c19e63d6-46ec-48d7-9c18-3f10e58eee3f"
	],
	"pnfdInfoIds": [],
	"nestedNsdInfoIds": [],
	"nsdOnboardingState": "ONBOARDED",
	"onboardingFailureDetails": null,
	"nsdOperationalState": "ENABLED",
	"nsdUsageState": "NOT_IN_USE",
	"userDefinedData": {},
	"_links": {
	"self": "/nsd/v1/ns_descriptors/6f7ebae1-29b1-42bd-a52b-a60cb485ba7e",
	"nsd_content": "/nsd/v1/ns_descriptors/6f7ebae1-29b1-42bd-a52b-a60cb485ba7e/nsd_content"
	},
	"manosOnboardingStatus": {
	"NET_OSMR4.0": "ONBOARDED"
	},
	"c2cOnboardingState": "UNPUBLISHED"
	}

Returns	
NSD ID	String
Throws	
Success	
Fail: Invalid NSD Request	

getNSDList()	Query information about multiple NS descriptor resources.
Returns	
NSDs	Array of information in JSON Objects Example Value: [

```
{
  "id": "082ead84-3689-4c7b-a231-56c79a31398c",
  "nsdId": "95a6c14e-a1ad-4893-8865-
3442404a8d83",
  "nsdName": "vtranscoder_2_7_2_nsd",
  "nsdVersion": "1.1",
  "nsdDesigner": "IBM",
  "nsdInvariantId": "95a6c14e-a1ad-4893-8865-
3442404a8d83",
  "vnfPkgIds": [
   "a8bc9387-4e50-4315-9505-2ecc752f135e"
  ],
  "pnfdInfoIds": [],
  "nestedNsdInfoIds": [],
  "nsdOnboardingState": "ONBOARDED",
  "onboardingFailureDetails": null,
  "nsdOperationalState": "ENABLED",
  "nsdUsageState": "NOT_IN_USE",
  "userDefinedData": {},
  "_links": {
   "self": "/nsd/v1/ns_descriptors/082ead84-3689-
4c7b-a231-56c79a31398c",
    "nsd_content":
"/nsd/v1/ns_descriptors/082ead84-3689-4c7b-a231-
56c79a31398c/nsd_content"
  },
  "manosOnboardingStatus": {
   "NET_OSMR4.0": "ONBOARDED"
  },
  "c2cOnboardingState": "UNPUBLISHED"
 },
 {
  "id": "be984a9c-206b-4a30-8907-413ca80ec5ae",
  "nsdId": "7cdc0f3a-8023-4769-8315-
1ef31be9b84a",
  "nsdName": "vspeech_nsd",
  "nsdVersion": "1.1",
  "nsdDesigner": "IRT",
  "nsdInvariantId": "7cdc0f3a-8023-4769-8315-
1ef31be9b84a",
  "vnfPkgIds": [
   "c5b80ace-e8bb-498d-905e-3f49d5826267",
   "03c46515-f5c3-47a3-9086-b55bfac74233"
  ],
  "pnfdInfoIds": [],
  "nestedNsdInfoIds": [],
  "nsdOnboardingState": "ONBOARDED",
  "onboardingFailureDetails": null,
  "nsdOperationalState": "ENABLED",
  "nsdUsageState": "NOT_IN_USE",
  "userDefinedData": {},
  "_links": {
```

	"self": "/nsd/v1/ns_descriptors/be984a9c-206b-4a30-8907-413ca80ec5ae",
	"nsd_content": "/nsd/v1/ns_descriptors/be984a9c-206b-4a30-8907-413ca80ec5ae/nsd_content"
	},
	"manosOnboardingStatus": {
	"NET_OSMR4.0": "ONBOARDED"
	},
	"c2cOnboardingState": "UNPUBLISHED"
	}
	]

Throws	
Success	
Fail: Invalid Request	

getNSD(id: *string*)	Query a specific NSD
Parameters	
id	label of the NSD Info ID
Returns	
NSD	NSD information in JSON Object Example Value: { "id": "082ead84-3689-4c7b-a231-56c79a31398c", "nsdId": "95a6c14e-a1ad-4893-8865-3442404a8d83", "nsdName": "vtranscoder_2_7_2_nsd", "nsdVersion": "1.1", "nsdDesigner": "IBM", "nsdInvariantId": "95a6c14e-a1ad-4893-8865-3442404a8d83", "vnfPkgIds": ["a8bc9387-4e50-4315-9505-2ecc752f135e"], "pnfdInfoIds": [], "nestedNsdInfoIds": [], "nsdOnboardingState": "ONBOARDED", "onboardingFailureDetails": null,

	"nsdOperationalState": "ENABLED",
	"nsdUsageState": "NOT_IN_USE",
	"userDefinedData": {},
	"_links": {
	"self": "/nsd/v1/ns_descriptors/082ead84-3689-4c7b-a231-56c79a31398c",
	"nsd_content": "/nsd/v1/ns_descriptors/082ead84-3689-4c7b-a231-56c79a31398c/nsd_content"
	},
	"manosOnboardingStatus": {
	"NET_OSMR4.0": "ONBOARDED"
	},
	"c2cOnboardingState": "UNPUBLISHED"
	}

Success	
Fail: Invalid Request	

createVNFD(file: *CSAR*)	Create NSD Resource
Parameters	
file	Compressed package file in CSAR format
Returns	
VNFD Package ID	String
Throws	
Success	
Fail: Invalid VNFDRequest	

getVNFDList()	Query information about multiple VNF package resources.
Returns	
VNFDs	Array of Information in JSON Objects

	Example Value:
	[
	{
	"id": "1621907a-1e0c-4d74-a004-ffae8d902e44",
	"vnfdId": "d2d4a5f8-3a2a-496d-89c3-b42459f440e4",
	"vnfProvider": "OSM",
	"vnfProductName": "ping_vnfd",
	"vnfSoftwareVersion": null,
	"vnfdVersion": "1.1",
	"checksum": null,
	"softwareImages": null,
	"additionalArtifacts": null,
	"onboardingState": "ONBOARDED",
	"operationalState": "ENABLED",
	"usageState": "NOT_IN_USE",
	"userDefinedData": {},
	"_links": {
	"self": "/vnfpkgm/v1/vnf_packages/1621907a-1e0c-4d74-a004-ffae8d902e44",
	"vnfd": "/vnfpkgm/v1/vnf_packages/1621907a-1e0c-4d74-a004-ffae8d902e44/vnfd",
	"packageContent": "/vnfpkgm/v1/vnf_packages/1621907a-1e0c-4d74-a004-ffae8d902e44/package_content"
	},
	"manosOnboardingStatus": {
	"NET_OSMR4.0": "ONBOARDED"
	},
	"c2cOnboardingState": "UNPUBLISHED"
	},
	{
	"id": "d9fb09be-156e-4cb1-952c-6387a279e57f",
	"vnfdId": "e67faebc-52c0-4bf6-8f4a-e5049855eae1",
	"vnfProvider": "OSM",
	"vnfProductName": "pong_vnfd",
	"vnfSoftwareVersion": null,
	"vnfdVersion": "1.1",
	"checksum": null,
	"softwareImages": null,
	"additionalArtifacts": null,
	"onboardingState": "ONBOARDED",
	"operationalState": "ENABLED",
	"usageState": "NOT_IN_USE",
	"userDefinedData": {},
	"_links": {
	"self": "/vnfpkgm/v1/vnf_packages/d9fb09be-156e-4cb1-952c-6387a279e57f",

	"vnfd": "/vnfpkgm/v1/vnf_packages/d9fb09be-156e-4cb1-952c-6387a279e57f/vnfd",
	"packageContent": "/vnfpkgm/v1/vnf_packages/d9fb09be-156e-4cb1-952c-6387a279e57f/package_content"
	},
	"manosOnboardingStatus": {
	"NET_OSMR4.0": "ONBOARDED"
	},
	"c2cOnboardingState": "UNPUBLISHED"
	}
	]

Throws	
Success	
Fail: Invalid Request	

getVNFD(id: *string*)	Query a specific VNFD
Parameters	
id	label of the VNFD Info ID
Returns	

VNFD	Information in JSON Objects
	Example Value:
	{
	"id": "1621907a-1e0c-4d74-a004-ffae8d902e44",
	"vnfdId": "d2d4a5f8-3a2a-496d-89c3-b42459f440e4",
	"vnfProvider": "OSM",
	"vnfProductName": "ping_vnfd",
	"vnfSoftwareVersion": null,
	"vnfdVersion": "1.1",
	"checksum": null,
	"softwareImages": null,
	"additionalArtifacts": null,
	"onboardingState": "ONBOARDED",
	"operationalState": "ENABLED",
	"usageState": "NOT_IN_USE",
	"userDefinedData": {},
	"_links": {

	"self": "/vnfpkgm/v1/vnf_packages/1621907a-1e0c-4d74-a004-ffae8d902e44",
	"vnfd": "/vnfpkgm/v1/vnf_packages/1621907a-1e0c-4d74-a004-ffae8d902e44/vnfd",
	"packageContent": "/vnfpkgm/v1/vnf_packages/1621907a-1e0c-4d74-a004-ffae8d902e44/package_content"
	},
	"manosOnboardingStatus": {
	"NET_OSMR4.0": "ONBOARDED"
	},
	"c2cOnboardingState": "UNPUBLISHED"
	}

Success	
Fail: Invalid Request	

2.5. Cognitive Network Optimizer

As mentioned in Deliverable *D5.2 REST APIs, Tutorials and Code Samples*, integration of Cognitive Network Optimizer (CNO) in the SDK includes the configuration of CNO Training and Deploying the Trained Model to the SVP. The 5G-MEDIA CNO training capabilities are also available through WEB API such as starting training, starting tensorboard and deploying it to MAPE.

The JavaScript wrappers for CNO Training in the SDK are given as follows

trainData(trainModel: *Object*)	Start Training
Parameters	

trainingConfigurations	Training Configurations in JSON Format
	Example Model:
	{
	"trainModel": {
	"alpha": “0.5”,
	"trafficPattern": “random”,
	"actorLearningRate": "0.0001",
	"criticLearningRate": “0.001”,
	"linkCapacity": "20",
	"rewardFunction": “bls”,
	"uploadFileName": “./cooked_traces/”,

	"parallelAgent": “2” } }
Throws	
Success	
Fail: Invalid request	

stopTraining()	Terminate training
Returns	
Status	Return Output “Training Stopped”
Throws	
Success	
Fail: Invalid request	

startTensorboard()	Start Tensorboard
Returns	
Status	Return Output “Tensorboard Started”
Throws	
Success	
Fail: Invalid request	

deployMape(*ip*: String, *pemfile*: File)	Deploy Training Model to production environment

Parameters	
ip	String
pemfile	File for permission to access to the server
Throws	
Success	
Fail: Invalid request	

3. SDK WEB APIs Tutorials and Code Samples

3.1. Serverless Framework

Here are some example JavaScript tutorials including some sample codes using the WEB APIs for the serverless framework. The developer can utilize these wrappers by simply importing functions from "./serverless" library. These examples include pieces of codes about how to get, create and delete OW actions, rules and triggers which constitute the FaaS based VNF image.

```
// Get all openwhisk actions

import { getAction, getSingleAction, createAction, deleteAction, getRules, getSingleRule,
createRule, deleteRule, getTriggers, getSingleTrigger, createTrigger, deleteTrigger } from
"./serverless"

const actionList= getAction({
   namespace: { ... },
   limit: { ... },
   skip:{ ... },
});


// Get single action

const action= getSingleAction({
   namespace: { ... },
   name: { ... },
   code:{ ... },
});


// Create an action

const actionCreated=createAction({
   namespace: { ... },
   name: { ... },
   action:{ ... },
});


// Delete an action

deleteAction({
   namespace: { ... },
   name: { ... },
});



// Get all rules
```

```
const ruleList= getRules({
   namespace: { ... },
   limit: { ... },
   skip:{ ... },
});


// Get a single rule

const rule= getSingleRule({
   namespace: { ... },
   name: { ... },
});


// Create a rule

const ruleCreated=createRule({
   namespace: { ... },
   name: { ... },
   rule:{ ... },
});


// Delete an rule

deleteRule({
   namespace: { ... },
   name: { ... },
});


// Get all triggers

const triggerList= getTriggers({
   namespace: { ... },
   limit: { ... },
   skip:{ ... },
});
```

```
// Get a single trigger

const trigger= getSingleTrigger({
   namespace: { ... },
   name: { ... },
});

// Create a rule

const triggerCreated=createTrigger({
   namespace: { ... },
   name: { ... },
   trigger:{ ... },
});

// Delete a trigger

deleteTrigger({
   namespace: { ... },
   name: { ... },
});
```

3.2. Monitoring Tools

Here are some example JS tutorials including some sample codes using the WEB APIs for the monitoring tools. The developer can utilize these wrappers by simply importing functions from "./monitoring" library. These examples include pieces of codes about how to get authentication token, how to create a dashboard, get monitoring data and container information.

```
// Create a dashboard in Grafana

import { createDashboard, monitoringAuth, getMonitoringData, getMonitoringContainer  }
from ’./monitoring’

// Get Monitoring Authentication Token

const AuthResponse = monitoringAuth({username, password, apiKeyName, role});

// Create Dashboard

 createDashboard({
   dashboard: {
          "dashboard": {
               "id": null,

               "uid": null,

              "title": "Production Overview",

              "tags": [ "templated" ],

             "timezone": "browser",

             "schemaVersion": 16,

             "version": 0

            },

     },

  });


// Get monitoring data information

const monitoringData= getMonitoringData({
   query: { ... },
});


// Get monitoring container information

const monitoringContainer=getMonitoringContainer({
   name: { ... },
```

```
});
```

3.3. Vim Emulator

Here are some example JS tutorials including some sample codes using the WEB APIs for the serverless framework. The developer can utilize these wrappers by simply importing functions from "./vimEmu" library. These examples include pieces of codes about how to get available datacenters, how to get a specific VNF instance information or delete a specific VNF instance.

```
// List all data centers

import { getVNFinfo, deleteVNF } from './vimEmu'

const datacenters =  listDatacenters();


// Get a specific VNF Information

const vnf = getVNFinfo({
   name: { ... },
});


// Delete a specific VNF Information

deleteVNF({
   name: { ... },
});
```

3.4. Private Catalogue

Here are some example JS tutorials including some sample codes using the WEB APIs for the private catalogue. The developer can utilize these wrappers by simply importing functions from "./catalogue" library. These examples include pieces of codes about how to onboard a network service or virtual network function, as well as query information about a single or multiple NS or VNF.

```
import { createNSD, getNSD, createVNFD, getVNFDList, getVNFD} from './catalogue'

// Create (Onboard) an NSD
const nsd1= createNSD({
   nsd: { ... },
});

// Query information about multiple NSD resources in the private catalogue
const nsds= getNSDList();

// Query a specific NSD
const nsd= getNSD ({
   id: { ... },
});

// Create (Onboard) a VNFD
const vnfd1= createVNFD({
   file: { ... },
});

// Query information about multiple VNFD packages in the private catalogue
const vnfds= getVNFDList();

// Query information about a specific VNFD package in the private catalogue
const nsd= getVNFD ({
   id: { ... },
});
```

3.5. Cognitive Network Optimizer

Here are some example JS tutorials including some sample codes using the WEB APIs for the Cognitive Network Optimizer in the SDK. The developer can utilize these wrappers by simply importing functions from "./cno" library. These examples include pieces of codes about how to start and stop training, starting tensorboard and deploying the latest training model which has ".ckpt.meta" extension to the production environment.

```
// start training

import { trainData,stopTraining,startTensorboard,deployMape } from './cno'

trainData({
  trainingConfigurations: { ... },
  });


// stop training

stopTraining();


// start tensorboard

startTensorboard();


// start tensorboard

deployMape(ip,pemfile);
```

4. SVP WEB APIs Definition

4.1. Serverless Framework

As mentioned in Deliverable D3.2 *Specification of the 5G-MEDIA Serverless Computing Framework* [D3.2], the 5G-MEDIA platform offers a developer the benefits of the FaaS

programming model in a way which is compatible with ETSI MANO and without vendor lock-in. A specific FaaS framework, Apache OpenWhisk, used for the reference implementation can be easily replaced by other FaaS frameworks in the future. Apache OpenWhisk is an extensible serverless computing platform that supports functions (also known as "actions") that can be written in multiple programming languages including Node.js, Python, Swift, Java, and PHP. Also, OpenWhisk supports native binaries. With a native binary, any executable that is compatible with a standard OpenWhisk container may run as a serverless function. These functions are termed blackbox actions. Blackbox actions derive their container image from the base OpenWhisk container that includes some basic management services allowing the OpenWhisk framework to interact with the action.

The project has developed a version of Apache OpenWhisk with a small footprint that we term Lean OpenWhisk [Lean]. Lean OpenWhisk is used as part of the 5G-MEDIA SDK as described in D5.1: "5G-MEDIA Programming Tools". The semantic flow of the Openwhisk that is used in SVP is identical to the one in the SDK. Therefore, the WEB APIs referred in Section 2.1 is also valid for the Serverless Framework in the SVP.

4.2. Public Catalogue

A key feature of the 5G-MEDIA architecture is the catalogue where the descriptors of available applications and NSs are stored. This catalogue is present in the SDK as a private catalogue, allowing developers to design and validate applications, and in the core of the SVP as a public catalogue, storing all available applications and NSs descriptors for all platform users. The WEB API of the public catalogue is same as the one for the private catalogue and has been provided in Section 2.4.

4.3. Open Source MANO

Open Source MANO (OSM) is an ETSI-hosted initiative for the development of open-source NFV Management and Orchestration software stacks. The initiative is fully aligned with the ETSI-developed NFV reference architecture.

As mentioned in *D5.1 Programming Tools* [D5.1] and *D5.2 REST APIs, Tutorials and Code Samples* [D5.2], the 5G-MEDIA platform utilizes developer's local Open Source Management and Orchestration (OSM) environment so that the developer can manage emulation environments such as vim-emu and Lean OW. It is also the main interface for managing the lifecycle operations on applications such as their instantiation and termination. More information on OSM can be found in Deliverable *D5.1 Programming Tools* [D5.1].

The 5G-MEDIA SDK exploits Open Source Management and Orchestration (OSM) Release 5. This section gives an overview of implemented WEB API commands of OSM which is also utilised in the SDK.

osmAuthToken({ "username": "admin", "password": "admin", "project_id": "admin" })	Create a specific NSD resource in the OSM with its content.
Returns	
OSMAuthResponse	JSON Example Value: {Value: ******}
Throws	
Success	
Fail: Invalid NSD Content	

getNsds ()	Get all NS descriptors onboarded in the OSM
Returns	

NSD List	Array of JSON Objects
	Example Value:
	[
	{
	"_id": "string",
	"id": "string",
	"name": "string",
	"description": "string"
	}
	]

Throws	
Success	
Fail: Invalid configuration	

getVnfds ()	Get all VNF packages onboarded in the OSM
Returns	

VNFD List	Array of JSON Object
	Example Value:
	[
	{
	"_id": "string",
	"id": "string",
	"name": "string",
	"description": "string"
	}
	]

Throws	
Success	
Fail: Invalid configuration	

createNsd(Body: *Object)*	Create a specific NSD resource in the OSM with its content.
Parameters	

Body	JSON Object
	Example Value:
	{
	"_id": "string",
	"id": "string",
	"name": "string",
	"description": "string"
	}

Returns	
NSD Id	String
Throws	
Success	

Fail: Invalid NSD Content

getNsdInfo (nsdId: *string)*	Get a specific NSD info using its unique ID.
Parameters	
nsdId	Unique NSD ID
Returns	

NSD Info	JSON Object
	{
	"_id": "string",
	"id": "string",
	"name": "string",
	"description": "string"
	}

Throws	
Success	
Fail: Invalid NSD ID	

createVnfd(Body: *Object)*	Create a specific VNFD package in the OSM with its content .
Parameters	

Body	JSON Object
	Example Value
	{
	"_id": "string",
	"id": "string",
	"name": "string",
	"description": "string"
	}

Returns	
VNFD Id	String

Throws
Success
Fail: Invalid VNFD Content

getVnfdInfo (vnfdId: *string)*	Get a specific VNFD info using its unique ID.
Parameters	
vnfdId	Unique VNFD ID
Returns	

VNFD Info	JSON Object
	Example Value:
	{
	"_id": "string",
	"id": "string",
	"name": "string",
	"description": "string"
	}

Throws	
Success	
Fail: Invalid VNFD ID	

4.4. Authentication

From the perspective of the SDK, the main advancement about the Authentication service since the release of the deliverable D4.1 and D5.1 at M15 has been the integration of OpenID Connect protocol in the MANO framework used in the project OSM [OSM] and within the 5G-MEDIA Catalogue. Details about the architecture and the main components involved can be found in the D4.2 "5G-MEDIA Catalogue Portal and Network Apps". As a consequence, the main change on the SDK is the login process on the Identity Server to the access and refresh tokens to be used to access the Catalogue NBI.

This section gives an overview of implemented WEB API command of Authentication. As provided in the following table, the developers can obtain a token using a simple JavaScript function without having to deal with complex REST APIs.

getToken (clientCredentials: Object*)*	Get an access token using resource owner password credentials
Parameters	
clientCredentials	JSON Object that includes client_id, username, password, grant_type and client_secret
Returns	
An object that includes access and refresh tokens	JSON Object
Throws	
Success	
Fail: Invalid Request	

5. SVP WEB APIs Tutorials and Code Samples

5.1. Serverless Framework

Code samples of serverless framework is as same as it is given for the SDK in Section 3.1.

5.2. Public Catalogue

Code samples for the public catalogue is same as the ones given for the private catalogue and has been provided in Section 3.2.

5.3. Open Source MANO

Here are some example JS tutorials with some sample codes using the WEB APIs for the Open Source MANO (OSM). The developer can utilize these wrappers by simply importing functions from "./osm" library. These examples include pieces of codes about how to onboard NSDs, VNFDs, getting the onboarded NSD or VNFD information.

```
// Get the List of NSDs onboarded in the OSM

import { getNsds, getVnfs,createNsd, getNsdInfo, createVnfd, getVnfdInfo  } from './osm'

const nsdList = getNsds();

// Get the List of VNFD packages onboarded in the OSM


const vnfdList = getVnfs();


// Create a specific NSD

const nsd_new=createNsd ({
   Body: { ... },
});


// Get  a specific NSD  information

const nsdInfo=getNsdInfo ({
   nsdId: { ... },
});


// Create a specific VNFD

const vnfd_new=createVnfd ({
   Body: { ... },
});

// Get  a specific VNFD  information

const vnfdInfo=getVnfdInfo ({
   vnfdId: { ... },
});
```

5.4. Authentication

Here are some example JS sample codes using the WEB APIs for the Authentication.

```
// Get a Token

import { getToken } from 'authentication'

var token=getToken ({
   clientCredentials: { ... },
});
```

6. Conclusions

In this deliverable, we have provided the WEB APIs of 5G-MEDIA SDK and SVP platform including tools such as serverless framework, private and public catalogues, OSM, monitoring, authentication and CNO. Furthermore, this deliverable has also presented some tutorials and Code samples using the WEB APIs for both SDK and SVP tools in order to show developers how to exploit the capabilities of the platform with the developed JavaScript libraries, without the hassle of having to manage the low level REST API calls.

7. References

[D3.4] 5G-MEDIA Deliverable "*D3.4 5G-MEDIA Operations and Configuration Platform"*

[D4.1] 5G-MEDIA Deliverable "*D4.1 5G-MEDIA Catalogue APIs and Network Apps*"

[D4.2] 5G-MEDIA Deliverable "*D4.2 5G-MEDIA Catalogue Portal and Network Apps*"

[D5.1] 5G-MEDIA Deliverable *"D5.1 5G-MEDIA Programming Tools*"

[5GMEDIASDK] U. Acar, R. F. Ustok, S. Keskin, D. Breitgand and A. Weit, "Programming Tools for Rapid NFV-Based Media Application Development in 5G Networks," *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Verona, Italy, 2018, pp. 1-5.

[cAdvisor] cAdvisor https://github.com/google/cadvisor

[ETSIvim] Vim-emu: A NFV multi-PoP emulation platform https://osm.etsi.org/wikipub/index.php/VIM_emulator

[Grafana] Grafana https://grafana.com/dashboards

[Lean] Lean OW https://github.com/kpavel/incubator-openwhisk/tree/lean

[OSM] OSM REST API
https://forge.etsi.org/swagger/ui/?url=https%3A%2F%2Fosm.etsi.org%2Fgitweb%2F%3Fp%3Dosm%2FSOL005.git%3Ba%3Dblob_plain%3Bf%3Dosm-openapi.yaml%3Bhb%3DHEAD#/

[OWLimits] Openwhisk Entities
https://github.com/apache/openwhisk/blob/master/docs/reference.md#openwhisk-entities

[KEYCLOAK] KEYCLOAK Open Source Identity and Access Management
https://www.keycloak.org/

[PROMETHEUS] PROMETHEUS REST API
https://prometheus.io/docs/prometheus/latest/querying/api/

[WEBAPI] 5G-MEDIA JavaScript Wrappers Public Repository, Online at "https://github.com/5g-media/5G-Media-Web-API.git"