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

D5.2 - 5G-MEDIA Rest 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, 2019	NET	TOC
0.2	May 22nd, 2019	NET	updated TOC
0.3	June 10th, 2019	NET, IBM, UCL, ENG, NXW	SDK REST APIs have been updated
0.4	July 25th, 2019	NET, ENG, IBM, NXW	SVP REST APIs have been updated
0.5	Sep. 10th,2019	NET, ENG, IBM, NXW, UCL	Initial draft
0.6	Oct. 30th, 2019	NET	First version has been released
0.7	Nov. 20th, 2019	NET	Revised after 1st internal review
1.0	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

List of Figures

List of Tables

Definitions and acronyms

API	Application Program Interface
CLI	Command Line Interface
FaaS	Function as a Service
NBI	Northbound Interface
JSON	Javascript Object Notation
NS	Network Service
NSD	NS Descriptor
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 proposes a solution for an integrated programmable service platform that facilitates the design, development and deployment of media services, exploiting the advancements of 5G technology while ensuring the applications allocate the resources they need to deliver high quality of experience and so that the network is not overwhelmed by media traffic. 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 available via REST APIs.

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

A RESTful API -- also referred to as a RESTful web service or REST API -- is based on representational state transfer (REST) technology, an architectural style and approach to communications often used in web services development. A REST API defines a set of functions which developers can perform requests and receive responses via HTTP protocol such as GET and POST. Because REST API's use HTTP, they can be used by practically any programming language. Several SDK and SVP services are directly accessible to the developers and/or 3rd parties from the official REST APIs exposed by the 5GMEDIA platform.

The goal of this document is to provide the REST APIs of 5G-MEDIA SDK and SVP platform which includes the tools such as validator, catalogue, OSM, monitoring and cognitive network optimizer. Furthermore, this deliverable also presents Tutorials and Code samples using the REST APIs for the 5G-MEDIA SDK and SVP. The aim of the 5G-MEDIA SDK Rest APIs is to provide reusable code to the developers as well as customizing the tools based on developers' needs. In this document, all REST APIs that are exposed by the 5G-MEDIA platform tools are listed.

This document is organized as follows. In Section 2, the REST APIs are listed for the SDK tools such as such as serverless framework, private and public catalogues, Open Source Management and Orchestration (OSM), emulating, monitoring, Authentication, Authorization and Accounting (AAA) and cognitive network optimizer (CNO) module. In Section 3, the tutorials with giving some code samples are provided using the SDK REST APIs in order to introduce developers how to use the REST APIs. Following the REST APIs of the SDK, the REST APIs of the SVP are discussed in Section 4. Similar to Section 3, tutorials with some code samples are provided for the SVP REST APIs in Section 5. Finally, the document is concluded in Section 6.

1.1. Scope of the Deliverable

This deliverable provides only the REST APIs of 5G-MEDIA platform including the SDK and the SVP. The 5G-MEDIA platform also offers Javascript wrappers which are functions that are intended to call REST APIs and are all together provided as the 5G-MEDIA WEB APIs. The WEB APIs of the platform are the subject of D5.3. 5G-MEDIA WEB API, Tutorials and Code Samples.

2. SDK Rest 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. One of the important challenges that we encounter in design and development of the 5G-MEDIA SDK [5GMEDIASDK] is the compatibility between the current VM oriented SDK tools, and the novel FaaS approach. In the architecture of 5G-MEDIA platform, we neatly harmonize the two approaches within a single framework which focuses on all phases of a media-type application life-cycle management in the 5G networks. The 5G-MEDIA Emulator leverages the vim emulator (a.k.a, vim-emu/son-emu) which is used for non-FaaS VNFs [5GMEDIASDK] and the FaaS-vim (Lean OW [Lean] running on top of Kubernetes) which is used for the FaaS VNFS [5GMEDIASDK] to equip the developer with an emulated environment. The 5G-MEDIA Emulator mimics the FaaS framework of the SVP, which comprises Apache OW and kubernetes [5GMEDIASDK]. The 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 VNF image.

As mentioned in 5G-MEDIA Deliverable D5.1 Programming Tools, FaaS emulation requires the Lean Openwhisk in the development environment. The developer can use and easily adapt Lean Openwhisk with their web applications or mobile applications using REST API calls as all the capabilities in the system are available through a REST API. There are collection and entity endpoints for actions, triggers, rules, packages, activations, and namespaces. Actions are stateless functions (code snippets) that run on the Openwhisk platform. Actions encapsulate application logic to be executed in response to events and they can be invoked by Openwhisk REST API. Alternatively these actions can be invoked automatically via triggers which are named channels for classes or kinds of events sent from Event sources. The event sources are services that generate events that often indicate changes in data or carry data themselves. Some examples of common Event sources include:

- Messages arriving on Message queues
- Changes in databases
- Website or Web application interactions
- Service APIs being invoked etc..

In Openwhisk rules are used to associate one trigger with one action. After this kind of association is created, each time a trigger event is fired and the action is invoked.

Openwhisk actions, triggers and rules belong in a namespace and optionally a package. Packages can contain actions and feeds. One can let users access entities by granting them

access to the namespace. The fully qualified name of an entity is /<namespace_ID>/<package_name>/<entity_name>.

The names of all entities, including actions, triggers, rules, packages, and namespaces, are a sequence of characters that follow the following format:

- The first character must be an alphanumeric character, or an underscore.
- The subsequent characters can be alphanumeric, spaces, or any of the following values: _, @, ., -.
- The last character can't be a space.

We also note that OpenWhisk has a few system limits, including how much memory an action can use and how many action invocations are allowed per minute. These limitations are given in [OWLimits].
* Required values

Operation	Method	Relative URL	Description
Get all actions	GET	/namespaces/{namespace}/actions	Retrieve information about all FaaS actions. **Path Parameters:** ● namespace (string)* : The entity namespace ● limit (integer) : Number of entities to include in the result (0-200). The default limit is 30. A value of 0 sets limit to the maximum. ● skip (integer): Number of entities to skip in the result.
Get single action information	GET	/namespaces/{namespace}/actions/{actionName}	Retrieve single action information with an action name. **Path Parameters** ● code (boolean) :Include action code in the result ● namespace (string) * :The entity namespace ● actionName (string) *: Name of action to fetch
Create or update an action	PUT	/namespaces/{namespace}/actions/{actionName}	Create a FaaS action with a given action name or update it if it exists **Path Parameters** ● overwrite (string): Overwrite item if it exists. Default is false. Available values : true, false

			• namespace (string) * :The entity namespace
			• actionName (string) * : Name of action to fetch
			The body of the PUT request must contain the FaaS action
			❑ action *: The action being updated or created.
			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
			}
			}

Delete an action	DELETE	/namespaces/{namespace}/actions/{actionName}	Delete an action given with an action name. **Path Parameters:** • namespace (string) * :The entity namespace • actionName (string) * :Name of action to fetch

Invoke an action	POST	/namespaces/{namespace}/actions/{actionName}	Invoke an action which returns an activation ID. In general, an action is invoked in response to an event and produces some observable output. **Path Parameters:** ● namespace (string) *: The entity namespace ● actionName (string) *: Name of action to fetch **Query Parameters:** ● blocking (string): Blocking or non-blocking invocation. Default is non-blocking. Available values : true, false ● result (string): Return only the result of a blocking activation. Default is false. Available values : true, false ● timeout (integer) : Wait no more than specified duration in milliseconds for a blocking response. Default value and max allowed timeout are 60000. The body of the POST request must contain the payload ● payload (body): The parameters for the action being invoked. Parameter content type: application/json
Get an action information inside a package	GET	/namespaces/{namespace}/actions/{packageName}/{actionName}	Retrieve single action information with an action name. **Path Parameters** ● namespace (string) * :The entity namespace ● packageName (string) *: Name of package that contains action ● actionName (string) *: Name of action to fetch **Query Parameters** ● code (boolean) :Include action code in the result
Create or update an action inside a package	PUT	/namespaces/{namespace}/actions/{package	Create a FaaS action inside a package with a given action name or update it if it already exists **Path Parameters**

		Name}/{action Name}	• overwrite (string): Overwrite item if it exists. Default is false. Available values : true, false • namespace (string) * :The entity namespace • packageName(string)*: The name of package that contains action • actionName (string) * : The name of action to fetch The body of the PUT request must contain the FaaS action ❑ action *: The action being updated or created. The example value is given in previous PUT request.
Delete an action inside a package	DELETE	/namespaces/{namespace}/actions/{packageName}/{actionName}	Delete an action inside a package, given with an action name. **Path Parameters:** • namespace (string) * :The entity namespace • packageName(string)*: The name of package that contains action • actionName (string) * :Name of action to fetch
Invoke an action inside a package	POST	/namespaces/{namespace}/actions/{packageName}/{actionName}	Invoke an action inside a package **Path Parameters:** • namespace (string) *: The entity namespace • packageName(string)*: The name of package that contains action • actionName (string) *: Name of action to fetch **Query Parameters:** • blocking (string): Blocking or non-blocking invocation. Default is non-blocking. Available values : true, false • result (string): Return only the result of a blocking activation. Default is false. Available values : true, false • timeout (integer) : Wait no more than specified duration in milliseconds for

			a blocking response. Default value and max allowed timeout are 60000. The body of the POST request must contain the payload ● payload (body): The parameters for the action being invoked. Parameter content type: application/json
Get all rules	GET	/namespaces/{namespace}/rules	Retrieve all rules information with a namespace **Path Parameters** ● namespace (string) * :The entity namespace ● limit (integer): 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 (integer) : Number of entities to skip in the result.
Get rule information	GET	/namespaces/{namespace}/rules/{ruleName}	Retrieve a specific rule with a given rule name **Path Parameters** ● namespace (string) * :The entity namespace ● rule Name (string) *: Name of rule to fetch
Create or update a rule	PUT	/namespaces/{namespace}/rules/{ruleName}	Create a FaaS rule with a given rulename or update it if it already exists **Path Parameters** ● namespace (string) * :The entity namespace ● ruleName (string) * : The name of rule to fetch **Query Parameters** ● overwrite (string): Overwrite them if it exits. Default is false The body of the PUT request must contain the FaaS rule ❑ rule*: The rule being updated or created. Example Value: {

			<pre>"name": "string",
			"version": "string",
			"publish": true,
			"annotations": [
			{
			"key": "string"
			}
			],
			"status": "active",
			"trigger": "string",
			"action": "string"
			}</pre>

Delete a rule	DELETE	/namespaces/{namespace}/rules/{ruleName}	Delete a specific rule with a given rule name. **Path Parameters** ● namespace (string) * :The entity namespace ● ruleName (string) * : The name of rule to delete

Enable or disable a rule	POST	/namespaces/{namespace}/rules/{ruleName}	Enable or disable a specific rule with a given rule name
			Path Parameters:
			● namespace (string) *: The entity namespace
			● ruleName (string) *: Name of rule to update
			
			The body of the POST request must contain the status
			● status (body)* : The parameters for setting status to active or inactive.
			Example value:
			<pre>{
			"status": "inactive"
			}</pre>
			Parameter content type: application/json

Get all triggers	GET	/namespaces/{namespace}/triggers	Retrieve all triggers information with a namespace **Path Parameters** ● namespace (string) * :The entity namespace ● limit (integer): 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 (integer) : Number of entities to skip in the result.
Get trigger information	GET	/namespaces /{namespace} /triggers /{triggerName}	Retrieve a specific trigger with a given trigger name **Path Parameters** ● namespace (string) * :The entity namespace ● trigger Name (string) *: Name of trigger to fetch

Create or update a trigger	PUT	/namespaces /{namespace} /triggers /{triggerName }	Create a FaaS trigger with a given trigger name or update it if it already exists
			Path Parameters
			● namespace (string) * :The entity namespace
			● triggerName (string) * : The name of trigger to update
			Query Parameters
			● overwrite (string): Overwrite them if it exits. Default is false
			The body of the PUT request must contain the FaaS trigger
			❑ trigger*: The trigger being updated or created.
			Example Value:
			{
			"namespace": "string",
			"name": "string",
			"version": "string",
			"publish": true,
			"annotations": [
			{
			"key": "string"
			}
			],
			"parameters": [
			{
			"key": "string"
			}
			],
			"limits": {}
			}

Delete a trigger	DELETE	/namespaces /{namespace} /triggers /{triggerName }	Delete a specific trigger with a given trigger name. **Path Parameters** ● namespace (string) * :The entity namespace ● triggerName (string) * : The name of trigger to delete
Enable or disable a trigger	POST	/namespaces /{namespace} /triggers /{triggerName }	Fire a specific trigger with a given trigger name **Path Parameters:** ● namespace (string) *: The entity namespace ● triggerName (string) *: Name of trigger being fired The body of the POST request must contain the payload ● payload (body)* : The trigger payload Parameter content type: application/json
Get activation summary	GET	/namespaces/ {namespace}/ activations	Get activation summary **Path Parameters:** ● namespace (string) *: The entity namespace **Query Parameters:** ● name (string) : The name of item ● limit (integer): 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 (integer): Number of entities to skip in the result. ● since (integer): Only include entities later than this timestamp (measured in milliseconds since Thu, 01 Jan 1970) ● upto (integer): Only include entities earlier than this timestamp (measured in milliseconds since Thu, 01 Jan 1970) ● docs (boolean): Whether to include full entity description.

Get activation log information	GET	/namespaces /{namespace} /activations /{activationid} /logs	Get activations log information **Path Parameters:** ● namespace (string) *: The entity namespace ● activationid (string)*: The name of activation to fetch
Get activation result.	GET	/namespaces /{namespace} /activations /{activationid} /result	Get activations results information **Path Parameters:** ● namespace (string) *: The entity namespace ● activationid (string)*: The name of activation to fetch
Get all packages	GET	/namespaces /{namespace} /packages	Retrieve information about all packages. **Path Parameters:** ● namespace (string)* : The entity namespace **Query Parameters:** ● limit (integer) : Number of entities to include in the result (0-200). The default limit is 30. A value of 0 sets limit to the maximum. ● skip (integer): Number of entities to skip in the result. ● public (integer): Include publicly shared entities in the result.
Get package information	GET	/namespaces /{namespace} /packages /{packageName}	Retrieve information about a single package. **Path Parameters:** ● namespace (string)* : The entity namespace ● packageName(string)* : The name of package to fetch
Create or update a package	PUT	/namespaces /{namespace} /packages /{packageName}	Create a package with a given package name or update it if it already exists **Path Parameters** ● namespace (string) * :The entity namespace ● packageName (string) * : The name of package to update **Query Parameters**

			• overwrite (string): Overwrite them if it exits. Default is false
			The body of the PUT request must contain the package
			❏ package*: The package being updated or created.
			Example Value:
			{
			"namespace": "string",
			"name": "string",
			"version": "string",
			"publish": true,
			"annotations": [
			{
			"key": "string"
			}
			],
			"parameters": [
			{
			"key": "string"
			}
			],
			"binding": {
			"namespace": "string",
			"name": "string"
			}
			}

Delete a package	DELETE	/namespaces/{namespace}/packages/{packageName}	Delete a specific package with a given package name. **Path Parameters** • namespace (string) * :The entity namespace • packageName(string) * : The name of trigger to delete

Table 1 Serverless Framework REST API

More information on Openwhisk REST APIs is provided in [OWAPI].

2.2. Monitoring Tools

The 5G-MEDIA service monitoring tool guides the media application developer by providing performance data of the emulated service and its components in a quantitative manner during testing. These monitoring tools can gather and centralize monitored metrics into a local database. Metrics can be queried from either VNFs deployed in the emulator or in the Media Service MAPE of SVP.

Container monitoring tool in emulator (i.e cAdvisor [cAdvisor]) gathers metrics related to the compute, storage or network and these metrics are exported by starting a query scheduling. Monitoring metrics of the emulator are then pushed to a Metrics Gateway (i.e Prometheus Push Gateway [PROMETHEUS]) from where the metrics will be pulled by the external Metrics Database (Prometheus database) in the SDK. The gathered metrics can be visualized using the visualization tool (i.e Grafana GUI [Grafana]) which visualizes the inquired metrics from the Metrics Database using a web-based GUI.

More detailed information about service monitoring tools and the workflow can be found in deliverable D5.1 Programming Tools. Monitoring tools consist of cAdvisor (data collector), Prometheus (database), and Grafana (dashboard). The workflow of how monitoring APIs are utilized are given in Figure 1.

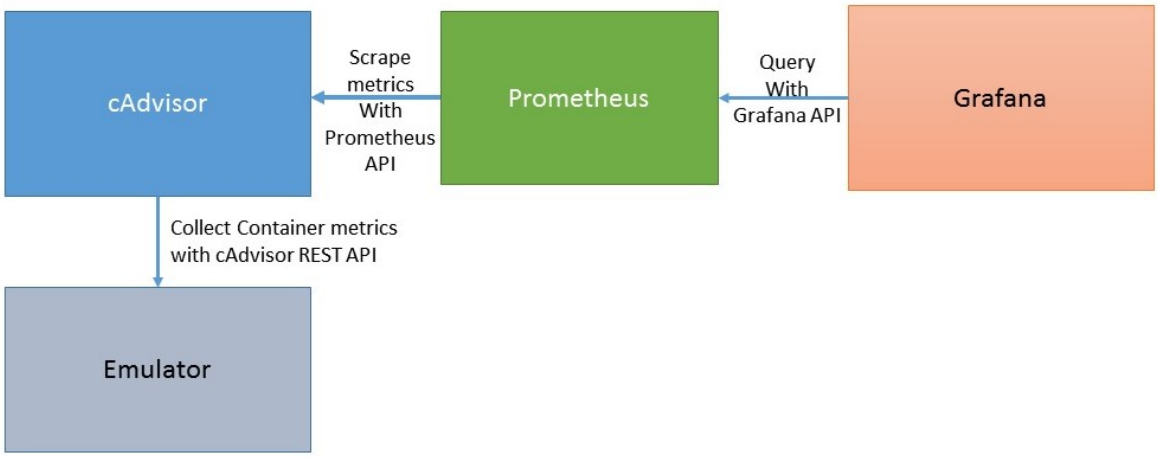

Figure 1 : Monitoring Tools Workflow

As shown in Figure 1, cAdvisor collects container metrics from the emulator environment using cAdvisor REST API. These metrics are then scraped to database with Prometheus API. Finally these metrics can be queried via Grafana REST API and visualised in a dashboard. In the following sections, REST API lists for monitoring tools are provided.

2.2.1. Grafana REST API

Grafana is the open source analytics & monitoring solution for every database. The open observability platform Grafana is the open source analytics & monitoring solution for every database. More information on Grafana can be found in deliverable *D5.1 Programming Tools* [D5.1]. The Grafana backend exposes an HTTP API, the same API is used by the frontend to do everything from saving dashboards and updating data sources. In this section we list the main functionalities of Grafana REST APIs that are utilized in 5G-MEDIA SDK. The Full list of Grafana REST APIs can be found in [Grafana]

Operation	Method	Relative URL	Description
Authentication	POST	/api/auth/keys	Creates Grafana authentication token. **Query Parameters:** ● username(username): username ● password(password): password ● name(body param): API key name ● role(body param): role
Dashboard Operation	POST	/api/dashboards /db	Creates a new dashboard or updates an existing dashboard **Query Parameters** ● dashboard(*JSON)* JSON Body Schema with following keys: ● dashboard - The complete dashboard model ● dashboard.id - null to create a new dashboard ● dashboard.uid - Optional unique identifier when creating a new dashboard. uid = null will generate a new uid ● folder Id - The id of the folder to save dashboard in. ● overwrite - Set true if you want to overwrite existing dashboard with newer version. Same dashboard title in folder or same dashboard uid. ● message - a Set commit message for the version history
Get data	GET	/api/datasources /{datasourceId}	Gets a single data source by Id **Path Parameters** ● datasourceId(*integer) :* ○ -1: default value for permissions of users with the Viewer and Editor roles. ○ 1: View ○ 2: Edit ○ 4: Admin

Delete dashboard	DELETE	/api/dashboards /uid/{uid}	Deletes a dashboard given the specified unique identifier uid **Path Parameters** ● uid(*string):* Specified unique identifier

Table 2 Grafana REST API

2.2.2. Prometheus API

Prometheus collects metrics from monitored targets by scraping metrics HTTP endpoints on these targets. Since Prometheus also exposes data in the same manner about itself, it can also scrape and monitor its own health. More information on Prometheus can be found in *D5.1 Programming Tools* [D5.1]. The current stable HTTP API of Prometheus is reachable under /api/v1 on a Prometheus server. Any non-breaking additions will be added under that endpoint. In addition to GET methods, you can also URL-encode these parameters directly in the request body by using the POST method. This is useful when specifying a large query that may breach server-side URL character limits. In this section we list the main functionalities of Prometheus REST APIs that are utilized in 5G-MEDIA SDK. The Full list of Prometheus REST APIs can be found in [PROMETHEUS]

Operation	Method	Relative URL	Description
Get data	GET	/api/v1/query	Retrieve data by query **Query Parameters** ● query(*string):* Prometheus expression query string.

Get data	POST	/api/v1/query	Encode parameters in the request body to get data by query
			● The body of the POST request must contain query JSON object
			
			query (body) : The query as JSON
			{
			"query": "string"
			}

Get query	GET	/api/v1/query_range	Retrieve query over a range of time **Query Parameters** ● query(string) : Prometheus expression query string.

			• start (unix timestamp): Start timestamp • end (unix timestamp): End timestamp • step (float): Query resolution step width in duration, float number of seconds

Get query	POST	/api/v1/query_range	Encode parameters in the request body to get data by query over a range of time
			
			The body of the POST request must contain query JSON object
			
			• query (body): The query as JSON
			{
			"query": "string"
			“start” : “unix_timestamp”
			“end” : “unix_timestamp”
			“step“ : “float”
			}

Table 3 Prometheus REST API

2.2.3. cAdvisor API

cAdvisor (Container Advisor) provides container users an understanding of the resource usage and performance characteristics of their running containers. It is a running daemon that collects, aggregates, processes, and exports information about running containers. Specifically, for each container it keeps resource isolation parameters, historical resource usage, histograms of complete historical resource usage and network statistics. This data is exported by container and machine-wide. More information on cAdvisor can be found in *D5.1 Programming Tools* [D5.1]. cAdvisor exposes its raw and processed stats via a versioned remote REST API which are listed in [cAdvisor].

Operation	Method	Relative URL	Description
Get machine information	GET	/api/v2.0/machine	Retrieve machine information where cAdvisor is working
Get container information	GET	/api/v1.2/docker/{containerName}	Retrieve the information of the specified container(s) **Path Parameters**

			• containerName(*string):* Name of the container to have the information. Blank for all containers information
Get events	GET	/api/v2.0/events /{containerName}	Retrieve a list of events **Path Parameters** • containerName(string) *: Name of the container to have the information. Blank for all containers information **Query Parameters** • start_time(date): Start time of the events to query. Default: Beginning of time (if stream is false) • end_time(date): End time of the events to query. Default: Now (if stream is false) • stream (Boolean): Whether to stream new events as they occur. If false returns historical events. Default: false • max_events (integer): The max number of events to return (if stream is false)

Table 4 cAdvisor REST API

2.3. Vim-Emulator

The 5G-MEDIA Emulator facilitates local prototyping and testing of NSs in realistic end-to-end multi-PoP scenarios in multi-vim environments. This platform allows execution of real network functions packaged as Docker containers in emulated network topologies running locally on the developer's machine. The 5G-MEDIA Emulator 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* [D5.1]

vim-emu also known as son-emu REST API is mainly used to control a deployed service in son-emu/vim-emu emulator environment. This section gives an overview of implemented REST API of non-FaaS emulator environment which can also be found in [ETSIVIM].

Operation	Method	Relative URL	Description
List datacenters	GET	/restapi/datacenter	Retrieve a list of all datacenters
Get information of container	GET	/restapi/datacenter/{dc_label}	Returns the information of the specified container(s) **Path Parameters** ● dc_label(*string):* label of the datacenter to list
List computation metrics	GET	/restapi/compute	Retrieve a list of all computation metrics
Get information of datacenter	GET	/restapi/compute/{dc_label}	Retrieve a list of computation metrics for a specific datacenter **Path Parameters** ● dc_label(string): label of the datacenter to list
Get information of datacenter	GET	/restapi/compute/{dc_label}/{compute_name}	Retrieve a list of computation metrics for a specific datacenter **Path Parameters** ● dc_label(string): label of the datacenter ● compute_name *(string*): label of the vnfs
Delete VNF instance	DELETE	/restapi/compute/{dc_label}/{compute_name}	Delete the vnf instance **Path Parameters** ● dc_label(string): label of the datacenter to list ● compute_name *(string*): label of the vnfs to list the computation metrics
Create chains	PUT	/restapi/network	Add chains between VNFs **Query Parameters** ● vnf_src_name(string): VNF name of the source of the link ● vnf_dst_name(string): VNF name of the destination of the link

			• vnf_src_interface (string): VNF interface name of the source of the link • vnf_dst_interface (string): VNF interface name of the destination of the link • weight(string): weight of the link (can be useful for routing calculations)
Create an action to monitor VNF	PUT	/restapi/monitor/interface	Create the action that monitors the counters of a VNF interface **Query Parameters** • vnf_name(string): Name of the VNF to be monitored • vnf_interface(string): Name of the VNF interface to be monitored • metric (string): One of the following metrics:[tx_bytes, rx_bytes, tx_packets, rx_packets]
Delete an action	DELETE	/restapi/monitor/interface	Delete the action that monitors the counters of a VNF interface **Query Parameters** • vnf_name(string): Name of the VNF to be monitored • vnf_interface(string): Name of the VNF interface to be monitored • metric (string): One of the following metrics:[tx_bytes, rx_bytes, tx_packets, rx_packets]

Table 5 Vim-emu REST API

2.4. Private Catalogue

The REST APIs of the private catalogue have been provided in as the 5G App and Service Catalogue NBI in Deliverable *D4.1 5G-MEDIA Catalogue APIs and Network Apps* [D4.1].

2.5. Cognitive Network Optimizer

Integration of Cognitive Network Optimizer (CNO) in the SDK includes the configuration of CNO Training and Deploying the Trained Model to the SVP. Using this tool, the developer can configure a training model, view the training performance and if the model achieves desirable performance, the developer can deploy it to SVP in order to have it utilized by the CNO. CNO

and the use of reinforcement learning algorithms are well documented in Deliverable *5G-MEDIA Operations and Configuration Platform* [D3.4]. The training model is configured with the following:

rewardFunction	The variants of reward function that describes how the agent "ought" to behave (e.g. “bls” considers bitrate, lossrate and smoothness, “bl” considers bitrate and “b” considers bitrate only)
alpha	A weight factor for the loss rate in the reward function, default =500.
traffic_pattern	Shape/pattern of background traffic, e.g., it could be sawtooth-like or with no shape (random), default = sawtooth
actorLearningRate	Learning rate of “Actor Network” which updates the policy distribution, default= 0.0001
criticLearningRate	Learning rate of “Critic Network” which estimates the value function, default=0.001
linkCapacity	This is related to the maximum link capacity of an emulated network path, default = 20 (Mbps)
uploadFileName	The zip file that includes trained data

parallelAgent	Number of parallel agents. The maximum value of this parameter is depended on available CPUs, default=1
nameNnModel	A name for a trained model (.ckpt.*) to use at the SVP

Table 6 CNO Training GUI Configurations Parameters

The CNO capabilities in the SDK are available through REST API calls which are given as follows

Operation	Method	Relative URL	Description

Start training	POST	/trainData	Start Training Model for CNO in the SDK
			
			The body of the POST request must contain query JSON object of train model:
			
			● trainModel (*Object):* 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”
			}
			}

Terminate training	GET	/shutdown	Terminate running training

Upload data	POST	/uploadZipFile	Upload Training Data The body of the POST request must contain ZIP file • file(File): The zip file that includes all training data to be used
Start tensorboard	GET	/tensorboard	Start tensorboard
Deploy model	POST	/deployMape	Deploy the latest training model to SVP MAPE The body of the POST request must contain: • pemFile (file) : File for permission to access to the server • ip(string) : IP address of the production environment to deploy the training model

Table 7 CNO REST API

3. SDK Rest APIs Tutorials and Code Samples

3.1. Serverless Framework

- Code sample to get all created actions for use-case 1.

```
curl -X GET "IP_ADDRESS:PORT/api/v1/namespaces/use-case1/actions" -H "accept: application/json"
```

```
[
  {
    "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
  }
]
```


- Code sample to get information of usecase1_action created for use-case 1

```
curl -X GET
"IP_ADDRESS:PORT/api/v1/namespaces/usecase1/actions/usecase1_action?code=true" -H
"accept: application/json"
```


```
{
 "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
}
```


- Code sample to create usecase1-action for use-case 1

```
curl -X PUT "IP_ADDRESS:PORT/api/v1/namespaces/usecase1/actions/usecase1-action?overwrite=true" -H "accept: application/json" -H "Content-Type: application/json" -d "{ \"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 }}"
```


```
{
 "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
}
```


- Code sample to delete usecase1-action

```
curl -X DELETE "IP_ADDRESS:PORT/api/v1/namespaces/usecase1/actions/usecase1_action" -H "accept: application/json"
```

```
200 Deleted Item
```

- Blocking usecase1-action for use-case 1 and return the result of blocking activation

```
curl -X POST "IP_ADDRESS:PORT/api/v1/namespaces/use-case1/actions/usecase1-action?blocking=true&result=true" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"
```


```
{
 "activationId": "string"
}
```


- Get information usecase1_action inside usecase1_package for use-case 1

```
curl -X GET "IP_ADDRESS:PORT/api/v1/namespaces/usecase1/actions/usecase1_package/usecase1_action?code=true" -H "accept: application/json"
```


```
{
 "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
}
```


- Code sample to create usecase1-action under usecase1_package for use-case 1

```
curl -X PUT "IP_ADDRESS:PORT/api/v1/namespaces/usecase1/actions/usecase1_package/usecase1_action?overwrite=true" -H "accept: application/json" -H "Content-Type: application/json" -d "{ \"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 }}"
```


```
{
 "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
}
```


- Code sample to delete usecase1-action under usecase1_package

```
curl -X DELETE "IP_ADDRESS:PORT/api/v1/namespaces/usecase1/actions/usecase1_package/usecase1_action" -H "accept: application/json"
```

```
200 Deleted Item
```

- Blocking usecase1-action under usecase1_package and return the result of blocking activation

```
curl -X POST "IP_ADDRESS:PORT/api/v1/namespaces/usecase1/actions/usecase1_package/usecase1_a
```

ction?blocking=true&result=true" -H "accept: application/json" -H "Content-Type: application/json" -d "{}"
200 Successful activation

- Code sample to create usecase1-action with extension under usecase1_package

curl -X GET "IP_ADDRESS:PORT/api/v1/namespaces/usecase1/packages?public=true" -H "accept: application/json"


```
[
 {
  "namespace": "string",
  "name": "string",
  "version": "string",
  "publish": true,
  "annotations": [
   {
    "key": "string"
   }
  ],
  "parameters": [
   {
    "key": "string"
   }
  ],
  "binding": {
   "namespace": "string",
   "name": "string"
  },
  "actions": [
   {
    "name": "string",
    "version": "string",
    "annotations": [
     {
      "key": "string"
     }
    ],
    "parameters": [
     {
      "key": "string"
     }
    ]
   }
  ],
  "feeds": [
```

```
    {}
   ],
   "updated": 0
  }
]
```

3.2. Monitoring Tools

3.2.1. Grafana Code Samples

- Code sample given below is to create a new dashboard or update an existing dashboard.

```
curl -X POST -H 'Authorization: Bearer
eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk' -H "Content-type:
application/json" -d '{
  "dashboard": {
        "id": null,
        "uid": null,
        "title": "Production Overview",
        "tags": [ "templated" ],
        "timezone": "browser",
        "schemaVersion": 16,
        "version": 0
  },
  "folderId": 0,
  "overwrite": false
}' 'localhost/api/dashboards/db HTTP/1.1'
```

```
{
  "id": 1,
  "uid": "cIBgcSjkk",
  "url": "/d/cIBgcSjkk/production-overview",
  "status": "success",
  "version": 1,
  "slug": "production-overview"
}
```

- Code sample below will return the dashboard given the dashboard unique identifier (uid).

```
curl -X GET -H 'Authorization: Bearer
eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk' -H "Content-type:
application/json" 'localhost/api/dashboards/uid/:uid HTTP/1.1'
```

```
{
  "dashboard": {
    "id": 1,
    "uid": "cIBgcSjkk",
    "title": "Production Overview",
    "tags": [
      "templated"
    ],
    "timezone": "browser",
    "schemaVersion": 16,
    "version": 0
  },
  "meta": {
    "isStarred": false,
    "url": "/d/cIBgcSjkk/production-overview",
    "slug": "production-overview"
  }
}
```

- Code sample below will delete the dashboard given the specified uid.

```
curl -X DELETE -H 'Authorization: Bearer
eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk' -H "Content-type:
application/json" 'localhost/api/dashboards/uid/:uid  HTTP/1.1'
```

```
{
  "title": "Production Overview"
}
```

- Code sample is given below to get permissions for a dashboard

curl -X GET -H 'Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk' -H "Content-type: application/json" 'localhost/api/dashboards/id/:dashboardId/permissions HTTP/1.1'


```
[
 {
  "id": 1,
  "dashboardId": -1,
  "created": "2017-06-20T02:00:00+02:00",
  "updated": "2017-06-20T02:00:00+02:00",
  "userId": 0,
  "userLogin": "",
  "userEmail": "",
  "teamId": 0,
  "team": "",
  "role": "Viewer",
  "permission": 1,
  "permissionName": "View",
  "uid": "",
  "title": "",
  "slug": "",
  "isFolder": false,
  "url": ""
 },
 {
  "id": 2,
  "dashboardId": -1,
  "created": "2017-06-20T02:00:00+02:00",
  "updated": "2017-06-20T02:00:00+02:00",
  "userId": 0,
  "userLogin": "",
  "userEmail": "",
  "teamId": 0,
  "team": "",
  "role": "Editor",
  "permission": 2,
  "permissionName": "Edit",
  "uid": "",
  "title": "",
  "slug": "",
  "isFolder": false,
  "url": ""
 }
]
```

- The code sample given below is to update permissions for a dashboard

```
curl -X POST -H 'Authorization: Bearer
eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk' -H "Content-type:
application/json" -d '{
  "items": [
        {
        "role": "Viewer",
        "permission": 1
        },
        {
        "role": "Editor",
        "permission": 2
        },
        {
        "teamId": 1,
        "permission": 1
        },
        {
        "userId": 11,
        "permission": 4
        }
  ]
}' 'localhost/api/dashboards/id/1/permissions'
```

```
{
  "message": "Dashboard permissions updated"
}
```

- The code sample given below is to get a single datasource by id

```
curl -XGET -H 'Authorization: Bearer
eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk' -H "Content-type:
application/json" 'localhost/api/datasources/:datasourceId'
```

```
{
  "id": 1,
  "orgId": 1,
```

```
  "name": "test_datasource",
  "type": "graphite",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "isDefault": false,
  "jsonData": null
}
```

3.2.2. Prometheus Code Samples

- The following code sample evaluates an instant query “up” at a single point in time: 2015-07-01T20:10:51.781Z

```
curl -X GET -H "Content-type: application/json"
'http://localhost:9090/api/v1/query?query=up&time=2015-07-01T20:10:51.781Z'
```

```
{
  "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"
      ]
     }
   ]
  }
}
```


- The following example evaluates the expression up over a 30-second range with a query resolution of 15 seconds.

```
curl -X GET -H "Content-type: application/json"
'http://localhost:9090/api/v1/query_range?query=up&start=2015-07-01T20:10:30.781Z&end=2015-07-01T20:11:00.781Z&step=15s'
```


```
{
  "status": "success",
  "data": {
    "resultType": "matrix",
    "result": [
      {
        "metric": {
          "__name__": "up",
          "job": "prometheus",
          "instance": "localhost:9090"
        },
        "values": [
          [
            1435781430.781,
            "1"
          ],
          [
            1435781445.781,
            "1"
          ],
          [
            1435781460.781,
            "1"
          ]
        ]
      },
      {
        "metric": {
          "__name__": "up",
          "job": "node",
```

```
      "instance": "localhost:9091"
     },
     "values": [
      [
       1435781430.781,
       "0"
      ],
      [
       1435781445.781,
       "0"
      ],
      [
       1435781460.781,
       "1"
      ]
     ]
    }
   ]
  }
}
```


3.2.3. cAdvisor Code Samples

- The code sample given below is to get machine information where cAdvisor is working

```
curl -X GET -H "Content-type: application/json" 'http://IP_ADDRESS:PORT/api/v2.0/machine'
```


```
{
 "num_cores": 2,
 "cpu_frequency_khz": 2095076,
 "memory_capacity": 16826998784,
 "hugepages": [
  {
   "page_size": 2048,
   "num_pages": 0
  }
 ],
 "machine_id": "2878e88e46be77ef2edbf1455d651ad5",
 "system_uuid": "DBD878DC-229D-46AC-9608-3300CB3E0705",
 "boot_id": "e0e70c7d-bb7f-4f82-8981-0ced6bfbc1c8",
 "filesystems": [
  {
   "device": "/dev/mapper/vm--kiwi--vg-root",
   "capacity": 103783211008,
   "type": "vfs",
   "inodes": 6447104,
   "has_inodes": true
  },
```

```
  {
   "device": "/dev/sda1",
   "capacity": 754434048,
   "type": "vfs",
   "inodes": 46848,
   "has_inodes": true
  },
  {
   "device": "shm",
   "capacity": 67108864,
   "type": "vfs",
   "inodes": 2054077,
   "has_inodes": true
  },
  {
   "device": "overlay",
   "capacity": 103783211008,
   "type": "vfs",
   "inodes": 6447104,
   "has_inodes": true
  },
  {
   "device": "tmpfs",
   "capacity": 67108864,
   "type": "vfs",
   "inodes": 2054077,
   "has_inodes": true
  }
 ],
 "disk_map": {
  "252:0": {
   "name": "dm-0",
   "major": 252,
   "minor": 0,
   "size": 105574825984,
   "scheduler": "none"
  },
  "252:1": {
   "name": "dm-1",
   "major": 252,
   "minor": 1,
   "size": 1023410176,
   "scheduler": "none"
  },
  "8:0": {
   "name": "sda",
   "major": 8,
   "minor": 0,
   "size": 107374182400,
   "scheduler": "deadline"
  }
 },
```

```
"network_devices": [
 {
  "name": "br-357c161c9175",
  "mac_address": "02:42:4c:9e:46:40",
  "speed": 0,
  "mtu": 1500
 },
 {
  "name": "br-5e8571a35e0a",
  "mac_address": "02:42:df:bd:2d:0d",
  "speed": 0,
  "mtu": 1500
 },
 {
  "name": "br-d9468dd60980",
  "mac_address": "02:42:21:49:22:e8",
  "speed": 0,
  "mtu": 1500
 },
 {
  "name": "br-e1c19f042066",
  "mac_address": "02:42:d1:6a:c2:ba",
  "speed": 0,
  "mtu": 1500
 },
 {
  "name": "enp0s3",
  "mac_address": "08:00:27:40:1c:0e",
  "speed": 1000,
  "mtu": 1500
 },
 {
  "name": "lxdbr0",
  "mac_address": "fe:b7:98:2f:53:8e",
  "speed": 0,
  "mtu": 1500
 }
],
"topology": [
 {
  "node_id": 0,
  "memory": 16826998784,
  "cores": [
   {
    "core_id": 0,
    "thread_ids": [
     0
    ],
    "caches": [
     {
      "size": 32768,
      "type": "Data",
```

```
          "level": 1
        },
        {
          "size": 32768,
          "type": "Instruction",
          "level": 1
        },
        {
          "size": 1048576,
          "type": "Unified",
          "level": 2
        },
        {
          "size": 11534336,
          "type": "Unified",
          "level": 3
        }
      ]
    },
    {
      "core_id": 1,
      "thread_ids": [
        1
      ],
      "caches": [
        {
          "size": 32768,
          "type": "Data",
          "level": 1
        },
        {
          "size": 32768,
          "type": "Instruction",
          "level": 1
        },
        {
          "size": 1048576,
          "type": "Unified",
          "level": 2
        },
        {
          "size": 11534336,
          "type": "Unified",
          "level": 3
        }
      ]
    }
  ],
  "caches": null
 }
],
"cloud_provider": "Unknown",
```

```
 "instance_type": "Unknown",
 "instance_id": "None"
}
```


- The code sample given below is to get information of a specified container.

```
curl -X GET -H "Content-type: application/json"
'http://127.0.0.1:9911/api/v1.2/events/redis'
```


```
{
 "/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": [
   {
    "timestamp": "2019-11-11T07:11:51.015754247Z",
    "cpu": {
     "usage": {
      "total": 3461280500296,
      "per_cpu_usage": [
       1928271364140,
```

```
        1533009136156
       ],
       "user": 438050000000,
       "system": 833930000000
      },
      "cfs": {
       "periods": 0,
       "throttled_periods": 0,
       "throttled_time": 0
      },
      "schedstat": {
       "run_time": 0,
       "runqueue_time": 0,
       "run_periods": 0
      },
      "load_average": 0
     },
     "diskio": {
      "io_service_bytes": [
       {
        "device": "/dev/sda",
        "major": 8,
        "minor": 0,
        "stats": {
         "Async": 8032256,
         "Read": 8032256,
         "Sync": 0,
         "Total": 8032256,
         "Write": 0
        }
       },
       {
        "device": "/dev/dm-0",
        "major": 252,
        "minor": 0,
        "stats": {
         "Async": 8032256,
         "Read": 8032256,
         "Sync": 0,
         "Total": 8032256,
         "Write": 0
        }
       }
      ],
      "io_serviced": [
       {
        "device": "/dev/sda",
        "major": 8,
        "minor": 0,
```

```
        "stats": {
          "Async": 259,
          "Read": 259,
          "Sync": 0,
          "Total": 259,
          "Write": 0
        }
       },
       {
        "device": "/dev/dm-0",
        "major": 252,
        "minor": 0,
        "stats": {
          "Async": 259,
          "Read": 259,
          "Sync": 0,
          "Total": 259,
          "Write": 0
        }
       }
     ]
    }
```


3.3. Vim-Emulator

- Code sample below will return the datacenter information

```
curl -X GET http://127.0.0.1:5001/restapi/datacenter
```


```
[
 {
  "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"
 },
 {
  "internalname": "dc1",
```

```
  "label": "datacenter1",
  "metadata": {},
  "n_running_containers": 0,
  "switch": "dc1.s1"
 },
 {
  "internalname": "dc4",
  "label": "datacenter4",
  "metadata": {
   "mydata": "we can also add arbitrary metadata to each DC"
  },
  "n_running_containers": 0,
  "switch": "dc4.s1"
 }
]
```


- Code sample below will return a specific datacenter information

```
curl -X GET http://127.0.0.1:5001/restapi/datacenter/datacenter1
```


```
[
 {
  "internalname": "dc1",
  "label": "datacenter1",
  "metadata": {},
  "n_running_containers": 0,
  "switch": "dc1.s1"
 }
]
```


- Code sample below will list all computational metrics

```
curl -X GET http://127.0.0.1:5001/restapi/compute
```


```
[
 [
  "vnf2":
  {
   "cpu_period": null,
   "cpu_quota": -1,
   "cpu_shares": null,
   "cpuset": null,
```

```
    "datacenter": "cvim1",
    "docker_network": "172.17.0.3",
    "flavor_name": "tiny",
    "id": "41a22d8621a9de28f849df07141b491c47842f4b3238080558089e560ed37e42",
    "image": "ubuntu",
    "mem_limit": null,
    "memswap_limit": null,
    "name": "vnf2",
    "network": [
     {
      "dc_portname": "dc1.s1-eth2",
      "intf_name": "vnf2-eth0",
      "ip": "10.0.0.4",
      "mac": "ba:68:19:2e:f6:f9",
      "status": "MISSING",
      "up": false
     }
    ],
    "short_id": "41a22d8621a9",
    "state": {
     "Dead": false,
     "Error": "",
     "ExitCode": 0,
     "FinishedAt": "0001-01-01T00:00:00Z",
     "OOMKilled": false,
     "Paused": false,
     "Pid": 15400,
     "Restarting": false,
     "Running": true,
     "StartedAt": "2017-03-23T15:32:19.127081166Z",
     "Status": "running"
    }
   }
  ],
  [
   "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"
    }
   }
 ]
]
```


- Code sample below will list computation metrics for a specific datacenter.

```
curl -X GET http://127.0.0.1:5001/restapi/compute/cvim1/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"
  }
}
```


3.4. Private Catalogue

- The following code sample is to get information about multiple NS descriptor resources

```
curl -X GET --header 'Accept: application/json'
'http://IP_ADDRESS:PORT/nsd/v1/ns_descriptors'
```


```
[
 {
  "id": "691a15dc-3419-4e99-aa74-62f43643b4ed",
  "nsdId": "622bc54d-3e4c-4304-b1a2-3704666f835a",
  "nsdName": "pingpong_nsd",
  "nsdVersion": "1.1",
  "nsdDesigner": "OSM",
  "nsdInvariantId": "622bc54d-3e4c-4304-b1a2-3704666f835a",
  "vnfPkgIds": [
   "1621907a-1e0c-4d74-a004-ffae8d902e44",
   "d9fb09be-156e-4cb1-952c-6387a279e57f"
  ],
  "pnfdInfoIds": [],
  "nestedNsdInfoIds": [],
  "nsdOnboardingState": "ONBOARDED",
  "onboardingFailureDetails": null,
  "nsdOperationalState": "ENABLED",
  "nsdUsageState": "NOT_IN_USE",
  "userDefinedData": {},
  "_links": {
```

```
    "self": "/nsd/v1/ns_descriptors/691a15dc-3419-4e99-aa74-62f43643b4ed",
    "nsd_content": "/nsd/v1/ns_descriptors/691a15dc-3419-4e99-aa74-62f43643b4ed/nsd_content"
   },
   "manosOnboardingStatus": {
    "NET_OSMR4.0": "ONBOARDED"
   },
   "c2cOnboardingState": "UNPUBLISHED"
  },
  {
   "id": "6f7ebae1-29b1-42bd-a52b-a60cb485ba7e",
   "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"
  },
  {
   "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"
 },
 {
  "id": "dd0b66a3-96f3-4e76-ad67-7fef6d6feb47",
  "nsdId": "e9dfab2e-f437-401f-b1e9-7b2c3f105de4",
  "nsdName": "vcache_nsd",
  "nsdVersion": "1.1",
  "nsdDesigner": "NXW",
  "nsdInvariantId": "e9dfab2e-f437-401f-b1e9-7b2c3f105de4",
  "vnfPkgIds": [
   "7580a9da-9b68-48b6-8d24-47dfe704f9fc",
   "21480b3b-8763-4163-8eda-da702c9e3b2b"
  ],
  "pnfdInfoIds": [],
  "nestedNsdInfoIds": [],
  "nsdOnboardingState": "ONBOARDED",
  "onboardingFailureDetails": null,
  "nsdOperationalState": "ENABLED",
```

```
  "nsdUsageState": "NOT_IN_USE",
  "userDefinedData": {},
  "_links": {
   "self": "/nsd/v1/ns_descriptors/dd0b66a3-96f3-4e76-ad67-7fef6d6feb47",
   "nsd_content": "/nsd/v1/ns_descriptors/dd0b66a3-96f3-4e76-ad67-7fef6d6feb47/nsd_content"
  },
  "manosOnboardingStatus": {
   "NET_OSMR4.0": "ONBOARDED"
  },
  "c2cOnboardingState": "UNPUBLISHED"
 }
]
```


- The following code sample is to create a new NS descriptor resource

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -
d '{
 "userDefinedData": {}
}' 'http://IP_ADDRESS:PORT/nsd/v1/ns_descriptors'
```


```
{
 "id": "a3436820-4d62-457f-a0b1-2ccdb4e3420a",
 "nsdId": null,
 "nsdName": null,
 "nsdVersion": null,
 "nsdDesigner": null,
 "nsdInvariantId": null,
 "vnfPkgIds": [],
 "pnfdInfoIds": [],
 "nestedNsdInfoIds": [],
 "nsdOnboardingState": "CREATED",
 "onboardingFailureDetails": null,
 "nsdOperationalState": "DISABLED",
 "nsdUsageState": "NOT_IN_USE",
 "userDefinedData": {},
 "_links": {
  "self": "/nsd/v1/ns_descriptors/a3436820-4d62-457f-a0b1-2ccdb4e3420a",
  "nsd_content": "/nsd/v1/ns_descriptors/a3436820-4d62-457f-a0b1-2ccdb4e3420a/nsd_content"
 },
 "manosOnboardingStatus": {},
 "c2cOnboardingState": "UNPUBLISHED"
}
```


- The code sample below is to delete a NS descriptor resource

```
curl -X DELETE --header 'Accept: application/json'
'http://IP_ADDRESS:PORT/nsd/v1/ns_descriptors/a3436820-4d62-457f-a0b1-
2ccdb4e3420a'
```

```
204
```

- The code sample given below is to query all VNF Packages info that are onboarded into the private catalogue

```
curl -X GET --header 'Accept: application/json'
'http://10.254.182.155:8083/vnfpkgm/v1/vnf_packages'
```


```
[
 {
  "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"
 },
 {
  "id": "44aa96b6-2c55-4d20-a7c0-faca82d492d2",
  "vnfdId": "7d8b84df-c536-443f-b73e-2f1aeb848fd6",
  "vnfProvider": "IBM",
  "vnfProductName": "faas_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/44aa96b6-2c55-4d20-a7c0-faca82d492d2",
   "vnfd": "/vnfpkgm/v1/vnf_packages/44aa96b6-2c55-4d20-a7c0-faca82d492d2/vnfd",
   "packageContent": "/vnfpkgm/v1/vnf_packages/44aa96b6-2c55-4d20-a7c0-
faca82d492d2/package_content"
  },
  "manosOnboardingStatus": {
   "NET_OSMR4.0": "ONBOARDED"
  },
  "c2cOnboardingState": "UNPUBLISHED"
 }
]
```


- The code sample given below is create a VNF Package resource

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -
d '{
 "userDefinedData": {}
}' 'http://IP_ADDRESS:PORT/vnfpkgm/v1/vnf_packages'
```


```
{
 "id": "4af66182-865c-45c9-8104-ad155dcd4fb9",
 "vnfdId": null,
 "vnfProvider": null,
 "vnfProductName": null,
 "vnfSoftwareVersion": null,
 "vnfdVersion": null,
```

```
 "checksum": null,
 "softwareImages": null,
 "additionalArtifacts": null,
 "onboardingState": "CREATED",
 "operationalState": "DISABLED",
 "usageState": "NOT_IN_USE",
 "userDefinedData": {},
 "_links": {
  "self": "/vnfpkgm/v1/vnf_packages/4af66182-865c-45c9-8104-ad155dcd4fb9",
  "vnfd": "/vnfpkgm/v1/vnf_packages/4af66182-865c-45c9-8104-ad155dcd4fb9/vnfd",
  "packageContent": "/vnfpkgm/v1/vnf_packages/4af66182-865c-45c9-8104-
ad155dcd4fb9/package_content"
 },
 "manosOnboardingStatus": {},
 "c2cOnboardingState": "UNPUBLISHED"
}
```


- The code given below is to delete a VNF package that was onboarded to the private catalogue

```
curl -X DELETE --header 'Accept: application/json'
'http://IP_ADDRESS:PORT/vnfpkgm/v1/vnf_packages/4af66182-865c-45c9-8104-
ad155dcd4fb9'
```

```
204
```

3.5. Cognitive Network Optimizer

- The following sample is to upload data set for training

```
curl -X POST -H  "Content-type: multipart/form-data" -d './cooked_traces/'
IP_ADDRESS:PORT/uploadZip'
```

```
200 OK
```

- The following sample is to start reinforcement training to build a training model for CNO.

```
curl -X POST -H  "Content-type: application/json" -d '{
 "trainModel": {
        "alpha": “0.5”,
        "trafficPattern": “random”,
        "actorLearningRate": "0.0001",
        "criticLearningRate": “0.001”,
        "linkCapacity": "20",
        "rewardFunction": “bls”,
        "uploadFileName": “./cooked_traces/”,
        "parallelAgent": “2”
 },
 "folderId": 0,
 "overwrite": false
}' IP_ADDRESS:PORT/trainData'
```

```
200 OK
```

- The following sample is to stop reinforcement training to finalize the training model.

```
curl -X GET IP_ADDRESS:PORT/shutdown'
```

```
200 OK
```

- The following sample is to start tensorboard to visualize the reinforcement learning process. These graphics are used to decide whether training model is good enough to deploy to the production environment.

```
curl -X GET IP_ADDRESS:PORT/tensorboard'
```

```
200 OK
```

- The following sample is to deploy the latest generated training model file (with .ckpt.meta extension) into the SVP.

curl -X POST -H "Content-type: multipart/form-data" -F 'ip:X.X.X.X' -F 'pemFile:@path/to/local/file' IP_ADDRESS:PORT/deployMape'
200 OK

4. SVP Rest APIs Definition

4.1. Serverless Framework

In the 5G-MEDIA platform, offers 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.

While the SDK uses a version of Apache Openwhisk with a small footprint that is referred as Lean Openwhisk, however the semantic flow of the Openwhisk that is used in SVP is identical. Therefore, the REST APIs referred in Section 2.1 is also valid for the Serverless Framework in the SVP.

4.2. Public Catalogue

While the 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 REST API of the public catalogue is same as the one for the private catalogue and has been provided in as the 5G App and Service Catalogue NBI in Deliverable *D4.1 5G-MEDIA Catalogue APIs and Network Apps* [D4.1].

4.3. Open Source Management and Orchestration

ETSI Open Source Management and Orchestration (OSM) [OSM] is an operator-led ETSI community that is delivering a production-quality open source Management and

Orchestration (MANO) stack aligned with ETSI NFV Information Models and that meets the requirements of production NFV networks.

The 5G-MEDIA platform utilizes the OSM environment so that the developer can manage emulation environments such as vim-emu and Lean OW. The OSM is the main UI for managing the lifecycle operations on applications such as their instantiation and termination in both SDK and SVP. More information on OSM can be found in Deliverable *D5.1 Programming Tools* [D5.1].

The 5G-MEDIA platform exploits OSM Release 5. This release expands the cloud-native, dockerized OSM installation with the kafka bus for asynchronous communications, a lighter orchestrator with Network Services and Slicing capabilities, performance/fault/policy management features, a North Bound Interface (NBI), an enhanced GUI with a service composer, and OSM client leveraging the unified and improved REST API that the NBI exposes. The OSM NBI is a REST-full following ETSI SOL005 standard. By default it serves https (auto-signed certificate) on port 9999. Bearer authentication (with token) is used. Basic authentication or no authentication is also possible changing 'nbi.cfg' file. For developing purposes, it admits web browser navigation using a basic http format. In the following table, we give some example OSM[1] requests that we utilized in the 5G-MEDIA platform.

Operation	Method	Relative URL	Description

Get OSM Access Token	POST	/osm/admin/v1/tokens	Retrieve OSM Token.
			The body of the POST request must contain the following JSON object
			{
			"username": "admin",
			"password": "admin",
			"project_id": "admin"
			}

Get NS descriptors onboarded	GET	/osm/nsd/v1/ns_descriptors	Get NS descriptors onboarded in the OSM.
Get NSD with a specific ID onboarded	GET	/osm/nsd/v1/ns_descriptors/{nsdInfoId}	Get a specific NS descriptor onboarded in the OSM with an ID. **Path Parameters:**

[1] https://osm.etsi.org/wikipub/index.php/NBI_API_Description

			• nsdInfoId (string):* The ID of the network service descriptor

Create a new NSD resource	POST	/nsd/v1/ns_descriptors	Create a new NS descriptor resource. The pody of the POST request must contain a JSON object. Example Value:
			{
			"additionalProp1": {}
			}
			It returns a JSON that includes NSD ID
			{
			"id": "string"
			}

Upload the content of the NSD	PUT	/nsd/v1/ns_descriptors/{nsdInfoId}/nsd_content	Upload the content to the NS descriptor resource **Path Parameters:** • nsdInfoId (string):* The ID of the network service descriptor The body of the PUT request must contain the NSD content in zip file. Content-type: application/zip
Get NSD content with a specific ID onboarded	GET	/nsd/v1/ns_descriptors/{nsdInfoId}/nsd_content	Get NS descriptor content onboarded with a specific ID **Path Parameters:** • nsdInfoId (string):* The ID of the network service descriptor
Delete a specific NSD resource	DELETE	/nsd/v1/ns_descriptors/{nsdInfoId}	Delete NS descriptor resource a specific ID **Path Parameters:** • nsdInfoId (string):* The ID of the network service descriptor
Modify the data of an individual NSD resource	PATCH	/nsd/v1/ns_descriptors/{nsdInfoId}	Modify the data of an individual NS descriptor resource **Path Parameters:**

			• nsdInfoId (string):* The ID of the network service descriptor

Get VNF packages onboarded	GET	/vnfpkgm/v1/vnf_packages	Retrieve all VNF packages onboarded
			
			Returns array of all VNF Packages
			[
			{
			"_id": "string",
			"id": "string",
			"name": "string",
			"description": "string"
			}
			]

Get a specific VNF package with a specific ID onboarded	GET	/vnfpkgm/v1/vnf_packages/{vnfPkgId}	Get a VNF package onboarded with a specific ID
			Path Parameters:
			• vnfPkgId (string):* The ID of the VNF Package
			Returns a JSON Object
			{
			"_id": "string",
			"id": "string",
			"name": "string",
			"description": "string"
			}

Create a new VNF package resource	POST	/vnfpkgm/v1/vnf_packages	Create a new VNF package resource
			The body of the POST request must contain a
			JSON object. Example Value:
			{
			"additionalProp1": {}
			}
			
			It returns the ID of the VNF package resource. Example value:
			{

			"id": "string" }
Upload a VNF package by providing the content of the VNF package	PUT	/vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content	Upload a VNF package by providing the content of the VNF package **Path Parameters:** ● vnfPkgId (string):* The ID of the VNF Package The body of the PUT request must contain the NSD content in zip file. Content-type: application/zip
Get the content of a specific VNF package with an ID onboarded	GET	/osm/vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content	Fetch an onboarded VNF package with an ID **Path Parameters:** ● vnfPkgId (string):* The ID of the VNF Package

Table 8 Open Source MANO NBI Examples

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

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.

The OpenID Connect compliant Identity Server used in 5G-MEDIA is Keycloak[KEYCLOAK], which provides specific API to obtain “access” tokens to be added to the HTTP calls to remote resources. The OIDC integration in 5G-MEDIA comprehends two profiles (“Authorization Code Flow” and “Resource Owner Password Credentials”), one for web UI and the other for API interaction; the SDK uses only the latter to get access to the Catalogue NBI with a very easy workflow: first retrieves the “access” token as described in the deliverable D4.2 section 6, then add it to the HTTP calls as a “bearer” token, with an additional HTTP Header with the format:

Authorization: Bearer <TOKEN>

The “access” token has a short expiration date (by default, 300 seconds), so a “refresh” token with longer expiration date (by default, 1800 seconds) is provided at the login and can be used to produce new “access” token when necessary. When the “refresh” token expires, a new login is required. More details are in the deliverable D4.2 section 6.

The interactions from the SDK to the Catalogue NBI does not require any additional step for the authentication and authorization. The Catalogue can reuse the token to access OIDC resources or check with the AAA if a non OIDC resource is available to the logged in user and get a specific token/credentials to access it, in a transparent way from the SDK perspective.

The Figure 2 shows the high level workflow described above, with the login call to Keycloak that comprehends

- the resource identifier (“client_id”)
- the credentials (“username” and “password”)
- the OIDC profile used (“grant_type”)

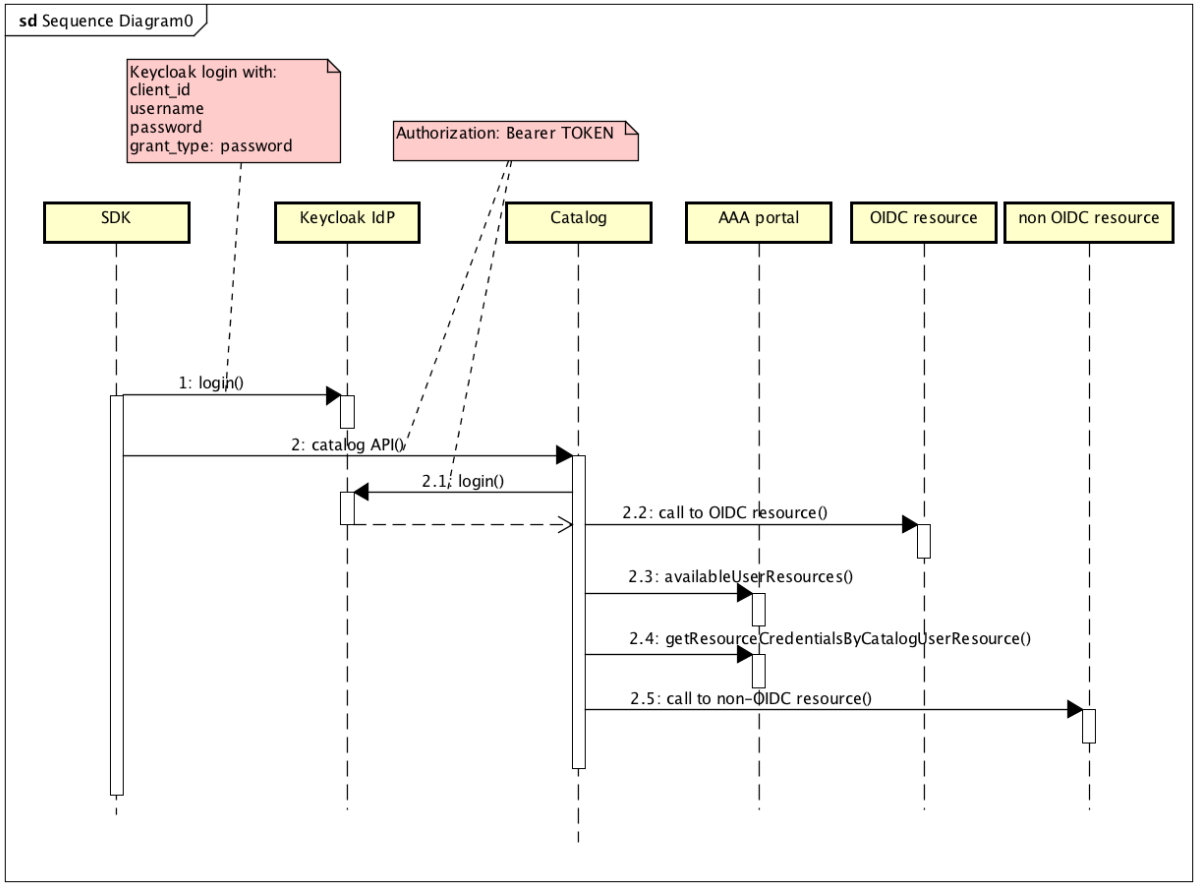

Figure 2 : SDK login on the Catalogue

Apart from the user credentials, the only other parameter to handle is the Keycloak “realm” name; a realm is a way Keycloak uses to group together a set of registered users, resources and authorizations. Such parameter needs to be specified as a URL parameter, as it is shown in Section 5.4.

5. SVP Rest APIs Tutorials and Code Samples

5.1. Serverless Framework

As mentioned in Section 4.1, the SDK uses a version of Apache Openwhisk with a small footprint that is referred to as Lean Openwhisk, however the semantic flow of the Openwhisk that is used in SVP is identical. Due to the fact that both exposes same WEB APIs, code samples for the SVP is the same as given for the SDK in Section 3.1.

5.2. Public Catalogue

Code samples for the public catalogue are the same as the ones given for the private catalogue. Therefore, the code samples for the private catalogue in the SDK which have been provided in Section 3.4 can also be utilized in the public catalogue in the SVP.

5.3. Open Source Management and Orchestration

- Code sample to get information of ns descriptors

```
curl -X GET 'http://IP_ADDRESS:PORT/nsd/v1/ns_descriptors
```

```
[
 {
  "_id": "string",
  "id": "string",
  "name": "string",
  "description": "string"
 }
]
```

- Code sample to create a new NS descriptor resource

```
curl -X POST -H  "Content-type: application/json" -d '{
  "additionalProp1": {},
  "folderId": 0,
  "overwrite": false
} 'http://IP_ADDRESS:PORT/nsd/v1/ns_descriptors
```

```
{
  "id": "string"
}
```


- Code sample to upload the content of a NSD

```
curl -X PUT "https://IP_ADDRESS:PORT/nsd/v1/ns_descriptors/123/nsd_content" -H "accept: application/json" -H "Content-Type: application/zip" -d "\"string\""
```

Accepted

- Code sample to delete a specific NSD descriptor resource

```
curl -X DELETE "https://IP_ADDRESS:PORT/nsd/v1/ns_descriptors/123" -H "accept: application/json"
```

No content

- Code sample to modify a specific NSD descriptor resource

```
curl -X PATCH "https://IP_ADDRESS:PORT/nsd/v1/ns_descriptors/1234" -H "accept: application/json" -H "Content-Type: application/json" -d "{\"id\":\"string\",\"name\":\"string\",\"description\":\"string\"}"
```

Accepted

- Code sample to get information of multiple VNF packages

```
curl -X GET "https://IP_ADDRESS:PORT/vnfpkgm/v1/vnf_packages" -H "accept: application/json"
```


```
[
  {
    "_id": "string",
```

```
  "id": "string",
  "name": "string",
  "description": "string"
 }
]
```

- Code sample to create a new VNF descriptor resource

```
curl -X POST -H  "Content-type: application/json" -d '{
 "additionalProp1": {},
 "folderId": 0,
 "overwrite": false
} “https://IP_ADDRESS:PORT/vnfpkgm/v1/vnf_packages”
```

```
{
 "id": "string"
}
```

- Code sample to upload the content of a VNF Package

```
curl -X PUT
"https://IP_ADDRESS:PORT/vnfpkgm/vnfpkgm/v1/vnf_packages/123/package_content" -H
"accept: application/json" -H "Content-Type: application/zip" -d "\"string\""
```

```
Accepted
```

- Code sample to delete a specific VNF Package

```
curl -X DELETE "https://IP_ADDRESS:PORT/vnfpkgm/v1/vnf_packages/123" -H "accept:
application/json"
```

```
No content
```

- Code sample to modify a specific VNF package resource

curl -X PATCH "https://IP_ADDRESS:PORT/vnfpkgm/v1/vnf_packages/123" -H "accept: application/json" -H "Content-Type: application/json" -d "{\"id\":\"string\",\"name\":\"string\",\"description\":\"string\"}"
Accepted

5.4. Authentication

The Figure 2 shows an example of REST API on Keycloak to get an “access” token using the “Resource Owner Password Credentials” code flow with the user credentials sent in the HTTP Post body, the resource identifier and secret (“client_id” and “client_secret”) and the realm parameter (valued “osm” in the example) sent in the URL.

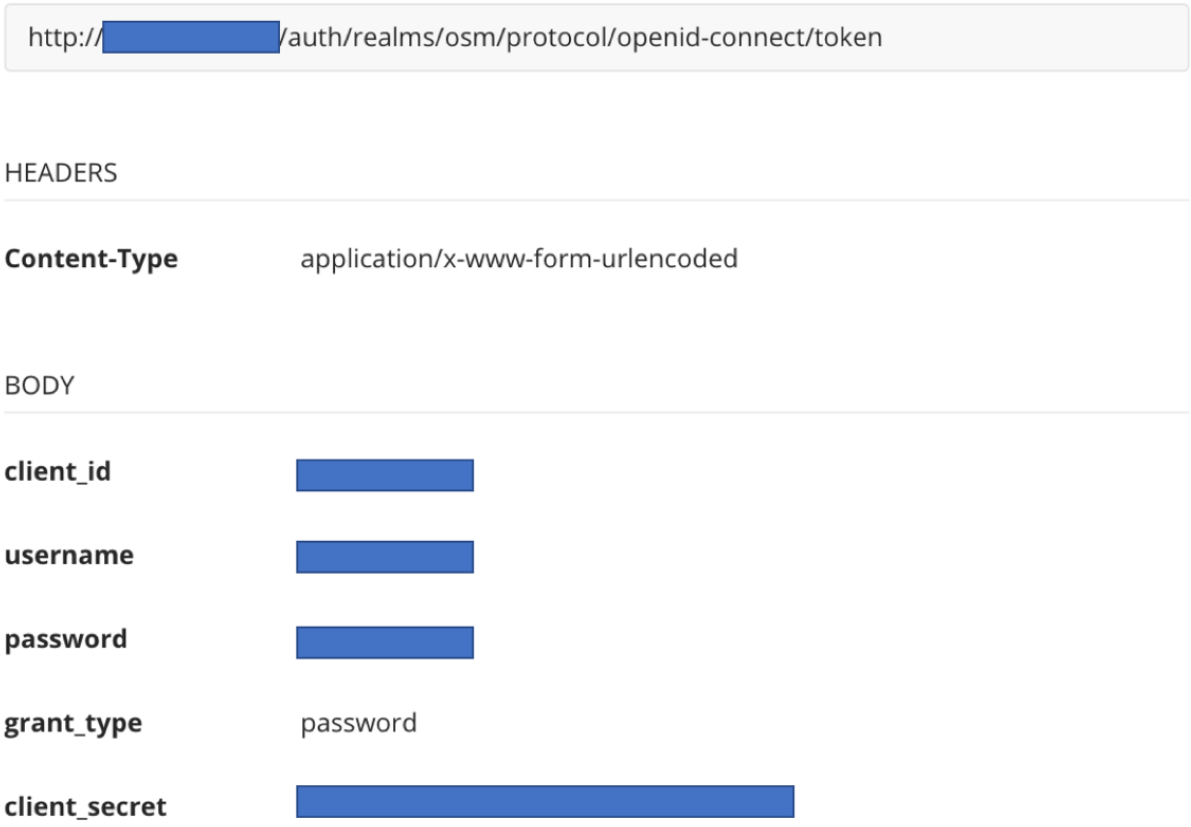

Figure 3 : Keycloak REST API to get an "access" and “refresh” token pair

The shows an example of REST API on Keycloak to get a new pair of “access” and “refresh” tokens using a valid “refresh” token and the resource identifier and secret.

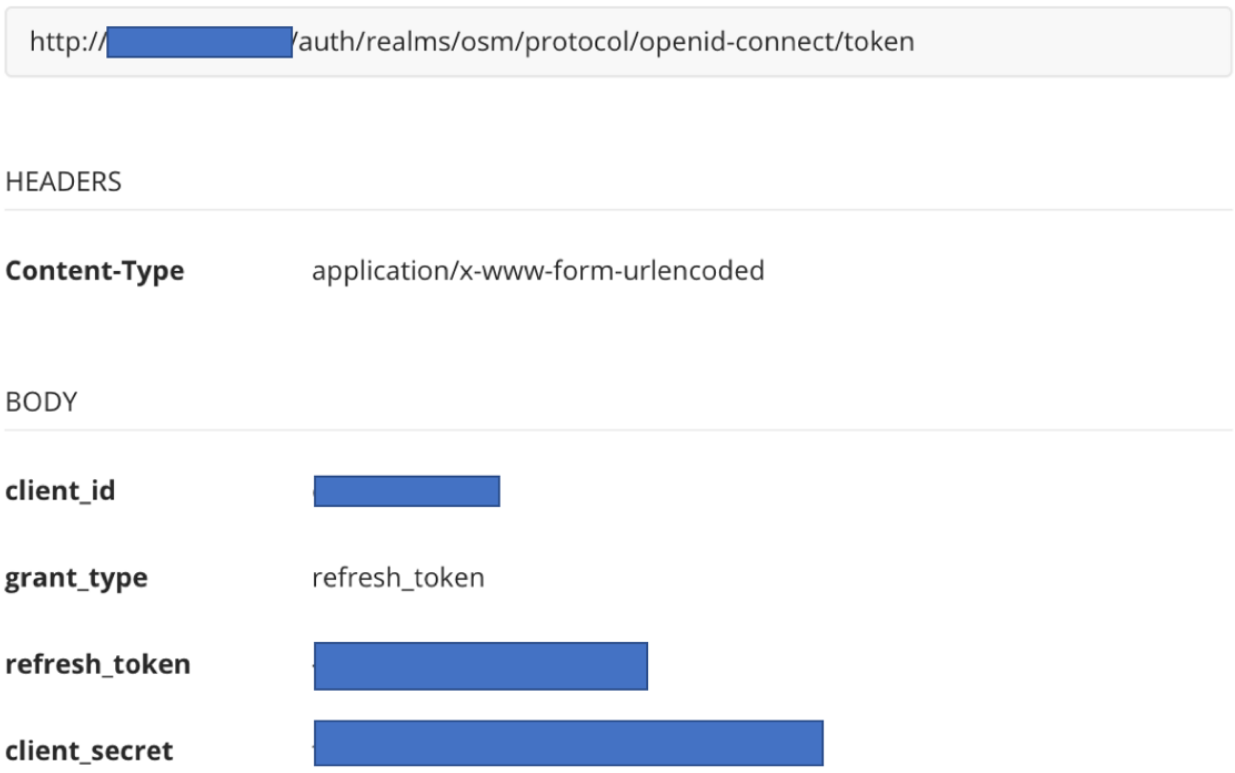

Figure 4 : Keycloak REST API to get new tokens using a "refresh" token

Both calls return the same response containing “access” and “refresh” tokens as shown in the Figure 4.


```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJhb0JIWEIwUXZP
    "expires_in": 300,
    "refresh_expires_in": 1800,
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICIwMzcwNWM3OC0
    "token_type": "bearer",
    "not-before-policy": 1557783005,
    "session_state": "2c3920ac-f762-4cc2-9702-9c3a2517ba9a",
    "scope": "email profile"
}
```


Figure 5 : Keycloak "access" and "refresh" token pair

6. Conclusions

In this deliverable, we have provided the REST APIs of 5G-MEDIA SDK and SVP platform including tools such as catalogue, OSM, monitoring, vim-emulator, serverless framework and cognitive network optimizer. Furthermore, this deliverable has also presented some tutorials with giving code samples using the REST APIs for both SDK and SVP tools.

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#/

[OWAPI] OpenWhisk REST API
http://petstore.swagger.io/?url=https://raw.githubusercontent.com/openwhisk/openwhisk/master/core/controller/src/main/resources/apiv1swagger.json#/

[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/