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

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

<table>
<thead>
<tr>
<th>Work Package:</th>
<th>WP5 – 5G-MEDIA APIs and SDK Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead partner:</td>
<td>NET</td>
</tr>
<tr>
<td>Author(s):</td>
<td>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]</td>
</tr>
<tr>
<td>Delivery date (DoA):</td>
<td>November 30th, 2019</td>
</tr>
<tr>
<td>Actual delivery date</td>
<td>November 29th, 2019</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>Public</td>
</tr>
<tr>
<td>Version number:</td>
<td>1.0</td>
</tr>
<tr>
<td>Status:</td>
<td>Final</td>
</tr>
</tbody>
</table>

| 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                         |
## Revision History

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

## Quality Control

<table>
<thead>
<tr>
<th>Role</th>
<th>Date</th>
<th>Who</th>
<th>Approved/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Reviewer</td>
<td>November 18th, 2019</td>
<td>ENG</td>
<td></td>
</tr>
<tr>
<td>Internal Reviewer</td>
<td>November 18th, 2019</td>
<td>UPM</td>
<td></td>
</tr>
<tr>
<td>2nd Internal Reviewer</td>
<td>November 26th, 2019</td>
<td>ENG</td>
<td></td>
</tr>
</tbody>
</table>
Disclaimer

This document may contain material that is copyright of certain 5G-MEDIA project beneficiaries, and may not be reproduced or copied without permission. The commercial use of any information contained in this document may require a license from the proprietor of that information. The 5G-MEDIA project is part of the European Community's Horizon 2020 Program for research and development and is as such funded by the European Commission. All information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability with respect to this document, which is merely representing the authors’ view.

The 5G-MEDIA Consortium is the following:

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

**Executive summary**  
6  
1. **Introduction**  
7  
2. **SDK WEB APIs Definition**  
8  
2.1. **Serverless Framework**  
8  
2.2. **Monitoring Tools**  
20  
2.3. **Vim-Emulator**  
23  
2.4. **Private Catalogue**  
27  
2.5. **Cognitive Network Optimizer**  
34  
3. **SDK WEB APIs Tutorials and Code Samples**  
36  
3.1. **Serverless Framework**  
36  
3.2. **Monitoring Tools**  
39  
3.3. **Vim Emulator**  
41  
3.4. **Private Catalogue**  
41  
3.5. **Cognitive Network Optimizer**  
43  
4. **SVP WEB APIs Definition**  
43  
4.1. **Serverless Framework**  
43  
4.2. **Public Catalogue**  
44  
4.3. **Open Source MANO**  
44  
4.4. **Authentication**  
48  
5. **SVP WEB APIs Tutorials and Code Samples**  
49  
5.1. **Serverless Framework**  
49  
5.2. **Public Catalogue**  
49  
5.3. **Open Source MANO**  
49  
5.4. **Authentication**  
51  
6. **Conclusions**  
51  
7. **References**  
52
## Definitions and acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>FaaS</td>
<td>Function as a Service</td>
</tr>
<tr>
<td>NBI</td>
<td>Northbound interface</td>
</tr>
<tr>
<td>NS</td>
<td>Network Service</td>
</tr>
<tr>
<td>NSD</td>
<td>NS Descriptor</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>REST</td>
<td>Representational state transfer</td>
</tr>
<tr>
<td>SDK</td>
<td>Service Development Kit</td>
</tr>
<tr>
<td>SVP</td>
<td>Service Virtualisation Platform</td>
</tr>
<tr>
<td>OSM</td>
<td>Open Source Management and Orchestration</td>
</tr>
<tr>
<td>VIM</td>
<td>Virtualized Infrastructure Manager</td>
</tr>
<tr>
<td>VNF</td>
<td>Virtual Network Function</td>
</tr>
<tr>
<td>VNFD</td>
<td>VNF Descriptor</td>
</tr>
</tbody>
</table>
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 5G-MEDIA 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, monitoring, vim emulator, private catalogue, validator and CNO module. 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.

```plaintext
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
}

<table>
<thead>
<tr>
<th>Throws</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
</tr>
<tr>
<td>Fail: Invalid namespace</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>The entity namespace</td>
</tr>
<tr>
<td>name:</td>
<td>Name of action to fetch</td>
</tr>
<tr>
<td>code:</td>
<td>Include action code</td>
</tr>
</tbody>
</table>

### Returns

**OW Action Information**

JSON Object that includes a specific action’s information such as its name, namespace, version, image, code, limits etc.

Example:

```json
{
  "namespace": "string",
  "name": "string",
  "version": "string",
  "publish": true,
  "exec": {
    "kind": "blackbox",
    "code": "string",
    "image": "string",
    "main": "string",
    "binary": true,
    "components": ["string"
                    ],
    "annotations": ["string"
                     ],
    "parameters": ["string"
                    ],
    "limits": {
      "timeout": 0,
      "memory": 0,
      "logs": 0,
      "concurrency": 0
    },
    "updated": 0
  }
}```
<table>
<thead>
<tr>
<th>Throws</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
</tr>
<tr>
<td>Fail: Invalid namespace or action name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>createAction (namespace: string, name: string, action: object)</th>
<th>Create an action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>namespace</td>
<td>The entity namespace</td>
</tr>
<tr>
<td>name:</td>
<td>Name of action</td>
</tr>
<tr>
<td>action:</td>
<td>Action Entities</td>
</tr>
<tr>
<td>Example Value:</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;namespace&quot;: &quot;string&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;name&quot;: &quot;string&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;version&quot;: &quot;string&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;publish&quot;: true,</td>
<td></td>
</tr>
<tr>
<td>&quot;exec&quot;: {</td>
<td></td>
</tr>
<tr>
<td>&quot;kind&quot;: &quot;blackbox&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;code&quot;: &quot;string&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;image&quot;: &quot;string&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;main&quot;: &quot;string&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;binary&quot;: true,</td>
<td></td>
</tr>
<tr>
<td>&quot;components&quot;: [</td>
<td></td>
</tr>
<tr>
<td>&quot;string&quot;</td>
<td></td>
</tr>
<tr>
<td>]</td>
<td></td>
</tr>
<tr>
<td>},</td>
<td></td>
</tr>
<tr>
<td>&quot;annotations&quot;: [</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;key&quot;: &quot;string&quot;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>],</td>
<td></td>
</tr>
<tr>
<td>&quot;parameters&quot;: [</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;key&quot;: &quot;string&quot;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>],</td>
<td></td>
</tr>
<tr>
<td>&quot;limits&quot;: {</td>
<td></td>
</tr>
<tr>
<td>&quot;timeout&quot;: 0,</td>
<td></td>
</tr>
<tr>
<td>&quot;memory&quot;: 0,</td>
<td></td>
</tr>
<tr>
<td>&quot;logs&quot;: 0,</td>
<td></td>
</tr>
<tr>
<td>&quot;concurrency&quot;: 0</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>
Returns

<table>
<thead>
<tr>
<th>OW Action Information</th>
<th>JSON Object that includes the recently-created action’s information such as its name, namespace, version, image, code, limits etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;namespace&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;version&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;publish&quot;: true,</td>
</tr>
<tr>
<td></td>
<td>&quot;exec&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;kind&quot;: &quot;blackbox&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;code&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;image&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;main&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;binary&quot;: true,</td>
</tr>
<tr>
<td></td>
<td>&quot;components&quot;: [</td>
</tr>
<tr>
<td></td>
<td>&quot;string&quot;</td>
</tr>
<tr>
<td></td>
<td>]</td>
</tr>
<tr>
<td></td>
<td>},</td>
</tr>
<tr>
<td></td>
<td>&quot;annotations&quot;: [</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;key&quot;: &quot;string&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>],</td>
</tr>
<tr>
<td></td>
<td>&quot;parameters&quot;: [</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;key&quot;: &quot;string&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>],</td>
</tr>
<tr>
<td></td>
<td>&quot;limits&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;timeout&quot;: 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;memory&quot;: 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;logs&quot;: 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;concurrency&quot;: 0</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

Throws

Success

Fail: Invalid namespace or action name

<table>
<thead>
<tr>
<th>deleteAction (namespace: string, name: string)</th>
<th>Delete an action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td></td>
</tr>
</tbody>
</table>
### getRules

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>The entity namespace</td>
</tr>
<tr>
<td>limit</td>
<td>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.</td>
</tr>
<tr>
<td>skip</td>
<td>Number of entities to skip in the result.</td>
</tr>
</tbody>
</table>

**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:

```json
[
  {
    "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:

```json
{
  "namespace": "string",
  "name": "string",
  "version": "string",
  "publish": true,
  "annotations": [
    {
      "key": "string"
    }
  ],
  "status": "active",
  "trigger": {
    "path": "string",
    "name": "string"
  },
  "action": {
    "path": "string",
    "name": "string"
  }
}
```
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Create a rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>The entity namespace</td>
</tr>
<tr>
<td>name:</td>
<td>Name of the rule</td>
</tr>
<tr>
<td>rule:</td>
<td>Rule Body, Example value:</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;version&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;publish&quot;: true,</td>
</tr>
<tr>
<td></td>
<td>&quot;annotations&quot;: [</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;key&quot;: &quot;string&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>],</td>
</tr>
<tr>
<td></td>
<td>&quot;status&quot;: &quot;active&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;trigger&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;action&quot;: &quot;string&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

Returns

| JSON Object that includes the recently-created rule’s information such as its name, namespace, version, annotations, trigger, action etc. |
| Example: |
| { |
|   "name": "string", |
|   "version": "string", |
|   "publish": true, |
|   "annotations": [ |
|     { |
|       "key": "string" |
|     } |
|   ], |
|   "status": "active", |
|   "trigger": "string", |
|   "action": "string" |
| } |
```json
{
  "status": "active",
  "trigger": "string",
  "action": "string"
}
```

**Throws**

**Success**

Fail: Invalid namespace or rule

### deleteRule (namespace: `string`, name: `string`)

Delete a rule

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>The entity namespace</td>
</tr>
<tr>
<td>name:</td>
<td>Name of rule</td>
</tr>
</tbody>
</table>

**Throws**

**Success**

Fail: Invalid namespace or rule name

### getTriggers (namespace: `string`, limit: `integer`, skip: `integer`)

Get all OW triggers

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>The entity namespace</td>
</tr>
<tr>
<td>limit</td>
<td>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.</td>
</tr>
<tr>
<td>skip</td>
<td>Number of entities to skip in the result.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>OW Triggers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An array with JSON objects that includes all triggers information such as their</td>
</tr>
</tbody>
</table>
names, namespaces, versions, annotations, parameters, limits etc. Example Value:

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

**Throws**

**Success**

**Fail: Invalid namespace**

<table>
<thead>
<tr>
<th><strong>getSingleTrigger</strong> (namespace: <code>string</code>, name: <code>string</code>)</th>
<th>Get a specific trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameters</strong></td>
<td></td>
</tr>
<tr>
<td>namespace</td>
<td>The entity namespace</td>
</tr>
<tr>
<td>name:</td>
<td>Name of trigger to fetch</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td></td>
</tr>
<tr>
<td>OW Trigger Information</td>
<td>JSON Object that includes the information of a specific trigger. Example Value:</td>
</tr>
</tbody>
</table>
```json
{
    "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:

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

```json
{
  "namespace": "string",
  "name": "string",
  "version": "string",
  "publish": true,
  "annotations": [
    {
      "key": "string"
    }
  ],
  "parameters": [
    {
      "key": "string"
    }
  ],
  "limits": {}
}
```

**Throws**

**Success**

Fail: Invalid namespace or trigger

---

```markdown
### deleteTrigger (namespace: string, name: string)
Delete a trigger
```

**Parameters**

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>The entity namespace</td>
</tr>
</tbody>
</table>
name: Name of trigger

<table>
<thead>
<tr>
<th>Throws</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td></td>
</tr>
<tr>
<td>Fail: Invalid namespace or trigger name</td>
<td></td>
</tr>
</tbody>
</table>

### 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.

<table>
<thead>
<tr>
<th>createDashboard (dashboard: object)</th>
<th>Create/ Update a dashboard in Grafana</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>dashboard</td>
<td>The complete dashboard model (JSON) Example Model:</td>
</tr>
</tbody>
</table>
|                                     | {
|                                     |   "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**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>getMonitoringData</strong>(query: <em>string</em>)</td>
<td>Get data by query via Prometheus</td>
</tr>
</tbody>
</table>

**Parameters**

- **query**
  - Prometheus expression query string

**Returns**

- **Monitoring Data**
  - Monitoring Data in JSON object
  
  ```json
  {
  "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]
  ]
  }
  }
  ```
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",
                "events": [
                    "docker/86c1d6c61a49f7bb5be2296bd0648e0e8d0aed6c8e4bda6ae351f5120b558052"
                ],
            },
            "image": "redis",
            "labels": {},
            "container_id": "86c1d6c61a49f7bb5be2296bd0648e0e8d0aed6c8e4bda6ae351f5120b558052",
            "mounts": [],
            "name": "/docker/86c1d6c61a49f7bb5be2296bd0648e0e8d0aed6c8e4bda6ae351f5120b558052",
            "ports": [],
            "start_time": "2019-08-29T13:27:22.431584149Z"
        }
    },
    "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",
                "events": [
                    "docker/86c1d6c61a49f7bb5be2296bd0648e0e8d0aed6c8e4bda6ae351f5120b558052"
                ],
            },
            "image": "redis",
            "labels": {},
            "container_id": "86c1d6c61a49f7bb5be2296bd0648e0e8d0aed6c8e4bda6ae351f5120b558052",
            "mounts": [],
            "name": "/docker/86c1d6c61a49f7bb5be2296bd0648e0e8d0aed6c8e4bda6ae351f5120b558052",
            "ports": [],
            "start_time": "2019-08-29T13:27:22.431584149Z"
        }
    }
}
"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": [...]
}

<table>
<thead>
<tr>
<th>Throws</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td></td>
</tr>
<tr>
<td>Fail: Invalid container name</td>
<td></td>
</tr>
</tbody>
</table>

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 [DS.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.

<table>
<thead>
<tr>
<th>listDatacenters()</th>
<th>List all data centers</th>
</tr>
</thead>
</table>


Returns

<table>
<thead>
<tr>
<th>List of all data centers</th>
<th>The complete list of data centers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example Value:</td>
</tr>
<tr>
<td></td>
<td>[</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;internalname&quot;: &quot;dc2&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;label&quot;: &quot;datacenter2&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;metadata&quot;: {},</td>
</tr>
<tr>
<td></td>
<td>&quot;n_running_containers&quot;: 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;switch&quot;: &quot;dc2.s1&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;internalname&quot;: &quot;dc3&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;label&quot;: &quot;long_data_center_name3&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;metadata&quot;: {},</td>
</tr>
<tr>
<td></td>
<td>&quot;n_running_containers&quot;: 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;switch&quot;: &quot;dc3.s1&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>VNF Instance Information</th>
<th>Instance information in JSON Object</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example Value:</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;vnf1&quot;:</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;cpu_period&quot;: null,</td>
</tr>
<tr>
<td></td>
<td>&quot;cpu_quota&quot;: -1,</td>
</tr>
<tr>
<td></td>
<td>&quot;cpu_shares&quot;: null,</td>
</tr>
<tr>
<td></td>
<td>&quot;cpuset&quot;: null,</td>
</tr>
<tr>
<td></td>
<td>&quot;datacenter&quot;: &quot;cvim1&quot;,</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>
"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"
}
}

<table>
<thead>
<tr>
<th>Throws</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
</tr>
<tr>
<td>Fail: Invalid vnf name</td>
</tr>
</tbody>
</table>

del**eteVNF**(name: *string*)  Delete a specific VNF instance

| Parameters               |
|--------------------------|-----------------|
|                          |                 |
### createMonitoring

**Parameters**

- **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

<table>
<thead>
<tr>
<th>Throws</th>
<th>Success</th>
<th>Fail: Invalid vnf name</th>
</tr>
</thead>
</table>

### deleteMonitoring

delete the action that monitors the counters of a VNF interface.
### Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>Name of the VNFs to monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>interface</td>
<td>Name of the VNF interface to be monitored</td>
</tr>
</tbody>
</table>
| 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(CreateNsdlInfoRequest: JSON Object)

**Create NSD Resource**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>JSON Object</th>
</tr>
</thead>
</table>
| CreateNsdlInfoRequest | Example Value:  
| | {}  
| | "nsdId": "85ee1962-f65b-4965-a73f-96b5e1d9d068",  
| | "nsdName": "faas_pingpong_nsd",  
| |} |
```json
"nsdVersion": "1.1",
"nsdDesigner": "IBM",
"nsdInvariantId": "85ee1962-f65b-4965-a73f-96b5e1d9d068",
"vnfPkgIds": [
  "44aa96b6-2c55-4d20-a7c0-faca82d492d2",
  "c19e63d6-46ec-48d7-9c18-3f10e58eee3f"
],
"pnfdInfolds": [],
"nestedNsdInfolds": [],
"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

<table>
<thead>
<tr>
<th>NSD ID</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throws</td>
<td></td>
</tr>
<tr>
<td>Success</td>
<td></td>
</tr>
<tr>
<td>Fail: Invalid NSD Request</td>
<td></td>
</tr>
</tbody>
</table>

#### getNSDList()

Query information about multiple NS descriptor resources.

### Returns

<table>
<thead>
<tr>
<th>NSDs</th>
<th>Array of information in JSON Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example Value:</td>
<td>[ ]</td>
</tr>
</tbody>
</table>
```json
{
  "id": "082ead84-3689-4c7b-a231-56c79a31398c",
  "nsdId": "95a6c14e-a1ad-4893-8865-344240a8d83",
  "nsdName": "vtranscoder_2_7_2_nsd",
  "nsdVersion": "1.1",
  "nsdDesigner": "IBM",
  "nsdInvariantId": "95a6c14e-a1ad-4893-8865-344240a8d83",
  "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",
    "03c46515-5f5c3-47a3-9086-6b55bfac74233"
  ],
  "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"
}
```
### getNSD(id: string)

Query a specific NSD

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>label of the NSD Info ID</td>
</tr>
</tbody>
</table>

#### Returns

NSD information in JSON Object

Example Value:

```json
{
    "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,
    ...
}
```
### createVNFD(file: CSAR)

Create NSD Resource

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>Compressed package file in CSAR format</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>VNFD Package ID</th>
<th>String</th>
</tr>
</thead>
</table>

**Throws**

Success

Fail: Invalid VNFDRequest

### getVNFDList()

Query information about multiple VNF package resources.

**Returns**

<table>
<thead>
<tr>
<th>VNFDs</th>
<th>Array of Information in JSON Objects</th>
</tr>
</thead>
</table>

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-e5049855ea1",
      "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",
```
### getVNFD(id: string)

Query a specific VNFD

**Parameters**

- **id**
  
  label of the VNFD Info ID

**Returns**

**VNFD**

Information in JSON Objects

Example Value:

```json
{
  "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": {
```

**Throws**

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

```javascript
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/"
  }
}
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployMape(ip: String, pemfile: File)</td>
<td>Deploy Training Model to production environment</td>
</tr>
<tr>
<td>stopTraining()</td>
<td>Terminate training</td>
</tr>
<tr>
<td>startTensorboard()</td>
<td>Start Tensorboard</td>
</tr>
</tbody>
</table>

**stopTraining()**

- **Throws**
- **Success**
- **Fail: Invalid request**

**startTensorboard()**

- **Throws**
- **Success**
- **Fail: Invalid request**

**deployMape**

- **Throws**
- **Success**
- **Fail: Invalid request**

```
"parallelAgent": "2"
```
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip</td>
<td>String</td>
</tr>
<tr>
<td>pemfile</td>
<td>File for permission to access to the server</td>
</tr>
</tbody>
</table>

### 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: {...},
});
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

```javascript
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

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

// Get monitoring container information

```javascript
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.

```javascript
// 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.

```javascript
// 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

```javascript
osmAuthToken({
  "username": "admin",
  "password": "admin",
  "project_id": "admin"
})
```

Create a specific NSD resource in the OSM with its content.

**Returns**

<table>
<thead>
<tr>
<th>OSMAuthResponse</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example Value:</td>
</tr>
<tr>
<td></td>
<td><code>{ Value: ***** }</code></td>
</tr>
</tbody>
</table>

**Throws**

- Success
- Fail: Invalid NSD Content

### getNsds

Get all NS descriptors onboarded in the OSM

**Returns**

<table>
<thead>
<tr>
<th>NSD List</th>
<th>Array of JSON Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example Value:</td>
</tr>
<tr>
<td></td>
<td><code>[ </code></td>
</tr>
<tr>
<td></td>
<td><code>   </code></td>
</tr>
<tr>
<td></td>
<td><code>    </code></td>
</tr>
</tbody>
</table>
|          | `    _id": "string",
|          |   "id": "string",
|          |   "name": "string",
|          |   "description": "string"
|          | ` ]` |

**Throws**

- Success
- Fail: Invalid configuration
### `getVnfds()`

Get all VNF packages onboarded in the OSM

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNFD List</td>
</tr>
<tr>
<td>Array of JSON Object</td>
</tr>
<tr>
<td>Example Value:</td>
</tr>
</tbody>
</table>
| `[
  {
    "_id": "string",
    "id": "string",
    "name": "string",
    "description": "string"
  }
]` |

<table>
<thead>
<tr>
<th>Throws</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
</tr>
</tbody>
</table>

| Fail: Invalid configuration |

### `createNsd(Body: Object)`

Create a specific NSD resource in the OSM with its content.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body</td>
</tr>
<tr>
<td>JSON Object</td>
</tr>
<tr>
<td>Example Value:</td>
</tr>
<tr>
<td>`{</td>
</tr>
</tbody>
</table>
| "_id": "string",
| "id": "string",
| "name": "string",
| "description": "string" |
| } |

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSD Id</td>
</tr>
<tr>
<td>String</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throws</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
</tr>
</tbody>
</table>
### getNsdInfo (nsdId: string)

Get a specific NSD info using its unique ID.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsdId</td>
<td>Unique NSD ID</td>
</tr>
</tbody>
</table>

**Returns**

NSD Info: JSON Object

```json
{
    "_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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body</td>
<td>JSON Object</td>
</tr>
</tbody>
</table>

**Example Value**

```json
{
    "_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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getToken</td>
<td>Get an access token using resource owner password credentials</td>
</tr>
</tbody>
</table>

**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.

```javascript
// 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


[D4.1] 5G-MEDIA Deliverable “D4.1 5G-MEDIA Catalogue APIs and Network Apps”


[D5.1] 5G-MEDIA Deliverable “D5.1 5G-MEDIA Programming Tools”


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


[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%2FSOLO005.git%3Fb%3Dmain%2F%3Dso%60s-openapi.yaml%23%23HEAD#


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