**VNF Management Requirements for OpenO**

**Table of Content**

[1. Introduction 3](#_Toc489630750)

[2. GVNFM Scenario 3](#_Toc489630751)

[**2.1** **VNF Develop Steps** 3](#_Toc489630752)

[**2.2** **VNF Rest api** 4](#_Toc489630753)

[**2.2.1** **Set Initial Configuration** 4](#_Toc489630754)

[**2.2.2** **Response Code** 10](#_Toc489630755)

[3. SVNFM Scenario 10](#_Toc489630756)

[**3.1** **VNFM Driver Develop Steps** 10](#_Toc489630757)

[**3.2** **Create SVNFM Adaptor Mircoservice** 11](#_Toc489630758)

[**3.3** **Interfaces provided by SVNFM Driver** 11](#_Toc489630759)

[**3.3.1** **Instantiate VNF** 11](#_Toc489630760)

[**3.3.2** **Terminate VNF** 14](#_Toc489630761)

[**3.3.3** **Query VNF** 16](#_Toc489630762)

[**3.3.4** **Get operation status** 18](#_Toc489630763)

[**3.3.5** **Scale VNF** 19](#_Toc489630764)

# Introduction

This document is **NF Management Requirements for OPEN-O**, which describes the requirements for how VNFs interact and utilize OPEN-O.

OPEN-O is a platform that enables telecommunications and cable operators to effectively deliver end-to-end services across Network Functions Virtualization (NFV) Infrastructure, as well as Software Defined Network (SDN) and legacy network services.

OPEN-O provide two ways to Manage VNFs, one is managed directly by gvnfm that is already a component in OPEN-O, the other is managed by providing a svnfm to manage special VNFs. When providing a svnfm, a svnfm adaptor must be implemented the interfaces between nfvo and svnfm.

OPEN-O Model Design Framework provides the ability to design NFV resources including NF and NS. The vendor must provide VNF and NS packages that include a rich set of recipes, management and functional interfaces, policies, configuration parameters, and infrastructure requirements that can be utilized by the OPEN-O Model Design module to onboard and catalog these resources.

The current VNF Package Requirement is based on a subset of the Requirements contained in the ETSI Document: ETSI GS NFV-MAN 001 v1.1.1 and GS NFV IFA011 V0.3.0 (2015-10) - Network Functions Virtualization (NFV), Management and Orchestration, VNF Packaging Specification.

# GVNFM Scenario

* 1. **VNF Develop Steps**

Aid to help the VNF vendor to fasten the integration with the GVNFM, the OpenO provides the VNF SDK tools, and the documents. In this charter, the develop steps for VNF vendors will be introduced.

First, using the VNF SDK tools to design the VNF with TOSCA model and output the VNF TOSCA package. The VNF package can be validated, and tested.

Second, the VNF vendor should provide the VNF Rest API to integrate with the GVNFM if needed. The VNF Rest API is aligned to the ETSI IFA document.

Third, the TOSCA model supports the EPA feature.

Note:

1. The scripts to extend capacity to satisfy some special requirements. In the R2, the scripts is not implemented fully, and will be provided in the next release.
2. The monitoring and scale policy also be provide the next release.
   1. **VNF Rest api**

The VNF must provide a rest api to support initial configuration over HTTP(s).

* + 1. **Set Initial Configuration**

|  |  |
| --- | --- |
| If Definition | Description |
| URI | http(s)://[hostname][:port]/configuration |
| Operation | POST |
| Direction | VNFM->VNF |

* + - 1. **Request**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Qualifier | Cardinality | Content | Description |
| vnfInstanceId | M | 1 | Identifier | Identifier of the VNF instance which the VNF to set initial configuration. |
| vnfConfigurationData | O | 0..1 | VnfConfiguration | Configuration data for the VNF instance. |
| vnfcConfigurationData | O | 0..N | VnfcConfiguration | Configuration data for VNFC instances. |

**VnfConfiguration:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Qualifier | Cardinality | Content | Description |
| cp | O | 0..N | CpConfiguration | External CPs |
| vnfSpecificData | O | 0..1 | VnfConfigurableProperties | Configuration object containing values of  VNF configurable properties. |

**CpConfiguration:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Qualifier | Cardinality | Content | Description |
| cpId | M | 1 | Identifier | Uniquely identifies a CP instance within the  namespace of a specific VNF instance or  VNFC instance. |
| cpdId | M | 1 | Identifier | Uniquely identifies a type of CP instance within the namespace of a VNFD. |
| cpAddress | M | 1..N | CpAddress | Address and Port assigned to the CP. |

**CpAddress:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Qualifier | Cardinality | Content | Description |
| address | M | 0..N | NetworkAddress | The address assigned to the CP instance  (e.g. IP address, MAC address, etc.). It shall be provided for configuring a fixed address. |
| useDynamicAddress | M | 0..1 | ENUM | It determines whether an address shall be  assigned dynamically. It shall be provided if a dynamic address needs to be configured on the CP.  A cardinality of "0" indicates that no dynamic address needs to be configured on the CP.  Permitted values:   * TRUE * FALSE |
| port | M | 0..1 | Not specified | The port assigned to the CP instance (e.g. IP port number, Ethernet port number, etc.).  Reserved |

**VnfConfigurableProperties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Qualifier | Cardinality | Content | Description |
| autoScalable | O | 0..1 | ENUM | It permits to enable (TRUE) / disable (FALSE) the auto-scaling functionality.  A cardinality of "0" indicates that configuring this present VNF property is not supported.  Permitted values:   * TRUE * FALSE |
| autoHealable | O | 0..1 | ENUM | It permits to enable (TRUE) / disable (FALSE) the auto-healing functionality.  A cardinality of "0" indicates that configuring this present VNF property is not supported.  Permitted values:   * TRUE * FALSE |
|  |  |  |  |  |

**VnfcConfiguration:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Qualifier | Cardinality | Content | Description |
| vnfcId | M | 1 | Identifier | Uniquely identifies a VNFC instance within the namespace of a specific VNF instance. |
| cp | O | 0..N | CpConfiguration | Internal CPs. |
| vnfcSpecificData | O | 0..1 | KeyValuePair | Configuration object containing values of VNFC configurable properties |

{

"vnfInstanceId": "1",

"vnfConfigurationData": {

"cp": [

{

"cpId": "cp-1",

"cpdId": "cpd-a",

"cpAddress": [

{

"addresses": [

{

"addressType": "MAC",

"l2AddressData": "00:f3:43:20:a2:a3"

},

{

"addressType": "IP",

"l3AddressData": {

"iPAddressType": "IPv4",

"iPAddress": "192.168.104.2"

}

}

],

"useDynamicAddress": "FALSE"

}

]

}

],

"vnfSpecificData": {

"autoScalable": "FALSE",

"autoHealable": "FALSE"

}

},

"vnfcConfigurationData": {

"vnfcId": "vnfc-1",

"cp": [

{

"cpId": "cp-11",

"cpdId": "cpd-1a",

"cpAddress": [

{

"addresses": [

{

"addressType": "MAC",

"l2AddressData": "00:f3:43:21:a2:a3"

},

{

"addressType": "IP",

"l3AddressData": {

"iPAddressType": "IPv4",

"iPAddress": "192.168.105.2"

}

}

],

"useDynamicAddress": "FALSE"

}

]

}

],

"vnfcSpecificData": {}

}

}

* + - 1. **Response**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Qualifier | Cardinality | Content | Description |
| vnfConfigurationData | O | 0..1 | VnfConfiguration | Correspond to the vnfConfigurationData in the input information elements of the SetInitialConfiguration operation if it has. |
| vnfcConfigurationData | O | 0..N | VnfcConfiguration | Correspond to the vnfcConfigurationData in the input information elements of the SetInitialConfiguration operation if it has. |

{

"vnfConfigurationData": {

"cp": [

{

"cpId": "cp-1",

"cpdId": "cpd-a",

"cpAddress": [

{

"addresses": [

{

"addressType": "MAC",

"l2AddressData": "00:f3:43:20:a2:a3"

},

{

"addressType": "IP",

"l3AddressData": {

"iPAddressType": "IPv4",

"iPAddress": "192.168.104.2"

}

}

],

"useDynamicAddress": "FALSE"

}

]

}

],

"vnfSpecificData": {

"autoScalable": "FALSE",

"autoHealable": "FALSE",

…

}

},

"vnfcConfigurationData": {

"vnfcId": "vnfc-1",

"cp": [

{

"cpId": "cp-11",

"cpdId": "cpd-1a",

"cpAddress": [

{

"addresses": [

{

"addressType": "MAC",

"l2AddressData": "00:f3:43:21:a2:a3"

},

{

"addressType": "IP",

"l3AddressData": {

"iPAddressType": "IPv4",

"iPAddress": "192.168.105.2"

}

}

],

"useDynamicAddress": "FALSE"

}

]

}

],

"vnfcSpecificData": {…}

}

}

* + 1. **Response Code**

|  |  |  |
| --- | --- | --- |
| Code | Meaning | Description |
| 201 | Created | A VNF Instance identifier was created successfully. |
| 4xx/5xx | <name from RFC7231> | <description> |

# SVNFM Scenario

* 1. **VNFM Driver Develop Steps**

Aid to help the VNF vendor to fasten the integration with the NFVO via Special VNFM, the OpenO provides the documents. In this charter, the develop steps for VNF vendors will be introduced.

First, using the VNF SDK tools to design the VNF with TOSCA model and output the VNF TOSCA package. The VNF package can be validated, and tested.

Second, the VNF vendor should provide SVNFM Driver in the OpenO, which is a micro service and in duty of translation interface from NFVO to SVNFM. The interface of NFVO is aligned to the ETSI IFA interfaces and can be gotten in the charter 5.5. The interface of SVNFM is provided by the VNF vendor self.

* 1. **Create SVNFM Adaptor Mircoservice**

Some vnfs are managed by special vnfm, before add svnfm to openo, a svnfm adaptor must be added to openo to adapter the interface of nfvo and svnfm.

A svnfm adaptor is a micro service with unique name and an appointed port, when started up, it must be auto registered to MSB(Micro server bus),following describes an example rest of register to MSB:

POST /openoapi/microservices/v1/services

{

"serviceName": "catalog",

"version": "v1",

"url": "/openoapi/catalog/v1",

"protocol": "REST",

"visualRange": "1",

"nodes": [

{

"ip": "10.74.56.36",

"port": "8988",

"ttl": 0

}

]

}

A svnfm

* 1. **Interfaces provided by SVNFM Driver**

Interfaces use RESTful API and the format is as follows:

http(s)://[hostname][:port]/openoapi/{vnfmtype}/v1/{vnfm\_id}/[……]

**R1 vnfmtype:**

**zte-vnfm**

**fw-vnfm**

**juju**

* + 1. **Instantiate VNF**

|  |  |
| --- | --- |
| If Definition | Description |
| URI | http(s)://[hostname][:port]/openoapi/{vnfmtype}/v1/{vnfmid}/vnfs |
| Operation | POST |
| Direction | NSLCM->VNFMDriver |

* + - 1. **Request**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Qualifier | Cardinality | Content | Description |
| vnfInstanceName | M | 1 | String | Human-readable name of the VNF instance to be created. |
| vnfPackageId | M | 1 | String | VNF packageId |
| vnfDescriptorId | M | 1 | String | Information sufficient to identify the VNF Descriptor which defines the VNF to be created. |
| flavourId | M | 0..1 | String | Reserved |
| vnfInstanceDescription | M | 0..1 | String | Human-readable description of the VNF instance to be created. |
| extVirtualLink | M | 0..N | ExtVirtualLinkData | References to external virtual links to connect the VNF to. |
| additionalParam | M | 0..N | KeyValuePair | Additional parameters passed by the NFVO as input to the instantiation process, specific to the VNF being instantiated. |

**ExtVirtualLinkData:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Qualifier | Cardinality | Content | Description |
| vlInstanceId | M | 0..1 | String | Identifier of the VL instance. |
| vim | CM | 0..1 | VimInfo | Information about the VIM that manages this resource.  This attribute shall be supported and present if VNF-related resource management in direct mode is applicable. |
| networkId | M | 1 | String | The network UUID of VIM |
| cpdId | M | 0..1 | String | Identifier of the external CPD in VNFD |

**VimInfo:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **Qualifier** | **Cardinality** | **Content** | **Description** |
| vimInfoId | M | 1 | Identifier | The identifier of this VimInfo instance, for the purpose of referencing it from other information elements. |
| vimId | M | 1 | Identifier | The identifier of the VIM. |
| interfaceInfo | M | 0..N | KeyValuePair | Information about the interface to the VIM, including VIM provider type, API version, and protocol type. |
| accessInfo | M | 0..N | KeyValuePair | Authentication credentials for accessing the VIM. Examples may include those to support different authentication schemes, e.g., OAuth, Token, etc. |
| interfaceEndpoint | M | 1 | String | Information about the interface endpoint. An example is a URL. |

**interfaceInfo:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Qualifier | Cardinality | Content | Description |
| vimType | M | 1 | String | vim |
| apiVersion | M | 1 | String |  |
| protocolType | M | 1 | String | http  https |

**accessInfo:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Qualifier | Cardinality | Content | Description |
| tenant | M | 1 | String | Tenant Name of tenant |
| username | M | 1 | String | Username for login |
| password | M | 1 | String | Password of login user |

{

“vnfInstanceName”:”vFW”,

“vnfPackageId”:”1”,

“vnfDescriptorId”:”1”,

“vnfInstanceDescription”:”vFW\_1”,

“extVirtualLinkLink”:[

{

”vlInstanceId”:”1”,

“resourceId”:”1246”,

” cpdId”:”11111”,

”vim”:

{

“vimInfoId”:”1”,

“vimid”:”1”,

“interfaceInfo”:{

"vimType":”vim”,

"apiVersion":”v2”,

"protocolType":”http”

}

“accessInfo”:{

"tenant":”tenant\_vCPE”,

"username":”vCPE”,

"password":”vCPE\_321”

}

“interfaceEndpoint”:”http://10.43.21.105:80/”

}

}

]

“additionalParam”:{

……

}

}

* + - 1. **Response**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Qualifier | Cardinality | Content | Description |
| jobId | M | 1 | Identifier | Identifier of the VNF lifecycle operation occurrence.  [lifecycleOperationOccurrenceId**]** |
| vnfInstanceId | M | 1 | Identifier | Identifier of the VNF instance. |

{

“jobId”:”1”,

“vnfInstanceId”:”1”

}

* + 1. **Terminate VNF**

|  |  |
| --- | --- |
| IF Definition | Description |
| URI | http(s)://[hostname][:port]/openoapi/{vnfmtype}/v1/{vnfmid}/vnfs/{vnfInstanceId}/terminate |
| Operation | POST |
| Direction | NSLCM->VNFMDriver |

* + - 1. **Request**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Qualifier | Cardinality | Content | Description |
| terminationType | M | 1 | Enum | Signals whether forceful or graceful termination is requested.  In case of forceful termination, the VNF is shut down immediately, and resources are released. Note that if the VNF is still in service, this may adversely impact network service, and therefore, operator policies apply to determine if forceful termination is allowed in the particular situation.  In case of graceful termination, the VNFM first arranges to take the VNF out of service (by means out of scope of the present specification, e.g. involving interaction with EM, if required). Once this was successful, or after a timeout, the VNFM shuts down the VNF and releases the resources. |
| gracefulTerminationTimeout | M | 0..1 | TimeDuration | The time interval (second) to wait for the VNF to be taken out of service during graceful termination, before shutting down the VNF and releasing the resources.  If not given, it is expected that the VNFM waits for the successful taking out of service of the VNF, no matter how long it takes, before shutting down the VNF and releasing the resources (see note).  Minimum timeout or timeout range are specified by the VNF Provider (e.g. defined in the VNFD or communicated by other means).  Not relevant in case of forceful termination. |

{

“vnfInstanceId”:”1”,

“terminationType”:”graceful”,

“gracefulTerminationTimeout”:”60”

}

* + - 1. **Response**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Qualifier | Cardinality | Content | Description |
| jobId | M | 1 | Identifier | Identifier of the VNF lifecycle operation occurrence.  [lifecycleOperationOccurrenceId**]** |

{

“jobId”:”1”

}

* + 1. **Query VNF**

|  |  |
| --- | --- |
| IF Definition | Description |
| URI | http(s)://[hostname][:port]/openoapi/{vnfmtype}/v1/{vnfmid}/vnfs/{vnfInstanceId} |
| Operation | GET |
| Direction | NSLCM->VNFMDriver |

* + - 1. **Request**

VNF filter: vnfInstanceId via url [R1]

* + - 1. **Response**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Qualifier | Cardinality | Content | Description |
| vnfInfo | M | 0..N | VnfInfo | The information items about the selected VNF instance(s) that are returned.  If attributeSelector is present, only the attributes listed in attributeSelector will be returned for the selected VNF instance(s).  See note. |

**VnfInfo Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Qualifier | Cardinality | Content | Description |
| vnfInstanceId | M | 1 | String | VNF instance identifier. |
| vnfInstanceName | M | 0..1 | String | VNF instance name. See note. |
| vnfInstanceDescription | M | 0..1 | String | Human-readable description of the VNF instance. |
| vnfdId | M | 1 | String | Identifier of the VNFD on which the VNF instance is based. |
| vnfPackageId | M | 0..1 | String | Identifier of the VNF Package used to manage the lifecycle of the VNF instance. See note.  Shall be present for an instantiated VNF instance. |
| version | M | 1 | String | Version of the VNF. |
| vnfProvider | M | 1 | String | Name of the person or company providing the VNF. |
| vnfType | M | 1 | String | VNF Application Type |
| vnfStatus | M | 1 | Enum | The instantiation state of the VNF. Possible values:  INACTIVE (Vnf is terminated or not instantiated ),  ACTIVE (Vnf is instantiated).  [instantiationState] |

{

“vnfInfo”:

{

"nfInstanceId":”1”,

"vnfInstanceName":”vFW”,

"vnfInstanceDescription":”vFW in Nanjing TIC Edge”,

"vnfdId":”1”,

"vnfPackageId":”1”,

"version":”V1.1”,

"vnfProvider":”ZTE”,

"vnfType":”vFW”,

"vnfStatus":” ACTIVE”,

}

}

* + 1. **Get operation status**

|  |  |
| --- | --- |
| IF Definition | Description |
| URI | http(s)://[hostname][:port]/openoapi/{vnfmtype} /v1/{vnfmid}/jobs/{jobid}&responseId={ responseId } |
| Operation | GET |
| Direction | NSLCM->VNFMDriver |

* + - 1. **Request**

None

* + - 1. **Response**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Qualifier | Cardinality | Content | Description |
| jobId | M | 1 | String | Job ID |
| responseDescriptor | M | 1 | - | Including:  vnfStatus，statusDescription，errorCode，progress、responseHistoryList、responseId |
| status | M | 1 | String | JOB status  started  processing  finished  error |
| progress | M | 1 | Integer | progress (1-100) |
| statusDescription | M | 1 | String | Progress Description |
| errorCode | M | 1 | Integer | Errorcode |
| responseId | M | 1 | Integer | Response Identifier |
| responseHistoryList | M | 0..n | ArrayList<> | History Response Messages from the requested responseId to lastest one.  Including fields:  vnfStatus,  statusDescription,  errorCode,  progress,  responseId |

{

"jobId" : "1234566",

"responseDescriptor" : {

"progress" : "40",

"status" : "proccessing",

"statusDescription" : "OMC VMs are decommissioned in VIM",

"errorCode" : null,

"responseId" : "42",

"responseHistoryList" : [{

"progress" : "40",

"status" : "proccessing",

"statusDescription" : "OMC VMs are decommissioned in VIM",

"errorCode" : null,

"responseId" : "1"

}, {

"progress" : "41",

"status" : "proccessing",

"statusDescription" : "OMC VMs are decommissioned in VIM",

"errorCode" : null,

"responseId" : "2"

}

]

}

}

* + 1. **Scale VNF**

|  |  |
| --- | --- |
| If Definition | Description |
| URI | http(s)://[hostname][:port]/openoapi/{vnfmtype}/v1/{vnfmid}/vnfs/{vnfInstanceId}/scale |
| Operation | POST |
| Direction | NSLCM->VNFMDriver |

* + - 1. **Request**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Qualifier | Cardinality | Content | Description |
| type | M | 1 | Enum | Defines the type of the scale operation requested (scale out, scale in). The set of types actually supported depends on the capabilities of the VNF being managed, as declared in the VNFD. See note 1. |
| aspectId | M | 1 | Identifier | Identifies the aspect of the VNF that is requested to be scaled, as declared in the VNFD. |
| numberOfSteps | M | 0..1 | Integer | Number of scaling steps to be executed as part of this ScaleVnf operation. It shall be a positive number.  Defaults to 1.  The VNF Provider defines in the VNFD whether or not a particular VNF supports performing more than one step at a time. Such a property in the VNFD applies for all instances of a particular VNF.  See note 2. |
| additionalParam | M | 0..N | KeyValuePair | Additional parameters passed by the NFVO as input to the scaling process, specific to the VNF being scaled.  **Reserved** |
| NOTE 1: ETSI GS NFV-IFA 010 [[2]](#_bookmark7) specifies that the lifecycle management operations that expand or contract a VNF instance include scale in, scale out, scale up and scale down. Vertical scaling (scale up, scale down) is not supported in the present document.  SCALE\_IN designates scaling in.  SCALE\_OUT 1 designates scaling out.  NOTE 2: A scaling step is the smallest unit by which a VNF can be scaled w.r.t a particular scaling aspect. | | | | |

{

“vnfInstanceId”:”5”,

“type”:” SCALE\_OUT”,

“aspectId”:”101”,

“numberOfSteps”:”1”,

“additionalParam”:{

……

}

}

* + - 1. **Response**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Qualifier | Cardinality | Content | Description |
| jobId | M | 1 | Identifier | The identifier of the VNF lifecycle operation occurrence. |

{

“jobId”:”1”

}