ETSI GS NFV-SOL 014 V3.5.1 (2021-05)

Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; YAML data model specification for descriptor-based virtualised resource management

Disclaimer

The present document has been produced and approved by the Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
RGS/NFV-SOL014ed351

Keywords
management, model, NFV

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies a set of YAML-based data models for descriptor-based virtualised resource management fulfilling the requirements concerning the input and output information exchanged over the virtualised resource management interfaces specified in the ETSI GS NFV-IFA 005 [1], and the ETSI GS NFV-IFA 006 [2]. The present document focuses on data models used in the virtualised resource descriptors for the Virtualised Compute interfaces, Virtualised Network interfaces and Virtualised Storage interfaces, which are used to perform orchestration and lifecycle management for consumable virtualised resources comprised of compute, network and storage. Other virtualised resource management interfaces, as well as data models for information specified in ETSI GS NFV-IFA 011 [i.5] and ETSI GS NFV-IFA 014 [i.4], are out of the scope of the present document.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI GS NFV-IFA 005: "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification".

[2] ETSI GS NFV-IFA 006: "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification".

[3] Void.

[4] "YAML Ain't Markup Language (YAML™) Version 1.2", 3rd Edition. Oren Ben-Kiki, Clark Evans, Ingy döt Net.

NOTE: Available at http://www.yaml.org/spec/1.2/spec.html.

[5] IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format".

NOTE: Available at https://tools.ietf.org/html/rfc8259.

[6] ETSI GS NFV-SOL 001: "Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on TOSCA specification".

[7] JSON Schema.

NOTE: Available at https://json-schema.org/.

[8] ETSI GS NFV-SOL 013: "Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; Specification of common aspects for RESTful NFV MANO APIs".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[i.2] Heat Orchestration Template (HOT) specification.

NOTE: Available at https://docs.openstack.org/heat/latest/template_guide/hot_spec.html.

[i.3] Openstack-heat - Orchestration service APIs.

NOTE: Available at https://docs.openstack.org/api-ref/orchestration/.

[i.4] ETSI GS NFV-IFA 014: "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Network Service Templates Specification".

[i.5] ETSI GS NFV-IFA 011: "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; VNF Descriptor and Packaging Specification".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS NFV 003 [i.1] apply.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS NFV 003 [i.1] and the following apply:

JSON	JavaScript Object Notation
YAML	YAML Ain't Markup Language

4 General aspects

4.1 Overview

The present document defines the data model for the following interfaces used over the Vi-Vnfm and Or-Vi reference point, using YAML [4] as a data-serialization language:

- Virtualised Compute interfaces.
- Virtualised Network interfaces.

- Virtualised Storage interfaces.

The design of the data model for the above interfaces is based on the information model and requirements defined in ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]. Protocols that use these data models are out of the scope of the present version of the present document.

In clause 4, general aspects are specified that apply to multiple data model on the Vi-Vnfm and Or-Vi reference point. The present document defines data models for input and output parameters derived from the above-mentioned information model. The data instances are used as input and output parameters specified in a virtualised resource descriptor, e.g. a HOT [i.2]. As an alternative, output parameters can also be obtained from an API provided by the template system of the underlying VIM implementation, e.g. the HEAT API [i.3], and be mapped to the data model defined in the present document.

In the subsequent clauses, the data model of the parameters to be used in virtualised resource descriptors as input and output for the individual interfaces are specified. Annex A provides examples of the use of the input and output parameters using HOT [i.2].

4.2 Definition of input and output parameters in YAML

4.2.1 Introduction

Clause 4.2 specifies the types and section definitions in YAML that are applicable for the present document, in particular, for the declaration of the input and output parameters.

4.2.2 Input parameters syntax definition

The set of parameters that are used as input to an operation for which a corresponding template is defined shall be prefixed by a tag named "nfv" and shall comply with the following YAML syntax definition:

```
nfv:
  <parameter_name>:
    type: <the type of parameter>
    description: <description of the parameter>
    default: <default value of the parameter>
    enum:
      - <enumerated values 1>
      - <enumerated values 2>
    ...
  <parameter_name_N>:
    ...
```

Where applicable, then name of a structured input parameter ends with the string "Data" (e.g. subnetData). A description of the syntax definition fields for declaring an input parameter follows. The fields shall comply with the provisions set out in Table 4.2.2-1.

Table 4.2.2-1: Input parameters syntax definition

Field	Required	Description
nfv	yes	The tag emphasizes a group of parameters defined in the present document.
<parameter_name_1>	yes	The name of the first parameter.
<parameter_name_N>	no	The name of the last parameter.
type	yes	The type of each parameter. It shall be a simple data type as defined in clause 4.4.2 or structured data types in clause 4.4.3.
description	yes	A human readable description for each parameter.
default	no	A default value for each parameter.
enum	no	A set of enumerated values for a parameter to restrict the value. It is applicable to parameters of type string or number.

4.2.3 Output parameters syntax definition

If a set of output parameters of an operation is defined in a template, these parameters shall comply with the following YAML [4] syntax definition:

```
<parameter_name>: value
  description: <description of the parameter>
  type: <type>
```

Where applicable, then name of a structured output parameter ends with the string "Info" (e.g. nfvSubnetInfo). A description of the syntax definition fields for declaring an output parameter follows. The fields shall comply with the provisions set out in Table 4.2.3-1.

Table 4.2.3-1: Output parameters syntax definition

Field	Required	Description
parameter_name	yes	The name of the parameter, which shall start with the prefix "nfv".
type	yes	The type of the parameter.
description	yes	A human readable description for the parameter.

4.3 Definition of output parameters as mapping to an API

The present document defines the set of attributes for each output parameter in the data model in clauses 6, 7 and 8. Besides providing the output parameters that are defined in the data model using the output parameters facility of a template (e.g. parameters in the "outputs" section of a HOT [i.2]), it is also possible to obtain these parameters via VIM-levels APIs such as (such as the HEAT API [i.3]). In the latter case, the output parameters of a VIM-level API can be mapped to the data model for the output parameters defined in the present document. Taking this approach can offer performance advantages in case many resources are required to be managed by the same template. The choice of the mapping of a parameter to a template output parameter, or to a VIM-level API is a deployment decision outside the scope of the present document.

4.4 Common data types

4.4.1 Introduction

Clause 4.4 specifies the common data types that are used for declaring the parameters and grammar elements throughout the present document.

4.4.2 Simple data types

The present specification uses the following simple data types as defined in Table 4.4.2-1. In order to accommodate tags with a broader meaning, the YAML specification recommends JSON schema [7] to be supported as an option. JSON schema is commonly supported by modern computing languages. Virtualised resource descriptors complying with the present document shall comply with the YAML v1.2 [4] and JSON schema [7] specifications.

Table 4.4.2-1: Simple data types

Type name	Description	Example(s)
String	A string as defined in YAML v1.2 [4].	"a string"
Number	A number as defined in IETF RFC 8259 [5] referred in JSON Schema [7].	"23", "-1.023E3"
Boolean	A data type that can take the following values: true, false. The type is defined in JSON Schema [7] and referred in YAML v1.2 [4].	"true", "false"

4.4.3 Structured data types

Following the format stated with the label of "nfv" in Table 4.4.3-1, individual structured data type is represented in the present document using ">" recursively as inline definition.

Table 4.4.3-1: Input or Output data model for {parameter name}

Parameter Name and Attributes	Type	Description
{parameter name}	{object, array}	Type of the parameter
{description}	-	Description of the parameter
{attribute}	{attribute type}	Type of {attribute}
>{sub attribute}	{sub attribute type in the attribute}	Type of {sub attribute}

`object` in JSON schema [7] is a type representing mapping from "keys" to "values". The syntax of `object` for parameter definition is represented with the following definition:

```
{parameter name}:
  description: <description of the parameter>
  type: object
  required:
    - {1st mandatory attribute}
    - {2nd mandatory attribute}
    - …
  properties:
  {1st attribute}:
      type: e.g. object
      properties:
        {sub attribute}
  {2nd attribute}:
   …
```

`array` in JSON schema [7] is a type representing an ordered list of elements. The syntax of `array` for parameter definition is represented with the following definition:

```
{parameter name}:
  description: <description of the parameter>
  type: array
  minItems: {lower bound of cardinality}
  maxItems: {upper bound of cardinality}
  items:
  - type: e.g. object
    properties:
      {sub attribute}
```

5 Common data model

5.1 Description

This clause specifies data models for input and output parameters commonly used in different resource management.

5.2 Parameters to be used as input

5.2.1 Parameter: reservationId

The parameter used when pointing to a virtualised compute, network or storage resource shall follow the indications provided in Table 5.2.1-1.

Table 5.2.1-1: Input data model for reservationId

Parameter Name and Attributes	Type	Description
reservationId	String	Identifier of the resource reservation applicable to this virtualised resource management operation

The syntax of the reservationId shall comply with the following definition:

```
  reservationId:
    type: string
    description: >
      Identifier of the resource reservation applicable to this virtualised resource
      management operation
    default: ""
```

5.2.2 Parameter: resourceGroupId

The parameter used when pointing to a logical grouping of virtual resources assigned to a tenant shall follow the indications provided in Table 5.2.2-1.

Table 5.2.2-1: Input data model for resourceGroupId

Parameter Name and Attributes	Type	Description
resourceGroupId	String	Unique identifier of the "infrastructure resource group", logical grouping of virtual resources assigned to a tenant within an Infrastructure Domain

The syntax of the resourceGroupId shall comply with the following definition:

```
  resourceGroupId:
    description: >
      The identifier of the infrastructure resource group, logical grouping of virtual
      resources assigned to a tenant within an Infrastructure Domain of this
      virtualised resource management operation
    type: string
    default: ""
```

5.2.3 Parameter: groupName

The parameter used when giving a group name of a virtualised compute, network or storage resource affinity or anti-affinity constraints group to be created shall follow the indications provided in Table 5.2.3-1.

Table 5.2.3-1: Input data model for groupName

Parameter Name and Attributes	Type	Description
groupName	String	Name of the group, given by the consumer

The syntax of the groupName shall comply with the following definition:

```
  groupName:
    type: string
    description: >
      Name of the group, given by the consumer
    default: ""
```

5.2.4 Parameter: typeOfAffinityOrAntiAffinityConstraints

The parameter used when indicating whether this is an affinity or anti-affinity group for virtualised compute, network or storage resources shall follow the indications provided in Table 5.2.4-1.

Table 5.2.4-1: Input data model for `typeOfAffinityOrAntiAffinityConstraints`

Parameter Name and Attributes	Type	Description
typeOfAffinityOrAntiAffinityConstraints	String	Indicates whether this is an affinity or anti-affinity group

The syntax of the `typeOfAffinityOrAntiAffinityConstraints` shall comply with the following definition:

```
  typeOfAffinityOrAntiAffinityConstraints:
    description: >
      Indicates whether this is an affinity or anti-affinity group.
    type: string
    enum:
      - affinity
      - anti-affinity
```

5.3 Parameters to be used as output

None.

6 Data model for Virtualised Compute Management

6.1 Description

This clause specifies data models for input and output parameters for Virtualised Compute Management.

6.2 Parameters to be used as input

6.2.1 Parameter: computeName

The parameter used when providing a name for a virtualised compute resource to be allocated shall follow the indications provided in Table 6.2.1-1.

Table 6.2.1-1: Input data model for computeName

Parameter Name and Attributes	Type	Description
computeName	String	Name for a virtualised compute resource to be allocated

The syntax of the computeName shall comply with the following definition:

```
  computeName:
    type: string
    description: >
      Name provided by the consumer for the virtualised compute resource to
      allocate
    default: ""
```

6.2.2 Parameter: computeFlavourId

The parameter used when providing an identifier of the Compute Flavour for a virtualised compute resource to be allocated shall follow the indications provided in Table 6.2.2-1.

Table 6.2.2-1: Input data model for computeFlavourId

Parameter Name and Attributes	Type	Description
computeFlavourId	String	Identifier of the Compute Flavour that provides information about the particular memory, CPU and disk resources for virtualised compute resource to allocate

The syntax of the computeFlavourId shall comply with the following definition:

```
  computeFlavourId:
    type: string
    description: >
      Identifier of the Compute Flavour that provides information about the particular
      memory, CPU and disk resources for virtualised compute resource to allocate
    default: ""
```

6.2.3 Parameter: vcImageId

The parameter used when providing an identifier of the virtualisation container software image for a virtualised compute resource to be allocated shall follow the indications provided in Table 6.2.3-1.

Table 6.2.3-1: Input data model for vcImageId

Parameter Name and Attributes	Type	Description
vcImageId	String	Identifier of the virtualisation container software image

The syntax of the vcImageId shall comply with the following definition:

```
   vcImageId:
    type: string
    description: >
      Identifier of the virtualisation container software image
    default: ""
```

6.2.4 Parameter: locationConstraints

The parameter used when providing a location constraints for a virtualised compute resource to be allocated shall follow the indications provided in Table 6.2.4-1.

Table 6.2.4-1: Input data model for locationConstraints

Parameter Name and Attributes	Type	Description
locationConstraints	String	If present, it defines location constraints for the resource(s) is (are) requested to be allocated, e.g. in what particular resource zone

The syntax of the locationConstraints shall comply with the following definition:

```
  locationConstraints:
    type: string
    description: >
      If present, it defines location constraints for the resource(s) is (are)
requested to be allocated, e.g. in what particular resource zone.
    default: ""
```

6.2.5 Parameter: affinityOrAntiAffinityConstraintsForCompute

The parameter used when giving resource affinity or anti-affinity constraints related to virtualised compute resources shall follow the indications provided in Table 6.2.5-1. The parameter is a list of elements with affinity or anti affinity information of the virtualised compute resource to be allocated ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]. All the listed constraints shall be fulfilled for a successful operation.

Table 6.2.5-1: Input data model for affinityOrAntiAffinityConstraintsForCompute

Parameter Name and Attributes	Type	Description
affinityOrAntiAffinityConstraintsForCompute	Array of Object	Name of the parameter.
>typeOfAffinityOrAntiAffinityConstraintForCompute	String	Indicates whether this is an affinity or anti-affinity constraint. Allowed to affinity and anti-affinity.
>scopeOfAffinityOrAntiAffinityConstraintForCompute	String	Qualifies the scope of the constraint. In case of compute resource: e.g. "NFVI-PoP" or "NFVI-Node". Allowed to NFVI-PoP, NFVI-Node. Defaults to "NFVI-Node" if absent.
>affinityAntiAffinityResourceList	Object	Consumer-managed list of identifiers of virtualised resources with which the actual resource is requested to be affine or anti-affine. See note and condition.
>>resource	Array of Object	List of identifiers of virtualised resources.
>affinityAntiAffinityResourceGroup	String	Identifier of the producermanaged group of virtualised resources with which the actual resource is requested to be affine or anti-affine. See note and condition.
NOTE: It is a prerequisite for the consumer to create a VirtualisedComputeResourceAffinityOrAntiAffinityConstraintsGroup and get groupIdentifier using the appropriate operation, Create Virtualised Compute Resource Affinity Or AntiAffinity Constraints Group, defined in ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2].		
CONDITION: If explicit resource lists for affinity/anti-affinity (see clause 8.4.8.1 in ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]) are supported, the affinityAntiAffinityResourceList shall be supported. If named resource groups for affinity/anti-affinity (see clause 8.4.8.1 in ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]) are supported, affinityAntiAffinityResoruceGroup shall be supported. The mechanisms shall not be mixed in the scope of a resourceGroup (aka VIM tenant).		

The syntax of the affinityOrAntiAffinityConstraintsForCompute shall comply with the following definition:

```
affinityOrAntiAffinityConstraintsForCompute:
  description: >
    A list of elements with affinity or antiaffinity information of
    the virtualised compute resource to allocate.
  oneOf:
    - type: array
      minItems: 0  # lower bound of cardinality
      maxItems: N  # upper bound of cardinality
      items:
        type: object
        required:
          - typeOfAffinityOrAntiAffinityConstraintForCompute
        properties:
          typeOfAffinityOrAntiAffinityConstraintForCompute:
            type: string
            enum:
              - affinity
              - anti-affinity
          scopeOfAffinityOrAntiAffinityConstraintForCompute:
            type: string
            enum:
              - NFVI-PoP
              - NFVI-Node
            default: NFVI-Node
          affinityAntiAffinityResourceList:
            type: object
            required:
              - resource
            properties:
              resource:
                type: array
                minItems: 1  # lower bound of cardinality
                maxItems: N  # upper bound of cardinality
                items:
                  type: string
    - type: array
      minItems: 0  # lower bound of cardinality
```

```
      maxItems: N  # upper bound of cardinality
      items:
        type: object
        required:
          - typeOfAffinityOrAntiAffinityConstraintForCompute
        properties:
          typeOfAffinityOrAntiAffinityConstraintForCompute:
            type: string
            enum:
              - affinity
              - anti-affinity
          scopeOfAffinityOrAntiAffinityConstraintForCompute:
            type: string
            enum:
              - NFVI-PoP
              - NFVI-Node
            default: NFVI-Node
          affinityAntiAffinityResourceGroup:
            type: string
```

6.2.6 Parameter: interfaceData

The parameter used when giving interfaceData related to virtualised compute resources shall follow the indications provided in Table 6.2.6-1. The parameter is a list of data about network interface data which are specific to a Virtual Compute Resource instance.

NOTE: ">" is used to specify an "inline definition".

Table 6.2.6-1: Input data model for `interfaceData`

Parameter Name and Attributes	Type	Description
interfaceData	Array of Object	Name of the parameter.
>ipAddress	Array of Object	The virtual network interface can be configured with specific IP address(es) associated to the network to be attached to.
>macAddress	String	The MAC address desired for the virtual network interface.

The syntax of the `interfaceData` shall comply with the following definition:

```
  interfaceData:  # VirtualInterfaceData IE in ETSI GS NFV-IFA 005 and ETSI GS NFV-IFA 006
    description: >
      The data of network interfaces which are specific to a Virtual Compute
      Resource instance
    type: array
    minItems: 0  # lower bound of cardinality
    maxItems: N  # upper bound of cardinality
    items:
      type: object
      properties:
        ipAddress:  # IpAddress IE in SOL013
          type: array
          minItems: 0  # lower bound of cardinality
          maxItems: N  # upper bound of cardinality
          items:
            type: string
        macAddress:  # MacAddress IE in ETSI GS NFV-SOL 013
          type: string
```

6.2.7 Parameter: computeId

The parameter used when pointing to an identifier of the virtualised compute resource to operate shall follow the indications provided in Table 6.2.7-1.

Table 6.2.7-1: Input data model for computeId

Parameter Name and Attributes	Type	Description
computeId	String	Identifier of the virtualised compute resource to operate

The syntax of the computeId shall comply with the following definition:

```
computeId:
  type: string
  description: >
    Identifier of the virtualised compute resource to operate
  default: ""
```

6.2.8 Parameter: networkInterfaceNew

The parameter used when giving networkInterfaceNew related to virtualised compute resources shall follow the indications provided in Table 6.2.8-1. The parameter is a list of data about new virtual network interface(s) to add to the compute resource.

NOTE: ">" is used to specify an "inline definition".

Table 6.2.8-1: Input data model for `networkInterfaceNew`

Parameter Name and Attributes	Type	Description
networkInterfaceNew	Array of Object	Name of the parameter.
>networkId	String	In the case when the virtual network interface is attached to the network, it identifies such a network.
>networkPortId	String	If the virtual network interface is attached to a specific network port, it identifies such a network port.
>typeVirtualNic	String (see note)	Type of network interface. Allowed value: normal-virtual-NIC.
>typeConfiguration	Array of String (see note)	Extra configuration that the virtual network interface supports based on the type of virtual network interface.
>bandwidth	Number	The bandwidth of the virtual network interface (in Mbps).
>accelerationCapabilityForVirtualNetworkInterface	Array of String (see note)	It specifies if the virtual network interface requires certain acceleration capabilities (e.g. RDMA, packet dispatch, TCP Chimney).
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
NOTE: `networkInterfaceNew` parameter is used in Update Virtualised Compute Resource operation. In the case, the virtualised compute resource has been allocated with resource constraints (e.g. supported hardware). The new network interface, extra configurations and acceleration capability may not be accepted if those requests are unmatched to the constraints.		

The syntax of the `networkInterfaceNew` shall comply with the following definition:

```
networkInterfaceNew:  # VirtualNetworkInterfaceData in ETSI GS NFV-IFA 005 and ETSI GS NFV-IFA 006
  description: >
    The new virtual network interface(s) to add to the compute resource.
  type: array
  minItems: 0  # lower bound of cardinality
  maxItems: N  # upper bound of cardinality
  items:
    type: object
    required:
      - typeVirtualNic
    properties:
      networkId:
        type: string
      networkPortId:
        type: string
      typeVirtualNic:
        type: string
      typeConfiguration:
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
```

```
            type: string
          bandwidth:
            type: number
          accelerationCapabilityForVirtualNetworkInterface:
            type: array
            minItems: 0  # lower bound of cardinality
            maxItems: N  # upper bound of cardinality
            items:
              type: string
          metadata:
            description: >
              metadata is optional. It is out of scope to detail what are the sub-keys and possible
values
            type: array
            minItems: 0  # lower bound of cardinality
            maxItems: N  # upper bound of cardinality
            items:
              type: object
```

6.2.9 Parameter: networkInterfaceUpdate

The parameter used when giving networkInterfaceUpdate related to virtualised compute resources shall follow the indications provided in Table 6.2.9-1. The parameter is a list of data about virtual network interface(s) to update on the compute resource.

NOTE: ">" is used to specify an "inline definition".

Table 6.2.9-1: Input data model for `networkInterfaceUpdate`

Parameter Name and Attributes	Type	Description
networkInterfaceUpdate	Array of Object	Name of the parameter
>resourceId	String	Identifier of the virtual network interface.
>ownerId	String	Identifier of the owner of the network interface (e.g. a virtualised compute resource).
>networkId	String	In the case when the virtual network interface is attached to the network, it identifies such a network.
>networkPortId	String	If the virtual network interface is attached to a specific network port, it identifies such a network port.
>ipAddress	Array of String	The virtual network interface can be configured with specific IP address(es) associated to the network to be attached to.
>typeVirtualNic	String (see note)	Type of network interface. The type allows for defining how such interface is to be realized, e.g. normal virtual NIC, with direct PCI pass-through, etc.
>typeConfiguration	Array of String (see note)	Extra configuration that the virtual network interface supports based on the type of virtual network interface, including support for SR-IOV with configuration of Virtual Functions (VF).
>macAddress	String	The MAC address of the virtual network interface.
>bandwidth	Number	The bandwidth of the virtual network interface (in Mbps).
>accelerationCapabilityForVirtualNetworkInterface	Array of String (see note)	Shows the acceleration capabilities utilized by the virtual network interface.
>operationalState	String	The operational state of the virtual network interface. Allowed value: enabled, disabled.
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
NOTE: `networkInterfaceUpdate` parameter is used in Update Virtualised Compute Resource operation. In the case, the virtualised compute resource has been allocated with resource constraints (e.g. supported hardware). The new network interface, extra configurations and accelerationCapabilityForVirtualNetworkInterface may not be accepted if those requests are unmatched to the constraints.		

The syntax of the `networkInterfaceUpdate` shall comply with the following definition:

```
  networkInterfaceUpdate:  # VirtualNetworkInterface IE in ETSI GS NFV-IFA 005 and ETSI GS NFV-IFA
006
    description: >
      The virtual network interface(s) to update on the compute resource.
    type: array
    minItems: 0  # lower bound of cardinality
    maxItems: N  # upper bound of cardinality
    items:
      type: object
      required:
        - resourceId
        - ownerId
        - typeVirtualNic
        - macAddress
        - bandwidth
        - operationalState
      properties:
        resourceId:
          type: string
        ownerId:
          type: string
        networkId:
          type: string
        networkPortId:
          type: string
        ipAddress:  # IpAddress IE in ETSI GS NFV-SOL 013
          type: array
          minItems: 0  # lower bound of cardinality
          maxItems: N  # upper bound of cardinality
          items:
            type: string
        typeVirtualNic:
          type: string
        typeConfiguration:
          type: array
          minItems: 0  # lower bound of cardinality
          maxItems: N  # upper bound of cardinality
          items:
            type: string
        macAddress:
          type: string
        bandwidth:
          type: number
        accelerationCapabilityForVirtualNetworkInterface:
          type: array
          minItems: 0  # lower bound of cardinality
          maxItems: N  # upper bound of cardinality
          items:
            type: string
        operationalState:
          type: string
          enum:
            - enabled
            - disabled
        metadata:
          description: >
            metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
          type: array
          minItems: 0  # lower bound of cardinality
          maxItems: N  # upper bound of cardinality
          items:
            type: object
```

6.2.10 Parameter: flavour

The parameter used when requesting operations related to the creation of flavours shall follow the indications provided in Table 6.2.10-1. This parameter is applicable only for Or-Vi interface.

NOTE: ">" is used to specify an "inline definition".

Table 6.2.10-1: Input data model for `flavour`

Parameter Name and Attributes	Type	Description
flavour	Object	Name of the parameter.
>flavourId	String	Identifier given to the compute flavour.
>accelerationCapabilityForVirtualComputeFlavour	Array of String	Selected acceleration capabilities (e.g. crypto, GPU) from the set of capabilities offered by the compute node acceleration resources.
>virtualMemory	Object	The virtual memory of the virtualised compute.
>>virtualMemSize	Number	Amount of virtual Memory (e.g. in MB).
>>virtualMemOversubscriptionPolicy	String	The memory core oversubscription policy in terms of virtual memory to physical memory on the platform. The cardinality can be 0 during the allocation request, if no particular value is requested. E.g. virtual memory : physical memory.
>>numaEnabled	Boolean	It specifies the memory allocation to be cognisant of the relevant process/core allocation. The cardinality can be 0 during the allocation request, if no particular value is requested.
>virtualCpu	Object	The virtual CPU(s) of the virtualised compute. The cardinality can be 0 during the allocation request, if no particular CPU architecture type is requested.
>>cpuArchitecture	String	CPU architecture type. Examples are x86, ARM®.
>>numVirtualCpu	Number	Number of virtual CPUs.
>>cpuClock	Number	Minimum CPU clock rate (e.g. in MHz) available for the virtualised CPU resources. The cardinality can be 0 during the allocation request, if no particular value is requested.
>>virtualCpuOversubscriptionPolicy	String	The CPU core oversubscription policy, e.g. the relation of virtual CPU cores to physical CPU cores/threads. The cardinality can be 0 during the allocation request, if no particular value is requested. E.g. virtual CPU core : physical CPU core= 4:1.
>>virtualCpuPinning	Object	The virtual CPU pinning configuration for the virtualised compute resource.
>>>vitualCpuPinningPolicy	String	The policy can take values of "static" or "dynamic". In case of "static" the virtual CPU cores are requested to be allocated to logical CPU cores according to the rules defined in virtualCpuPinningRules. In case of "dynamic" the allocation of virtual CPU cores to logical CPU cores is decided by the VIM (e.g. SMT (Simultaneous Multi-Threading) requirements). Allowed value: static, dynamic.
>>>virtualCpuPinningRules	Array of Object	A list of rules that should be considered during the allocation of the virtual CPU-s to logical CPU-s in case of "static" virtualCpuPinningPolicy.
>>>>cores	Number	The number of core in the virtual CPU.
>>>>sockets	Number	The number of socket in the virtual CPU.
>>>>threads	Number	The number of thread in the virtual CPU.
>storageAttributes	Array of Object	Element containing information about the size of virtualised storage resource (e.g. size of volume, in GB), the type of storage (e.g. volume, object), and support for RDMA.
>>typeOfStorage	String	Type of virtualised storage resource (e.g. volume, object).
>>sizeOfStorage	Number	Size of virtualised storage resource (e.g. size of volume, in GB).
>virtualNetworkInterface	Array of Object	The virtual network interfaces of the virtualised compute.
>>networkId	String	In the case when the virtual network interface is attached to the network, it identifies such a network. The cardinality can be 0 in the case that a network interface is created without being attached to any specific network.

Parameter Name and Attributes	Type	Description
>>networkPortId	String	If the virtual network interface is attached to a specific network port, it identifies such a network port. The cardinality can be 0 in the case that a network interface is created without any specific network port attachment.
>typeVirtualNic	Not specified (see note)	Type of network interface. The type allows for defining how such interface is to be realized, e.g. normal virtual NIC, with direct PCI pass-through, etc.
>typeConfiguration	Not specified (see note)	Extra configuration that the virtual network interface supports based on the type of virtual network interface.
>>bandwidth	Number	The bandwidth of the virtual network interface (in Mbps).
>>accelerationCapabilityForVirtualNetworkInterface	Array of String	It specifies if the virtual network interface requires certain acceleration capabilities (e.g. RDMA, packet dispatch, TCP Chimney). The cardinality can be 0, if no particular acceleration capability is requested.
>>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
NOTE: There is only part of `flavour` as specified in ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2] are included in this version of the present document, the following are attributes not included: • typeVirtualNic; • typeConfiguration.		

The syntax of the `flavour` shall comply with the following definition:

```
flavour:
  description: >
    The flavour provides information about the particular memory, CPU
    and disk resources for virtualised compute resource to allocate
  type: object
  required:
    - flavourId
    - virtualMemory
    - virtualCpu
  properties:
    flavourId:
      type: string
    accelerationCapabilityForVirtualComputeFlavour:
      type: array
      minItems: 0  # lower bound of cardinality
      maxItems: N  # upper bound of cardinality
      items:
        type: string
    virtualMemory:
      type: object
      required:
        - virtualMemSize
      properties:
        virtualMemSize:
          type: number
        virtualMemOversubscriptionPolicy:
          type: string
        numaEnabled:
          type: boolean
    virtualCpu:
      type: object
      required:
        - numVirtualCpu
      properties:
        cpuArchitecture:
          type: string
        numVirtualCpu:
          type: number
        cpuClock:
          type: number
```

```
          virtualCpuOversubscriptionPolicy:
            type: string
          virtualCpuPinning:
            type: object
            required:
              - cpuPinningPolicy
            properties:
              cpuPinningPolicy:
                type: string
                enum:
                  - static
                  - dynamic
              cpuPinningRules:
                type: array
                minItems: 0  # lower bound of cardinality
                maxItems: N  # upper bound of cardinality
                items:
                  type: object
                  properties:
                    cores:
                      type: number
                    sockets:
                      type: number
                    threads:
                      type: number
      storageAttributes:
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
          properties:
            typeOfStorage:
              type: string
            sizeOfStorage:
              type: number
      virtualNetworkInterface:
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
          properties:
            networkId:
              type: string
            networkPortId:
              type: string
            bandwidth:
              type: number
            accelerationCapabilityForVirtualNetworkInterface:
              type: array
              minItems: 0  # lower bound of cardinality
              maxItems: N  # upper bound of cardinality
              items:
                type: string
              metadata:
                description: >
                  metadata is optional. It is out of scope to detail what are the sub-keys and
possible values.
                type: array
                minItems: 0  # lower bound of cardinality
                maxItems: N  # upper bound of cardinality
                items:
                  type: object
```

6.3 Parameters to be used as output

6.3.1 Parameter: nfvComputeInfo

The parameter is used when returning information for a virtualised compute resource, and its output data model shall follow the indications provided in Table 6.3.1-1. This parameter maps to the "computeData" parameter defined in ETSI GS NFV-IFA 005 [1].

Table 6.3.1-1: Output data model for nfvComputeInfo

Parameter Name and Attributes	Type	Description
nfvComputeInfo	Object	Element containing information of the newly instantiated virtualised compute resource as VirtualCompute.
>computeId	String	Identifier of the virtualised compute resource.
>computeName	String	Name of the virtualised compute resource.
>flavourId	String	Identifier of the given compute flavour used to instantiate this virtual compute.
>accelerationCapabilityForVirtualComputeFlavour	Array of String	Selected acceleration capabilities (e.g. crypto, GPU) from the set of capabilities offered by the compute node acceleration resources.
>virtualCpu	Object	The virtual CPU(s) of the virtualised compute as VirtualCpu.
>>cpuArchitecture	String	CPU architecture type. Examples are x86, ARM®. See note.
>>numVirtualCpu	Number	Number of virtual CPUs.
>>cpuClock	Number	Minimum CPU clock rate in Hz available for the virtualised CPU resources.
>>virtualCpuOversubscriptionPolicy	String	The CPU core oversubscription policy, e.g. the relation of virtual CPU cores to physical CPU cores/threads. The cardinality can be 0 if no policy has been defined during the allocation request.
>>virtualCpuPinning	Object	The virtual CPU pinning configuration for the virtualised compute resource.
>>>cpuPinningPolicy	String	The policy can take values of "static" or "dynamic". In case of "static" the virtual CPU cores are requested to be allocated to logical CPU cores according to the rules defined in virtualCpuPinningRules. In case of "dynamic" the allocation of virtual CPU cores to logical CPU cores is decided by the VIM (e.g. SMT (Simultaneous Multi-Threading) requirements). Allowed value: static, dynamic.
>>>cpuPinningRules	Array of Object	A list of rules that should be considered during the allocation of the virtual CPU-s to logical CPU-s in case of "static" virtualCpuPinningPolicy.
>>>>core	Number	The number of core in the virtual CPU.
>>>>sockets	Number	The number of socket in the virtual CPU.
>>>>threads	Number	The number of thread in the virtual CPU.
>virtualMemory	Object	The virtual memory of the compute as VirtualMemory.
>>virtualMemSize	Number	Amount of virtual memory in byte.
>>virtualMemOversubscriptionPolicy	String	The memory core oversubscription policy in terms of virtual memory to physical memory on the platform. The cardinality can be 0 if no policy has been defined during the allocation request.
>>numaEnabled	Boolean	It specifies the memory allocation to be cognisant of the relevant process/core allocation.
>virtualNetworkInterface	Array of Object	Element with information of the instantiated virtual network interfaces of the compute resource.
>>resourceId	String	Identifier of the virtual network interface.
>>ownerId	String	Identifier of the owner of the network interface (e.g. a virtualised compute resource).
>>networkId	String (Reference to VirtualNetwork)	In the case when the virtual network interface is attached to the network, it identifies such a network. The cardinality can be 0 in the case that a network interface is created without being attached to any specific network.
>>networkPortId	String (Reference to VirtualNetworkPort)	If the virtual network interface is attached to a specific network port, it identifies such a network port. The cardinality can be 0 in the case that a network interface is created without any specific network port attachment.

Parameter Name and Attributes	Type	Description
>>ipAddress	Array of String	The virtual network interface can be configured with specific IP address(es) associated to the network to be attached to. The cardinality can be 0 in the case that a network interface is created without being attached to any specific network, or when an IP address can be automatically configured, e.g. by DHCP.
>>typeVirtualNic	String	Type of network interface. The type allows for defining how such interface is to be realized, e.g. normal virtual NIC, with direct PCI pass-through, etc.
>>typeConfiguration	Array of String	Extra configuration that the virtual network interface supports based on the type of virtual network interface, including support for SR-IOV with configuration of Virtual Functions (VF).
>>macAddress	String	The MAC address of the virtual network interface.
>>bandwidth	Number	The bandwidth of the virtual network interface (in Mbps).
>> accelerationCapabilityForVirtualNetworkInterface	Array of String	Shows the acceleration capabilities utilized by the virtual network interface. The cardinality can be 0, if no acceleration capability is utilized.
>>operationalState	String	The operational state of the virtualised subnetwork. Allowed values are: enabled, disabled.
>>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
>>virtualDisks	Array of Object	Element with information of the virtualised storage resources (volumes, ephemeral) that are attached to the compute resource.
>>>storageId	String	Identifier of the virtualised storage resource.
>>>storageName	String	Name of the virtualised storage resource.
>>>flavourId	String	Identifier of the storage flavour used to instantiate this virtual storage.
>>>typeOfStorage	String	Type of virtualised storage resource (e.g. volume, object).
>>>sizeOfStorage	Number	Size of virtualised storage resource (e.g. size of volume, in GB).
>>>rdmaEnabled	Boolean	Indicates if the storage supports RDMA.
>>>ownerId	String	Identifier of the virtualised resource that owns and uses such a virtualised storage resource. The value can be NULL if the virtualised storage is not attached yet to any other resource (e.g. a virtual machine).
>>>zoneId	String	It identifies the resource zone where the virtual storage resources have been allocated.
>>>hostId	String	Identifier of the host where the virtualised storage resource is allocated.
>>>operationalState	String	Operational state of the resource. Allowed value: enabled, disabled.
>>>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
>vcImageId	String	Identifier of the virtualisation container software image (e.g. virtual machine image).
>zoneId	String	If present, it identifies the resource zone where the virtual compute resources have been allocated.
>hostId	String	Identifier of the host the virtualised compute resource is allocated on.
>operationalState	String	Operational state of the compute resource. Possible values are: "enabled" or "disabled".

Parameter Name and Attributes	Type	Description
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
NOTE: See "cpu_architecture" in tosca.datatypes.nfv.VirtualCpu, ETSI GS NFV-SOL 001 [6].		

When used as an output parameter in a template, the syntax of the nfvComputeInfo shall comply with the following definition:

```
  nfvComputeInfo:
    description: >
      Element containing information of the newly instantiated virtualised
      compute resource.
    type: object
    required:
      - computeId
      - flavourId
      - virtualCpu
      - virtualMemory
      - virtualDisks
      - hostId
      - operationalState
    properties:
      computeId:
        description: >
          Identifier of the virtualised compute resource.
        type: string
      computeName:
        description: >
          Name of the virtualised compute resource.
        type: string
      flavourId:
        description: >
          Identifier of the given compute flavour used to instantiate this
          virtual compute.
        type: string
      accelerationCapabilityForVirtualComputeFlavour:
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: string
      virtualCpu:
        description: >
          The virtual CPU(s) of the virtualised compute.
        type: object
          properties:
            cpuArchitecture:
              description: >
                CPU architecture type.
              type: string
            numVirtualCpu:
              description: >
                Number of virtual CPUs.
              type: number
            cpuClock:
              description: >
                Minimum CPU clock rate in Hz available for the virtualised
                CPU resources.
              type: number
            virtualCpuOversubscriptionPolicy:
              description: >
                The CPU core oversubscription policy, e.g. the relation of
                virtual CPU cores to physical CPU cores/threads. The cardinality
                can be 0 if no policy has been defined during the allocation request.
              type: string
            virtualCpuPinning:
              description: >
                The virtual CPU pinning configuration for the virtualised
                compute resource.
              type: object
              required:
                - cpuPinningPolicy
```

```
            properties:
              cpuPinningPolicy:
                type: string
                enum:
                  - static
                  - dynamic
              cpuPinningRules:
                type: array
                minItems: 0  # lower bound of cardinality
                maxItems: N  # upper bound of cardinality
                items:
                  type: object
                  properties:
                    cores:
                      type: number
                    sockets:
                      type: number
                    threads:
                      type: number
    virtualMemory:
      description: >
        The virtual memory of the compute.
      type: object
      properties:
        virtualMemSize:
          description: >
            Amount of virtual memory in byte.
          type: number
        virtualMemOversubscriptionPolicy:
          description: >
            The memory core oversubscription policy in terms of virtual memory
            to physical memory on the platform. The cardinality can be 0 if
            no policy has been defined during the allocation request.
          type: string
        numaEnabled:
          description: >
            It specifies the memory allocation to be cognisant of
            the relevant process/core allocation.
          type: boolean
    virtualNetworkInterface:
      description: >
        Element with information of the instantiated virtual network
        interfaces of the compute resource.
      resourceId:
        type: string
      ownerId:
        type: string
      networkId:
        type: string
      networkPortId:
        type: string
      ipAddress:  # IpAddress IE in ETSI GS NFV-SOL 013
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # maximum value of cardinality
        items:
          type: string
      typeVirtualNic:
        type: string
      typeConfiguration:
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: string
      macAddress:
        type: string
      bandwidth:
        type: number
      accelerationCapabilityForVirtualNetworkInterface":
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: string
      operationalState:
        type: string
        enum:
```

```
            - enabled
            - disabled
        metadata:
          description: >
            metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
          type: array
          minItems: 0  # lower bound of cardinality
          maxItems: N  # upper bound of cardinality
          items:
            type: object
      virtualDisks:
        description: >
          Element with information of the virtualised storage resources
          (volumes, ephemeral) that are attached to the compute resource.
        type: array
        minItems: 1  # lower bound of cardinality
        maxItems: N  # maximum value of cardinality
        items:
          type: object
          required:
            - storageId
            - flavourId
            - typeOfStorage
            - sizeOfStorage
            - operationalState
          properties:
            storageId:
              description: >
                Identifier of the virtualised storage resource
              type: string
            storageName:
              description: >
                Name of the virtualised storage resource
              type: string
            flavourId:
              description: >
                Identifier of the storage flavour used to instantiate
                this virtual storage
              type: string
            typeOfStorage:
              description: >
                Type of virtualised storage resource
              type: string
            sizeOfStorage:
              description: >
                Size of virtualised storage resource
              type: number
            rdmaEnabled:
              description: >
                Indicates if the storage supports RDMA.
              type: boolean
            ownerId:
              description: >
                Identifier of the virtualised resource that owns and uses such
                a virtualised storage resource. The value can be NULL if the
                virtualised storage is not attached yet to any other resource
              type: string
            zoneId:
              description: >
                It identifies the resource zone where the virtual storage
                resources have been allocated
              type: string
            hostId:
              description: >
                Identifier of the host where the virtualised storage resource
                is allocated.
              type: string
            operationalState:
              description: >
                Operational state of the resource.
              type: string
              enum:
                - enabled
                - disabled
            metadata:
              description: >
```

```
              metadata is optional. It is out of scope to detail what are the sub-keys and
possible values.
            type: array
            minItems: 0  # lower bound of cardinality
            maxItems: N  # upper bound of cardinality
            items:
              type: object
      vcImageId:
        description: >
          Identifier of the virtualisation container software image.
        type: string
      zoneId:
        description: >
          If present, it identifies the resource zone where the virtual
          compute resources have been allocated.
        type: string
      hostId:
        description: >
          Identifier of the host the virtualised compute resource is allocated on.
        type: string
      operationalState:
        description: >
          Operational state of the compute resource.
        type: string
        enum:
          - enabled
          - disabled
      metadata:
        description: >
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
```

7 Data model for Virtualised Network Management

7.1 Description

This clause specifies data models for input and output parameters for Virtualised Network Management.

7.2 Parameters to be used as input

7.2.1 Parameter: networkResourceName

The parameter used when providing a name for a virtualised network resource shall follow the indications provided in Table 7.2.1-1.

Table 7.2.1-1: Input data model for networkResourceName

Parameter Name and Attributes	Type	Description
networkResourceName	String	Name for a virtualised compute resource.

The syntax of the networkResourceName shall comply with the following definition:

```
  networkResourceName:
    description: >
      Name provided by the consumer for the virtualised network resource
    type: string
    default: ""
```

7.2.2 Parameter: networkResourceType

The parameter used when setting the type of a virtualised network resource shall follow the indications provided in Table 7.2.2-1.

Table 7.2.2-1: Input data model for networkResourceType

Parameter Name and Attributes	Type	Description
networkResourceType	String	The network data provides information about the particular virtual network resource. Possible values are: "network", "subnet", or "network-port".

The syntax of the networkResourceType shall comply with the following definition:

```
networkResourceType:
  description: >
    The network data information applicable to the particular virtual network
    resource of the virtualised resource management operation
  type: string
  enum:
  - network
  - subnet
  - network-port
  default: ""
```

7.2.3 Parameter: typeNetworkData

The parameter used when providing the network data information about the particular virtual network shall follow the indications provided in Table 7.2.3-1.

Table 7.2.3-1: Input data model for typeNetworkData

Parameter Name and Attributes	Type	Description
typeNetworkData	Object	The network data provides information about the particular virtual network resource.
>bandwidth	Number	Minimum network bandwidth (in Mbps).
>networkType	String	The type of network that maps to the virtualised network. This list is extensible. Examples are: "local", "vlan", "vxlan", "gre", "l3-vpn", etc.
>segmentType	String	The isolated segment for the virtualised network. For instance, for a "vlan" networkType, it corresponds to the vlan identifier; and for a "gre" networkType, this corresponds to a gre key.
>networkQos	Array of Object	Element providing information about Quality of Service attributes that the network is requested to support.
>>qosName	String	Name given to the QoS parameter.
>>qosValue	Number	Value of the QoS parameter.
>isShared	Boolean	It defines whether the virtualised network is shared among consumers.
>sharingCriteria	String	Only present for shared networks. Indicate the sharing criteria/constraint for this network. These criteria might be a list of authorized consumers.
>layer3Attributes	Arrayof Object	The attribute list allows setting up a network providing defined layer 3 connectivity.
>>networkId	String	The identifier of the virtualised network that the virtualised sub-network is attached to.
>>ipVersion	String	The IP version of the network/subnetwork. Allowed_values: IPv4, IPv6.
>>gatewayIp	String	Specifies the IP address of the network/subnetwork gateway when the gateway is selected by the requestor.
>>cidr	String	The CIDR of the network/subnetwork, i.e. network address and subnet mask.
>>isDhcpEnabled	Boolean	True when DHCP is to be enabled for this network/subnetwork, or false otherwise.

Parameter Name and Attributes	Type	Description
>>addressPool	Array of Object	Address pools for the network/subnetwork.
>>>start	String	The first IP address in the addressPool. See note.
>>>end	String	The last IP address in the addressPool. See note.
>>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. Metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. Metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
NOTE: In case of an IPV4 address, string that consists of four decimal integers separated by dots, each integer ranging from 0 to 255. In case of an IPV6 address, string that consists of groups of zero to four hexadecimal digits, separated by colons.		

The syntax of the typeNetworkData shall comply with the following definition:

```
typeNetworkData:
  description: >
    The network data information about the particular virtual network
    resource of the virtualised resource management operation
  required:
    - bandwidth
  type:  object  # VirtualNetworkData IE in ETSI GS NFV-IFA 005 and ETSI GS NFV-IFA 006
  properties:
    bandwidth:
      type: number
    networkType:
      type: string
    segmentType:
      type: string
    networkQos:
      type: array
      minItems: 0  # lower bound of cardinality
      maxItems: N  # upper bound of cardinality
      items:
        type: object
        properties:
          qosName:
            type: string
          qosValue:
            type: number
    isShared:
      type: boolean
    sharingCriteria:
      type: string
    layer3Attributes:  # NetworkSubnetData IE in ETSI GS NFV-IFA 005 and ETSI GS NFV-IFA 006
      type: array
      minItems: 0  # lower bound of cardinality
      maxItems: N  # upper bound of cardinality
      items:
        type: object
        properties:
          networkId:
            type: string
          ipVersion:
            type: string
            enum:
            - IPv4
            - IPv6
          gatewayIp:
            type: string
          cidr:
            type: string
          isDhcpEnabled:
            type: boolean
          addressPool:
            type: array
            minItems: 0  # lower bound of cardinality
```

```
              maxItems: N  # upper bound of cardinality
              items:
                type: object
                properties:
                  start:
                    type: string
                  end:
                    type: string
            metadata:
              description: >
                metadata is optional. It is out of scope to detail what are the sub-keys and
possible values.
              type: array
              minItems: 0  # lower bound of cardinality
              maxItems: N  # upper bound of cardinality
              items:
                type: object
      metadata:
        description: >
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
    default: ""
```

7.2.4 Parameter: typeNetworkPortData

The parameter used when setting the network port data provides information about the particular network port shall follow the indications in Table 7.2.4-1.

Table 7.2.4-1: Input data model for typeNetworkPortData

Parameter Name and Attributes	Type	Description
typeNetworkPortData	Object	The network port data provides information about the particular network port.
>portType	String	Type of network port. Examples of types are access ports (layer 2 or 3), or trunk ports (layer 1) that become transport for multiple layer 2 or layer 3 networks.
>networkId	String	Identifier of the network that the port belongs to.
>segmentId	String	The isolated segment the network port belongs to. For instance, for a "vlan", it corresponds to the vlan identifier; and for a "gre", this corresponds to a gre key.
>bandwidth	Number	The bandwidth of the virtual network port (in Mbps).
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.

The syntax of the typeNetworkPortData shall comply with the following definition:

```
  typeNetworkPortData:
    description: >
      The network port data information about the particular network port
      of the virtualised resource management operation
    required:
      - portType
    type:  object  # VirtualNetworkPortData IE in ETSI GS NFV-IFA 005 and ETSI GS NFV-IFA 006
    properties:
      portType:
        type: string
      networkId:
        type: string
      segmentId:
        type: string
      bandwidth:
        type: number
      metadata:
        description: >
```

```
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
    default: ""
```

7.2.5 Parameter: typeSubnetData

The parameter used when setting the subnet data information about the particular subnetwork resource shall follow the indications in Table 7.2.5-1.

Table 7.2.5-1: Input data model for typeSubnetData

Parameter Name and Attributes	Type	Description
typeSubnetData	Object	The subnet data provides information about the particular sub-network resource.
>networkId	String	The identifier of the virtualised network that the virtualised sub-network is attached to.
>ipVersion	String	The IP version of the network/subnetwork. Allowed Value: IPv4, IPv6.
>gatewayIp	String	The identifier of the virtualised network that the virtualised sub-network is attached to.
>cidr	String	The IP version of the network/subnetwork. Allowed Value: IPv4, IPv6.
>isDhcpEnabled	Boolean	Specifies the IP address of the network/subnetwork gateway when the gateway is selected by the requestor.
>addressPool	Array of Object	The CIDR of the network/subnetwork, i.e. network address and subnet mask.
>>start	String	The first IP address in the addressPool. See note.
>>end	String	The last IP address in the addressPool. See note.
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
NOTE: In case of an IPV4 address, string that consists of four decimal integers separated by dots, each integer ranging from 0 to 255. In case of an IPV6 address, string that consists of groups of zero to four hexadecimal digits, separated by colons.		

The syntax of the typeSubnetData shall comply with the following definition:

```
  typeSubnetData:
    description: >
      The subnet data information about the particular subnetwork of
      the virtualised resource management operation
    type:  object  # NetworkSubnetData IE in ETSI GS NFV-IFA 005 and ETSI GS NFV-IFA 006
    properties:
      networkId:
        type: string
      ipVersion:
        type: string
        enum:
        - IPv4
        - IPv6
      gatewayIp:
        type: string
      cidr:
        type: string
      isDhcpEnabled:
        type: boolean
      addressPool:
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
```

```
        items:
          type: object
          properties:
            start:
              type: string
            end:
              type: string
      metadata:
        description: >
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
    default: ""
```

7.2.6 Parameter: affinityOrAntiAffinityConstraintsForNetwork

The parameter used when providing the list of elements with affinity or anti affinity information of the virtualised network resource shall follow the indications in Table 7.2.6-1.

Table 7.2.6-1: Input data model for affinityOrAntiAffinityConstraintsForNetwork

Parameter Name and Attributes	Type	Description
affinityOrAntiAffinityConstraintsForNetwork	Array of Object	A list of elements with affinity or anti affinity information of the virtualised network resource. All the listed constraints shall be fulfilled for a successful operation.
>typeOfAffinityOrAntiAffinityConstraintForNetwork	String	Indicates whether this is an affinity or anti-affinity constraint. Allowed_values: affinity, anti-affinity.
>scopeOfAffinityOrAntiAffinityConstraintForNetwork	String	Qualifies the scope of the constraint. In case of ports: e.g. "virtual switch or router" or "physical NIC", or "physical network" or "NFVI Node". In case of networks: e.g. "physical NIC", "physical network" or "NFVI Node". In case of subnets: it should be ignored. Defaults to "NFVI Node" if absent. Allowed_values: virtual-switch, router, physical-NIC, physical-network, NFVI-Node.
>affinityAntiAffinityResourceList	Array	Consumer-managed list of identifiers of virtualised resources with which the actual resource is requested to be affine or anti-affine. See note and condition.
>>resource	Array of String	List of identifiers of virtualised resources.
>affinityAntiAffinityResourceGroup	String	Identifier of the producermanaged group of virtualised resources with which the actual resource is requested to be affine or anti-affine. See note and condition.
NOTE: It is a prerequisite for the consumer to create a `VirtualisedNetworkResourceAffinityOrAntiAffinityConstraintsGroup` and get `groupIdentifier` using the appropriate operation, Create Virtualised Network Resource Affinity Or AntiAffinity Constraints Group, defined in ETSI GS NFV-IFA 005 [1], and the ETSI GS NFV-IFA 006 [2].		
CONDITION: If explicit resource lists for affinity/anti-affinity (see clause 8.4.8.1 in ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]) are supported, the `affinityAntiAffinityResourceList` shall be supported. If named resource groups for affinity/anti-affinity (see clause 8.4.8.1 in ETSI GS NFV-IFA 005 [1], and the ETSI GS NFV-IFA 006 [2]) are supported, `affinityAntiAffinityResoruceGroup` shall be supported. The mechanisms shall not be mixed in the scope of a resourceGroup (aka VIM tenant).		

The syntax of the affinityOrAntiAffinityConstraintsForNetwork shall comply with the following definition:

```
affinityOrAntiAffinityConstraintsForNetwork:
  description: >
    A list of elements with affinity or anti affinity information of the virtualised
    network resource of the virtualised resource management
    operation
  oneOf:
    - type: array
      minItems: 0  # lower bound of cardinality
      maxItems: N  # upper bound of cardinality
      items:
        required:
          - typeOfAffinityOrAntiAffinityConstraintForNetwork
        type: object
        properties:
          typeOfAffinityOrAntiAffinityConstraintForNetwork:
            type: string
            enum:
            - affinity
            - anti-affinity
          scopeOfAffinityOrAntiAffinityConstraintForNetwork:
            type: string
            enum:
            - virtual-switch
            - router
            - physical-NIC
            - physical-network
            - NFVI-Node
            default: NFVI-Node
          affinityAntiAffinityResourceList:
            required:
              - resource
            type: object
            properties:
              resource:
                type: array
                minItems: 1  # lower bound of cardinality
                maxItems: N  # upper bound of cardinality
                items:
                  type: string
    - type: array
      minItems: 0  # lower bound of cardinality
      maxItems: N  # upper bound of cardinality
      items:
        required:
          - typeOfAffinityOrAntiAffinityConstraintForNetwork
        type: object
        properties:
          typeOfAffinityOrAntiAffinityConstraintForNetwork:
            type: string
            enum:
            - affinity
            - anti-affinity
          scopeOfAffinityOrAntiAffinityConstraintForNetwork:
            type: string
            enum:
            - virtual-switch
            - router
            - physical-NIC
            - physical-network
            - NFVI-Node
            default: NFVI-Node
          affinityAntiAffinityResourceGroup:
            type: string
  default: ""
```

7.2.7 Void

7.2.8 Parameter: locationConstraintsForNetwork

The parameter used when defining the location constraints for the resource(s) shall follow the indicators provided in Table 7.2.8-1.

Table 7.2.8-1: Input data model for locationConstraints

Parameter Name and Attributes	Type	Description
locationConstraintsForNetwork	String	Defines location constraints for the resource(s), e.g. in what particular resource zone.

The syntax of the locationConstraintsForNetwork shall comply with the following definition:

```
  locationConstraintsForNetwork:
    description: >
      The definition of the location constraints for the resource(s),
      e.g. in what particular resource zone, of the virtualised resource management
      operation
    type: string
    default: ""
```

7.2.9 Parameter: queryNetworkFilter

The parameters used when invoking the operation shall follow the indications provided in Table 7.2.9-1.

Table 7.2.9-1: Input data model for queryNetworkFilter

Parameter Name and Attributes	Type	Description
queryNetworkFilter	Not specified (see note)	Query filter based on e.g. name, identifier, metadata information or status information, expressing the type of information to be retrieved. It can also be used to specify one or more resources to be queried by providing their identifiers.
NOTE: Query operation is not covered in the present document.		

The syntax of the queryNetworkFilter shall comply with the following definition:

```
  queryNetworkFilter:
    description: >
      The query filter based on name, identifier, metadata information or status
      information, expressing the type of information to be retrieved of the
      virtualised resource management operation
    type: Not specified
    default: ""
```

7.2.10 Parameter: networkResourceId

The parameter used when pointing to a virtualised network resource shall follow the indications provided in Table 7.2.10-1.

Table 7.2.10-1: Input data model for networkResourceId

Parameter Name and Attributes	Type	Description
networkResourceId	String	Identifier of a virtualised resource.

The syntax of the networkResourceId shall comply with the following definition:

```
  networkResourceId:
    description: >
      Identifier of a virtualised network resource
    type: string
    default: ""
```

7.2.11 Parameter: updateNetworkData

The parameter used when providing the network data information about the particular virtual network shall follow the indications provided in Table 7.2.11-1.

Table 7.2.11-1: Input data model for updateNetworkData

Parameter Name and Attributes	Type	Description
updateNetworkData	Object	Network data information about the particular virtual network resource.
>bandwidth	Number	Minimum network bandwidth (in Mbps).
>networkType	String	The type of network that maps to the virtualised network. This list is extensible. Examples are: "local", "vlan", "vxlan", "gre", "l3-vpn", etc.
>segmentType	String	The isolated segment for the virtualised network. For instance, for a "vlan" networkType, it corresponds to the vlan identifier; and for a "gre" networkType, this corresponds to a gre key.
>networkQos	Array of Object	Element providing information about Quality of Service attributes that the network is requested to support.
>>qosName	String	Name given to the QoS parameter.
>>qosValue	Number	Value of the QoS parameter.
>isShared	Boolean	It defines whether the virtualised network is shared among consumers.
>sharingCriteria	String	Only present for shared networks. Indicate the sharing criteria/constraint for this network. These criteria might be a list of authorized consumers.
>layer3Attributes	Array of Object	The attribute list allows setting up a network providing defined layer 3 connectivity.
>>networkId	String	The identifier of the virtualised network that the virtualised sub-network is attached to.
>>ipVersion	String	The IP version of the network/subnetwork. Allowed_values: IPv4, IPv6.
>>gatewayIp	String	Specifies the IP address of the network/subnetwork gateway when the gateway is selected by the requestor.
>>cidr	String	The CIDR of the network/subnetwork, i.e. network address and subnet mask.
>>isDhcpEnabled	Boolean	True when DHCP is to be enabled for this network/subnetwork, or false otherwise.
>>addressPool	Array of Object	Address pools for the network/subnetwork.
>>>start	String	The first IP address in the addressPool. See note.
>>>end	String	The last IP address in the addressPool. See note.
>>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. Metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. Metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
NOTE: In case of an IPV4 address, string that consists of four decimal integers separated by dots, each integer ranging from 0 to 255. In case of an IPV6 address, string that consists of groups of zero to four hexadecimal digits, separated by colons.		

The syntax of the updateNetworkData shall comply with the following definition:

```
  updateNetworkData:
    description: >
      This element contains the network data information of a particular
      virtual network resource
    required:
      - bandwidth
    type:  object  # VirtualNetworkData in ETSI GS NFV-IFA 005 and ETSI GS NFV-IFA 006
    properties:
      bandwidth:
        type: number
      networkType:
```

```
        type: string
      segmentType:
        type: string
      networkQos:
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # maximum value of cardinality
        items:
          type: object
          required:
            - qosName
            - qosValue
          properties:
            qosName:
              type: string
            qosValue:
              type: number
      isShared:
        type: boolean
      sharingCriteria:
        type: string
      layer3Attributes:  # NetworkSubnetData IE in ETSI GS NFV-IFA 005 and ETSI GS NFV-IFA 006
        type: array
        items:
          type: object
          properties:
            networkId:
              type: string
            ipVersion:
              type: string
              enum:
              - IPv4
              - IPv6
            gatewayIp:
              type: string
            cidr:
              type: string
            isDhcpEnabled:
              type: boolean
            addressPool:
              type: array
              minItems: 0  # lower bound of cardinality
              maxItems: N  # upper bound of cardinality
              items:
                type: object
                  properties:
                    start:
                      type: string
                    end:
                      type: string
            metadata:
              description: >
                metadata is optional. It is out of scope to detail what are the sub-keys and
possible values.
              type: array
              minItems: 0  # lower bound of cardinality
              maxItems: N  # upper bound of cardinality
              items:
                - type: object
      metadata:
        description: >
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
    default: ""
```

7.2.12 Parameter: updateSubnetData

The parameter used when setting the subnet data information about the particular subnetwork resource shall follow the indications in Table 7.2.12-1.

Table 7.2.12-1: Input data model for updateSubnetData

Parameter Name and Attributes	Type	Description
updateSubnetData	Object	Subnet data information about the particular virtual subnet resource.
>networkId	String	The identifier of the virtualised network that the virtualised sub-network is attached to.
>ipVersion	String	The IP version of the network/subnetwork. Allowed Value: IPv4, IPv6.
>gatewayIp	String	Specifies the IP address of the network/subnetwork gateway when the gateway is selected by the requestor.
>cidr	String	The CIDR of the network/subnetwork, i.e. network address and subnet mask.
>isDhcpEnabled	Boolean	True when DHCP is to be enabled for this network/subnetwork, or false otherwise.
>addressPool	Array of Object	Address pools for the network/subnetwork.
>>start	String	The first IP address in the addressPool. See note.
>>end	String	The last IP address in the addressPool. See note.
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource.
NOTE: In case of an IPV4 address, string that consists of four decimal integers separated by dots, each integer ranging from 0 to 255. In case of an IPV6 address, string that consists of groups of zero to four hexadecimal digits, separated by colons.		

The syntax of the updateSubnetData shall comply with the following definition:

```
  updateSubnetData:
    description: >
      The subnet data information of a particular subnet resource
    type: object
    properties:
      networkId:
        type: string
      ipVersion:
        type: string
        enum:
        - IPv4
        - IPv6
      gatewayIp:
        type: string
      cidr:
        type: string
      isDhcpEnabled:
        type: boolean
      addressPool:
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # maximum value of cardinality
        items:
          type: object
          properties:
            start:
              type: string
            end:
              type: string
      metadata:
        description: >
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
    default: ""
```

7.2.13 Parameter: updateNetworkPort

The parameter used when providing the network port data provides information about the particular network port shall follow the indications in Table 7.2.13-1.

Table 7.2.13-1: Input data model for updateNetworkPort

Parameter Name and Attributes	Type	Description
updateNetworkPort	Object	Network port data information about the particular virtual network port resource.
>portType	String	Type of network port. Examples of types are access ports (layer 2 or 3), or trunk ports (layer 1) that become transport for multiple layer 2 or layer 3 networks.
>networkId	String	Identifier of the network that the port belongs to.
>segmentId	String	The isolated segment the network port belongs to. For instance, for a "vlan", it corresponds to the vlan identifier; and for a "gre", this corresponds to a gre key.
>bandwidth	Number	The bandwidth of the virtual network port (in Mbps).
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource.

The syntax of the updateNetworkPort shall comply with the following definition:

```
  updateNetworkPort:
    description: >
      The network port data information of a particular network port
      resource
    required:
      - portType    type:  object  # VirtualNetworkData in ETSI GS NFV-IFA 005 and ETSI GS NFV-IFA
006
    properties:
      portType:
        type: string
      networkId:
        type: string
      segmentId:
        type: string
      bandwidth:
        type: number
      metadata:
        description: >
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
    default: ""
```

7.2.14 Parameter: scopeOfAffinityOrAntiAffinityConstraintForNetwork

The parameter used when providing the type of the affinity or anti-affinity group shall follow the indications in Table 7.2.14-1.

Table 7.2.14-1: Input data model for scopeOfAffinityOrAntiAffinityConstraintForNetwork

Parameter Name and Attributes	Type	Description
scopeOfAffinityOrAntiAffinityConstraintForNetwork	String	Qualifies the scope of the constraint, e.g. NFVI Node, NIC. Defaults to NFVI Node if absent.

The syntax of the scopeOfAffinityOrAntiAffinityConstraintForNetwork shall comply with the following definition:

```
scopeOfAffinityOrAntiAffinityConstraintForNetwork:
  description: >
    It qualifies the scope of the constraint, e.g. NFVI Node, NIC of the
    virtualised resource management operation
  type: string
  enum:
  - NFVI-Node
  - NIC
  default: NFVI-Node
```

7.3 Parameters to be used as output

7.3.1 Parameter: nfvNetworkInfo

The parameter is used when returning information for a virtualised network resource, and its output data model shall follow the indications provided in Table 7.3.1-1. This parameter maps to the "networkData" parameter defined in ETSI GS NFV-IFA 005 [1].

Table 7.3.1-1: Output data model for nfvNetworkInfo

Parameter Name and Attributes	Type	Description
nfvNetworkInfo	Object	If network types are created satisfactorily, it contains the data relative to the instantiated virtualised network resource as VirtualNetwork.
>networkResourceId	String	Identifier of the virtualised network resource.
>networkResourceName	String	Name of the virtualised network resource.
>subnet	String	Only present if the network provides layer 3 connectivity.
>networkPort	Not specified (see note)	Element providing information of an instantiated virtual network port.
>bandwidth	Number	Minimum network bandwidth (in Mbps).
>networkType	String	The type of network that maps to the virtualised network. Examples are: "local", "vlan", "vxlan", "gre", "l3-vpn". The cardinality can be "0" to cover the case where this attribute is not required to create the virtualised network.
>segmentType	String	The isolated segment for the virtualised network. For instance, for a "vlan" networkType, it corresponds to the vlan identifier; and for a "gre" networkType, this corresponds to a gre key. The cardinality can be "0" for flat networks without any specific segmentation.
>networkQoS	Array of Object	Element providing information about Quality of Service attributes that the network supports. Cardinality can be "0" for virtual network without any QoS requirements.
>>qosName	String	Name given to the QoS parameter.
>>qosValue	Number	Value of the QoS parameter.
>isShared	Boolean	It defines whether the virtualised network is shared among consumers.
>sharingCriteria	String	Only present for shared networks. Indicate the sharing criteria for this network. This criteria might be a list of authorized consumers.
>zoneId	String	If present, it identifies the Resource Zone where the virtual network resources have been allocated.
>operationalState	String	The operational state of the virtualised network. Possible values are: "enabled", "disabled".
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. Metadata is optional. It is out of scope to detail what are the sub-keys and possible values.

Parameter Name and Attributes	Type	Description
NOTE: In Allocate Virtualised Network Resource operation output parameters, ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2], networkPort attribute is specified with duplication. The data model is specified as an attribute in `networkData` parameter as well as `networkPortData` as a parameter in the operation.		

When used as an output parameter in a template, the syntax of the `networkInfo` shall comply with the following definition:

```
  nfvNetworkInfo:
    type: object
    required:
      - networkResourceId
      - bandwidth
      - networkType
      - isShared
      - operationalState
    properties:
      networkResourceId:
        type: string
      networkResourceName:
        type: string
      subnet:
        type: string
      bandwidth:
        type: number
      networkType:
        type: string
      segmentType:
        type: string
      networkQoS:
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # maximum value of cardinality
        items:
          - type: object
            properties:
              qosName:
                type: string
              qosValue:
                type: number
      isShared:
        type: boolean
      sharingCriteria:
        type: string
      zoneId:
        type: string
      operationalState:
        type: string
        description: >
          Operational state of the compute resource.
        enum:
          - enabled
          - disabled
      metadata:
        type: array
        description: >
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
```

7.3.2 Parameter: nfvSubnetInfo

The parameter is used when returning information for a subnet resource, and its output data model shall follow the indications provided in Table 7.3.2-1. This parameter maps to the "subnetData" parameter defined in ETSI GS NFV-IFA 005 [1].

Table 7.3.2-1: Output data model for nfvSubnetInfo

Parameter Name and Attributes	Type	Description
nfvSubnetInfo	Object	If subnet types are created satisfactorily, it contains the data relative to the allocated subnet as NetworkSubnet.
>resourceId	String	Identifier of the virtualised sub-network.
>networkId	String	The identifier of the virtualised network that the virtualised sub-network is attached to. The cardinality can be 0 to cover the case where this type is used to describe the L3 attributes of a network rather than a subnetwork.
>ipVersion	String	The IP version of the network/subnetwork. Possible values are: "IPv4", "IPv6".
>gatewayIp	String	The IP V4 or IPV6 address of the network/subnetwork gateway.
>cidr	String	The CIDR of the network/subnetwork, i.e. network address and subnet mask.
>isDhcpEnabled	Boolean	True when DHCP is enabled for this network/subnetwork, or false otherwise.
>addressPool	Array of Object	Address pools for the network/subnetwork. The cardinality can be 0 when VIM is allowed to allocate all addresses in the CIDR except for the address of the network/subnetwork gateway.
>>start	String	The first IP address in the addressPool. See note.
>>end	String	The last IP address in the addressPool. See note.
>operationalState	String	The operational state of the virtualised subnetwork. Allowed values are: enabled, disabled.
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.
NOTE: In case of an IPV4 address, string that consists of four decimal integers separated by dots, each integer ranging from 0 to 255. In case of an IPV6 address, string that consists of groups of zero to four hexadecimal digits, separated by colons.		

When used as an output parameter in a template, the syntax of the `nfvSubnetInfo` shall comply with the following definition:

```
nfvSubnetInfo:
  type: object
  required:
    - resourceId
    - ipVersion
    - gatewayIp
    - cidr
    - isDhcpEnabled
    - operationalState
  object:
    resourceId:
      type: string
    networkId:
      type: string
    ipVersion:
      type: string
      enum:
        - IPv4
        - IPv6
    gatewayIp:
      type: string
    cidr:
      type: string
    isDhcpEnabled:
      type: boolean
    addressPool:
      type: array
      items:
        - type: object
          properties:
            start:
              type: string
```

```
              end:
                type: string
      operationalState:
        type: string
        enum:
          - enabled
          - disabled
      metadata:
        type: array
        description: >
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
```

7.3.3 Parameter: nfvNetworkPortInfo

The parameter is used when returning information for a network port resource, and its output data model shall follow the indications provided in Table 7.3.3-1. This parameter maps to the "networkPortData" parameter defined in ETSI GS NFV-IFA 005 [1].

Table 7.3.3-1: Output data model for nfvNetworkPortInfo

Parameter Name and Attributes	Type	Description
nfvNetworkPortInfo	Object	If network port types are created satisfactorily, it contains the data relative to the allocated network port as VirtualNetworkPort.
>resourceId	String	Identifier of the virtual network port.
>networkId	String	Identifier of the network that the port belongs to.
>attachedResourceId	String	Identifier of the attached resource to the network port (e.g. a virtualised compute resource, or identifier of the virtual network interface). The cardinality can be "0" if there is no specific resource connected to the network port.
>portType	String	Type of network port. Examples of types are access ports (layer 2 or 3), or trunk ports (layer 1) that become transport for multiple layer 2 or layer 3 networks. Possible values are: "access ports", "trunk ports".
>segmentId	String	The isolated segment the network port belongs to. For instance, for a "vlan", it corresponds to the vlan identifier; and for a "gre", this corresponds to a gre key. The cardinality can be "0" for flat networks without any specific segmentation.
>bandwidth	Number	The bandwidth of the virtual network port (in Mbps). Cardinality can be "0" for virtual network ports without any specific allocated bandwidth.
>operationalState	String	The operational state of the virtualised network port. Possible values are: "enabled", "disabled".
>metadata	Array of Object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.

When used as an output parameter in a template, the syntax of the `nfvNetworkPortInfo` shall comply with the following definition:

```
  nfvNetworkPortInfo:
    type: object
    required:
      - resourceId
      - portType
      - operationalState
    properties:
      resourceId:
        type: string
      attachedResourceId:
        type: string
      portType:
        type: string
```

```
        enum:
          - access ports
          - trunk ports
      segmentId:
        type: string
      bandwidth:
        type: number
      operationalState:
        type: string
        enum:
          - enabled
          - disabled
      metadata:
        type: array
        description: >
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
```

8 Data model for Virtualised Storage Management

8.1 Description

This clause specifies data models for input and output parameters for Virtualised Storage Management.

8.2 Parameters to be used as input

8.2.1 Parameter: storageName

The parameter used when providing a name for a virtualised storage resource shall follow the indications provided in Table 8.2.1-1.

Table 8.2.1-1: Input data model for storageName

Parameter Name and Attributes	Type	Description
storageName	String	Name provided by the consumer for the virtualised storage resource to allocate. It can be used for identifying resources from consumer side.

The syntax of the `storageName` shall comply with the following definition:

```
  storageName:
    description: >
      Name provided by the consumer for the virtualised storage resource to allocate.
      It can be used for identifying resources from consumer side.
    type: string
    default: ""
```

8.2.2 Parameter: affinityOrAntiAffinityConstraintsForStorage

The parameter used when giving resource affinity or anti-affinity constraints related to virtualised storage resources shall follow the indications provided in Table 8.2.2-1. The parameter is a list of elements with affinity or anti affinity information of the virtualised storage resource to be allocated ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]. All the listed constraints shall be fulfilled for a successful operation.

Table 8.2.2-1: Input data model for affinityOrAntiAffinityConstraintsForStorage

Parameter Name and Attributes	Type	Description
affinityOrAntiAffinityConstraintsForStorage	Array of Object	A list of elements with affinity or anti-affinity information of the virtualised storage resource to be allocated.
>typeOfAffinityOrAntiAffinityConstraintForStorage	String	Indicates whether this is an affinity or anti-affinity constraint. Allowed to affinity and anti-affinity.
>scopeOfAffinityOrAntiAffinityConstraintForStorage	String	Qualifies the scope of the constraint for the virtualised storage resource. In case of storage resource: e.g. NFVI-Node. Persistent storage node is a type of NFVI-Node which supports, for example, Object, Block or File-based storage service. Ephemeral storage service is supported in a compute node. So this is not included in this attribute. Allowed to NFVI-Node. Defaults to "NFVI-Node" if absent.
>affinityAntiAffinityResourceList	Object	Consumer-managed list of identifiers of virtualised resources with which the actual resource is requested to be affine or anti-affine. See note and condition.
>>resource	Array of String	List of identifiers of virtualised resources.
>affinityAntiAffinityResoruceGroup	String	Identifier of the producer managed group of virtualised resources with which the actual resource is requested to be affine or anti-affine. See note and condition.
NOTE: It is a prerequisite for the consumer to create a `VirtualisedStorageResourceAffinityOrAntiAffinityConstraintsGroup` and get `groupIdentifier` using the appropriate operation, Create Virtualised Storage Resource Affinity Or AntiAffinity Constraints Group, defined in ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2].		
CONDITION: If explicit resource lists for affinity/anti-affinity (see clause 8.4.8.1 in ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]) are supported, the `affinityAntiAffinityResourceList` shall be supported. If named resource groups for affinity/anti-affinity (see clause 8.4.8.1 in ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]) are supported, `affinityAntiAffinityResoruceGroup` shall be supported. The mechanisms shall not be mixed in the scope of a resourceGroup (aka VIM tenant).		

The syntax of the affinityOrAntiAffinityConstraintsForCompute shall comply with the following definition:

```
  affinityOrAntiAffinityConstraintsForStorage:
    description: >
      A list of elements with affinity or anti-affinity information of
      the virtualised storage resource to allocate.
    oneOf:
      - type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
          required:
            - typeOfAffinityOrAntiAffinityContrraintForStorage
          properties:
            typeOfAffinityOrAntiAffinityConstraintForStorage:
              type: string
              enum:
                - affinity
                - anti-affinity
            scopeOfAffinityOrAntiAffinityConstraintForStorage:
              type: string
              enum:
                - NFVI-Node
              default: NFVI-Node
            affinityAntiAffinityResourceList:
              type: object
              required:
```

```
              - resource
            properties:
              resource:
                type: array
                minItems: 1  # lower bound of cardinality
                maxItems: N  # upper bound of cardinality
                items:
                  type: string
    - type: array
      minItems: 0  # lower bound of cardinality
      maxItems: N  # upper bound of cardinality
      items:
        type: object
        required:
          - typeOfAffinityOrAntiAffinityContrraintForStorage
        properties:
          typeOfAffinityOrAntiAffinityConstraintForStorage:
            type: string
            enum:
              - affinity
              - anti-affinity
          scopeOfAffinityOrAntiAffinityConstraintForStorage:
            type: string
            enum:
              - NFVI-Node
            default: NFVI-Node
          affinityAntiAffinityResourceGroup:
            type: string
```

8.2.3 Parameter: storageData

The parameter used when providing information about the type and size of the storage shall follow the indications provided in Table 8.2.3-1.

Table 8.2.3-1: Input data model for `storageData`

Parameter Name and Attributes	Type	Description
storageData	Object	The storage data provides information about the type and size of the storage.
NOTE: This storageData is specified for input parameter with VirtualStorageFlavour IE in allocate Virtualised Storage Resource operation.		

The syntax of the `storageData` shall comply with the following definition:

```
  storageData:
    description: >
      The storage data provides information about the type and size of the storage.
    type: object
    required:
      - flavourId
      - storageAttributes
    properties:
      flavourId:
        type: string
      storageAttributes:
        type: object
        properties:
          typeOfStorage:
            type: string
          sizeOfStorage:
            type: number
```

8.2.4 Parameter: updateStorageData

The parameter used when providing information about the type and size of the storage to be updated shall follow the indications provided in Table 8.2.4-1.

Table 8.2.4-1: Input data model for `updateStorageData`

Parameter Name and Attributes	Type	Description
updateStorageData	Object	The element contains the fields that can be updated of a storage resource.

The syntax of the `updateStorageData` shall comply with the following definition:

```
updateStorageData:
  description: >
    The element contains the fields that can be updated of a storage resource.
  type: object
  required:
    - flavourId
    - storageAttributes
  properties:
    flavourId:
      type: string
    storageAttributes:
      typeOfStorage:
        type: string
      sizeOfStorage:
        type: number
```

8.2.5 Parameter: storageOperation

The parameter used when providing a type of operation for a virtualised storage operation shall follow the indications provided in Table 8.2.5-1.

Table 8.2.5-1: Input data model for storageOperation

Parameter Name and Attributes	Type	Description
storageOperation	String	Type of operation to perform on the virtualised storage resource. Possible values include: "create snapshot", and "delete snapshot".

The syntax of the `storageOperation` shall comply with the following definition:

```
storageOperation:
  description: >
    Type of operation to perform on the virtualised storage resource.
  type: string
  enum:
    - create-snapshot
    - delete-snapshot
  default: ""
```

8.2.6 Parameter: newSize

The parameter used when providing a resized amount of an allocated virtualised storage resource shall follow the indications provided in Table 8.2.6-1.

Table 8.2.6-1: Input data model for storageOperation

Parameter Name and Attributes	Type	Description
newSize	Number	Resized amount of allocated virtualised storage resource.

The syntax of the `newSize` shall comply with the following definition:

```
newSize:
  description: >
    Resized amount of allocated virtualised storage resource.
  type: number
  default: ""
```

8.2.7 Parameter: scopeOfAffinityOrAntiAffinityConstraintsForStorage

The parameter used when qualifying the scope of the affinity constraint shall follow the indications provided in Table 8.2.7-1.

Table 8.2.7-1: Input data model for `scopeOfAffinityOrAntiAffinityConstraintsForStorage`

Parameter Name and Attributes	Type	Description
scopeOfAffinityOrAntiAffinityConstraintsForStorage	String	If applicable. Qualifies the scope of the affinity constraint, e.g. NFVI-Node. Defaults to NFVI-Node if absent.

The syntax of the `scopeOfAffinityOrAntiAffinityConstraints` shall comply with the following definition:

```
scopeOfAffinityOrAntiAffinityConstraints:
  description: >
    Qualifies the scope of the affinity constraint,
  type: string
  enum:
    - NFVI-Node
  default: NFVI-Node
```

8.3 Parameters to be used as output

8.3.1 Parameter: nfvStorageInfo

The parameter is used when returning information for a virtualised storage resource, and its output data model shall follow the indications provided in Table 8.3.1-1. This parameter maps to the "storageResource" parameter defined in ETSI GS NFV-IFA 005 [1].

Table 8.3.1-1: Output data model for nfvStorageInfo

Parameter Name and Attributes	Type	Description
nfvStorageInfo	Object	Information of an instantiated virtualised storage resource.
>storageId	String	Identifier of the virtualised storage resource.
>storageName	String	Name of the virtualised storage resource.
>flavourId	String	Identifier of the storage flavour used to instantiate this virtual storage.
>sizeOfStorage	Number	Size of virtualised storage resource (e.g. size of volume, in GB).
>rdmaEnabled	Boolean	Indicates if the storage supports RDMA.
>ownerId	String	Identifier of the virtualised resource that owns and uses such a virtualised storage resource. The value can be NULL if the virtualised storage is not attached yet to any other resource (e.g. a virtual machine).
>zoneId	String	It identifies the resource zone where the virtual storage resources have been allocated.
>hostId	String	Identifier of the host where the virtualised storage resource is allocated.
>operationalState	String	Operational state of the resource. Allowed value: enabled, disabled.
>metadata	Array of object	List of metadata key-value pairs used by the consumer to associate meaningful metadata to the related virtualised resource. metadata is optional. It is out of scope to detail what are the sub-keys and possible values.

When used as an output parameter in a template, the syntax of the `nfvStorageInfo` shall comply with the following definition:

```
  nfvStorageInfo:
    description: >
      Information of an instantiated virtualised storage resource
    type: object
    required:
      - storageId
      - flavourId
      - typeOfStorage
      - sizeOfStorage
      - operationalState
    properties:
      storageId:
        description: >
          Identifier of the virtualised storage resource
        type: string
      storageName:
        description: >
          Name of the virtualised storage resource
        type: string
      flavourId:
        description: >
          Identifier of the storage flavour used to instantiate this virtual storage
        type: string
      typeOfStorage:
        description: >
          Type of virtualised storage resource
        type: string
      sizeOfStorage:
        description: >
          Size of virtualised storage resource
        type: number
      rdmaEnabled:
        description: >
          Indicates if the storage supports RDMA.
        type: boolean
      ownerId:
        description: >
          Identifier of the virtualised resource that owns and uses such
a virtualised storage resource. The value can be NULL if the
virtualised storage is not attached yet to any other resource
        type: string
      zoneId:
        description: >
          It identifies the resource zone where the virtual
storage resources have been allocated
        type: string
      hostId:
        description: >
          Identifier of the host where the virtualised storage resource is allocated.
        type: string
      operationalState:
        description: >
          Operational state of the resource.
        type: string
        enum:
          - enabled
          - disabled
      metadata:
        description: >
          metadata is optional. It is out of scope to detail what are the sub-keys and possible
values.
        type: array
        minItems: 0  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: object
```

Annex A (informative): Examples using OpenStack® Heat Orchestration Template

A.1 Introduction

The present Annex provides implementation examples of the data models defined for the various interfaces over the Or-Vi and Vi-Vnfm reference points using the OpenStack's Heat Orchestration Template (HOT). The purpose is to describe how the input and output parameters of the interfaces' operations can be mapped onto the HOT. In this context, an overview of the HOT template and its structure is provided, followed by selected implementation examples of interface operations using HOT templates.

A.2 Overview

A.2.1 Introduction

An OpenStack's HOT template describes the intended virtualised resource topology, the relationship between the virtualised resources to be provisioned, the type of virtualised resources and their setup in YAML text files. The template is treated as "code" by the orchestration engine while provisioning the set of virtualised resources that are declared. In addition, the template specifies input and output parameters to be exchanged with the user (e.g. the API client).

A.2.2 Template structure

The structure of a HOT is specified in HOT template guide [i.2].

A.3 Examples

A.3.1 Example#1: Allocate Virtualised Compute Resource operation

This is an example of "Create stack" in OpenStack Orchestration Service API corresponding to the Allocate Virtualised Compute Resource operation (ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]).

The input data is given as an argument and starts with `"parameters"`. Parameters grouped by `"nfv"` are specified in the present document. The input data is expressed in JSON format, as this is determined by the HOT specification.

Following the input parameter specifications in the present example (see further below), input parameters `"computeName"`, `"affinityOrAntiAffinityConstraints"`, `"computeFlavourId"`, `"vcImageId"`, `"interfaceData"`, `"metaData"`, `"locationConstraints"`, `"userData"` are given as keys with their values. This covers the input parameter values part.

The following example illustrates the input parameters that can be passed to a HEAT API call for the Allocate Virtualised Compute Resource operation.

```
{
  "parameters": {
    "nfv": {
      "computeName": "test-instance-from-stack",
      "affinityOrAntiAffinityConstraintsForCompute": {
        "type": "affinity",
        "scope": "NFVI Node",
        "affinityAntiAffinityResourceGroup": "d10312a4-9d68-4cd4-831e-fd0edf3c0649"
      },
      "computeFlavourId": 10,
```

```
      "vcImageId": "2fe6b099-f936-429b-b644-048b96b70417",
      "interfaceData": {
        "count": 3,
        "0": {
          "ipAddress": "172.17.1.15",
          "macAddress": "fa:16:3e:aa:bb:cc",
        },
        "1": {
          "ipAddress": "20.20.20.15",
          "macAddress": "fa:16:3e:ab:ca:bc",
        },
        "2": {
          "ipAddress": "30.30.30.15",
          "macAddress": "fa:16:3e:dd:ee:ff",
        }
      },
      "metaData": {
        "test-key": "test-value"
      },
      "locationConstraints": "nova",
      "userData": "test-userData"
    },
    "gap": {
      "networkId": {
        "0": "c4440f4f-66ef-4e42-9c41-7b3b449813d1",
        "1": "f6a6471d-032d-4c58-9432-c17ea1f72873",
        "2": "91652985-5fed-43f9-adae-8af0471bda03"
       }
    }
  }
}
```

Below is the corresponding example of the referred HOT that uses the parameters provided in the example above.

The "`parameters`" section in the template has definitions for input data to be provided when instantiating the template. In this case the parameter "`nfv`" is represented in the JSON format.

When input data, `"computeName": "test-instance-from-stack"`, is given, then `test-instance-from-stack` as a string value is assigned to the input parameter, `computeName`. In the same way, other values are captured and assigned to the other input parameters in the template.

The `resources` section describes what type and how virtualised resources are provisioned. In the resource handling, actual values of the input parameters are assigned to parameters used by OpenStack when performing the resource handling. For example, the line:

```
  external_id: { get_param: computeFlavourId }
```

assigns `3` (see in the example of input data above) to `external_id`.

The `outputs` section of the template describes output data for the user, e.g. when in terms of the Allocate Virtualised Compute Resource the `nfvComputeInfo` is requested. The naming and structure of output parameters in the present document and the ones used and provided by default by the OpenStack Heat Orchestration can differ. Because of this, name translation and output parameter structuring are necessary. The `template` section in the `nfvComputeInfo` resolves such a translation. For instance, a key/value pair for `computeId` is written with:

```
"computeId": "$computeId"
```

and the value of `computeId`, which is determined by the variable `$computeId`, whose value is assigned by using an intrinsic function as shown below:

```
  $computeId: { get_attr: [ virtualisedComputeResource, show, id ] }
```

The intrinsic function, `get_attr`, gets the virtualised compute identifier from the `virtualisedComputeResource`.

NOTE: An alternative to putting output parameters in the template is to use API mapping, as defined in clause 4.3.

The following is an example of a HOT for the Allocate Virtualised Compute Resource operation.

```
heat_template_version: pike
description: Allocate Virtualised Compute Resource operation

parameters:
  nfv:
    type: json
    description:
    default: ""

  gap:
    type: json
    description:
    default: ""

conditions:
  Constraints_ResourceList_is_null: { equals: [ { get_param: [ nfv,
affinityOrAntiAffinityConstraintsForCompute, affinityAntiAffinityResourceList ] }, "" ] }
  Constraints_type_is_affinity: { equals : [ { get_param: [ nfv,
affinityOrAntiAffinityConstraintsForCompute, type ] }, "affinity" ] }

resources:
  interfaceResource:
    type: OS::Heat::ResourceGroup
    properties:
      count: { get_param: [ nfv, interfaceData, count ] }
      resource_def:
        type: http://controller/Allocate-Virtualised-Compute-Resource-operation/createPort.yaml
        properties:
          interfaceData: { get_param: [ nfv, interfaceData ] }
          networkId: { get_param: [ gap, networkId ] }
          index: "%index%"

  forOutput-flavorResource:
    type: OS::Nova::Flavor
    external_id: { get_param: [ nfv, computeFlavourId ] }

  forOutput-flavorExtraSpecs:
    type: OS::Heat::ResourceGroup
    depends_on: [ forOutput-flavorResource ]
    properties:
      count: 1
      resource_def:
        type: http://controller/Allocate-Virtualised-Compute-Resource-
operation/getFlavorExtraSpecs.yaml
        properties:
          policy: { get_attr: [ forOutput-flavorResource, extra_specs, "hw:cpu_policy" ] }
          cores: { get_attr: [ forOutput-flavorResource, extra_specs, "hw:cpu_cores" ] }
          sockets: { get_attr: [ forOutput-flavorResource, extra_specs, "hw:cpu_sockets" ] }
          threads: { get_attr: [ forOutput-flavorResource, extra_specs, "hw:cpu_threads" ] }

  forOutput-portStatusResource:
    type: OS::Heat::ResourceGroup
    depends_on: [ virtualisedComputeResource ]
    properties:
      count: { get_param: [ nfv, interfaceData, count ] }
      resource_def:
        type: http://controller/Allocate-Virtualised-Compute-Resource-operation/getPortStatus.yaml
        properties:
          status: { get_attr: [ interfaceResource, show, status ] }
          index: "%index%"

  forOutput-virtualisedComputeResourceStatus:
    type: OS::Heat::ResourceGroup
    depends_on: [ virtualisedComputeResource ]
    properties:
      count: 1
      resource_def:
        type: http://controller/Allocate-Virtualised-Compute-Resource-
operation/getComputeResourceStatus.yaml
        properties:
          status: { get_attr: [ virtualisedComputeResource, show, status ] }

  virtualisedComputeResource:
    type: OS::Nova::Server
    depends_on: [ interfaceResource ]
    properties:
```

```
      name: { get_param: [ nfv, computeName ] }
      scheduler_hints:
        if:
          - Constraints_ResourceList_is_null
          - group: { get_param: [ nfv, affinityOrAntiAffinityConstraintsForCompute,
affinityAntiAffinityResourceGroup ] }
          - if:
            - Constraints_type_is_affinity
            - same_host:
                repeat:
                  for_each:
                    <%Resource%>: { get_param: [ nfv, affinityOrAntiAffinityConstraintsForCompute,
affinityAntiAffinityResourceList ] }
                  template:
                    <%Resource%>
            - different_host:
                repeat:
                  for_each:
                    <%Resource%>: { get_param: [ nfv, affinityOrAntiAffinityConstraintsForCompute,
affinityAntiAffinityResourceList ] }
                  template:
                    <%Resource%>
      flavor: { get_param: [ nfv, computeFlavourId ] }
      image: { get_param: [ nfv, vcImageId ] }
      networks:
        repeat:
          for_each:
            <%Port%>: { get_attr: [ interfaceResource, id ] }
          template:
            port: <%Port%>
      metadata: { get_param: [ nfv, metaData ] }
      availability_zone: { get_param: [ nfv, locationConstraints ] }
      user_data: { get_param: [ nfv, userData ] }
      user_data_format: "RAW"

outputs:
  nfvComputeInfo:
    value:
      str_replace:
        template: |
          {
              "computeId": "$computeId",
              "computeName": "$computeName",
              "flavourId": "$flavourId",
              "virtualCpu": {
                "numVirtualCpu": "$numVirtualCpu",
                "virtualCpuPinning": "$virtualCpuPinning"
              },
              "virtualMemory": {
                "virtualMemSize": "$virtualMemSize"
              },
              "virtualNetworkInterface": "$virtualNetworkInterface",
              "virtualDisks": [
                {
                  "storageId": "$storageId",
                  "typeOfStorage": "disk",
                  "sizeOfStorage": "$sizeOfStorageDisk",
                  "operationalState": "$storageOperationalState"
                },
                {
                  "storageId": "$storageId",
                  "typeOfStorage": "ephemeral",
                  "sizeOfStorage": "$sizeOfStorageEphemeral",
                  "operationalState": "$storageOperationalState"
                },
                {
                  "storageId": "$storageId",
                  "typeOfStorage": "swap",
                  "sizeOfStorage": "$sizeOfStorageSwap",
                  "operationalState": "$storageOperationalState"
                }
              ],
              "vcImageId": "$vcImageId",
              "zoneId": "$zoneId",
              "hostId": "$hostId",
              "operationalState": "$operationalState",
              "metaData": "$metadata"
          }
```

```
        params:
          $computeId: { get_attr: [ virtualisedComputeResource, show, id ] }
          $computeName: { get_attr: [ virtualisedComputeResource, show, name ] }
          $flavourId: { get_attr: [ virtualisedComputeResource, show, flavor, id ] }
          $numVirtualCpu: { get_attr: [ forOutput-flavorResource, show, vcpus ] }
          $virtualCpuPinning: {get_attr: [ forOutput-flavorExtraSpecs, resource.0.cpuPinning ] }
          $virtualMemSize: { get_attr: [ forOutput-flavorResource, show, ram ] }
          $virtualNetworkInterface:
            repeat:
              for_each:
                <%resourceId%>: { get_attr: [ interfaceResource, show, id ] }
                <%ownerId%>: { get_attr: [ interfaceResource, show, device_id ] }
                <%networkId%>: { get_attr: [ interfaceResource, show, network_id ] }
                <%ipAddress%>: { get_attr: [ interfaceResource, show, fixed_ips, 0, ip_address ] }
                <%typeVirtualNic%>: { get_attr: [ interfaceResource, show, "binding:vnic_type" ] }
                <%macAddress%>: { get_attr: [ interfaceResource, show, mac_address ] }
                <%operationalState%>: { get_attr: [ forOutput-portStatusResource, status ] }
              template:
                "resourceId": <%resourceId%>
                "ownerId": <%ownerId%>
                "networkId": <%networkId%>
                "ipAddress": <%ipAddress%>
                "typeVirtualNic": <%typeVirtualNic%>
                "macAddress": <%macAddress%>
                "operationalState": <%operationalState%>
              permutations: false
          $storageId: { get_attr: [ virtualisedComputeResource, show, id ] }
          $sizeOfStorageDisk: { get_attr: [ forOutput-flavorResource, show, disk ] }
          $sizeOfStorageEphemeral: { get_attr: [ forOutput-flavorResource, show, "OS-FLV-EXT-
DATA:ephemeral" ] }
          $sizeOfStorageSwap:
            yaql:
              expression: $.data.swap_MB/1024
              data:
                swap_MB: { get_attr: [ forOutput-flavorResource, show, swap ] }
          $storageOperationalState: { get_attr: [ forOutput-virtualisedComputeResourceStatus,
resource.0.status ] }
          $vcImageId: { get_attr: [ virtualisedComputeResource, show, image, id ] }
          $zoneId: { get_attr: [ virtualisedComputeResource, show, "OS-EXT-AZ:availability_zone" ] }
          $hostId: { get_attr: [ virtualisedComputeResource, show, "OS-EXT-SRV-ATTR:host" ] }
          $operationalState: { get_attr: [ forOutput-virtualisedComputeResourceStatus,
resource.0.status ] }
          $metadata: { get_attr: [ virtualisedComputeResource, show, metadata ] }
```

Below is an output example using template output parameters related to the allocated compute resource as provided by "Show output" in OpenStack Orchestration Service API [i.2]. Attributes defined in "`nfvComputeInfo`" of the HOT can be seen in the body of "`output_value`". `virtualDisks` is prepared in the ephemeral storage in the hypervisor of the host compute node in this sample. Thus, the value of `storageId` is equal to the `computeId`. The storage resource is managed in the hypervisor of the host compute node.

```
{
  "output":
    "output_value":
    {
        "computeId": "735ee7f9-92ce-4c00-8c7d-63eeeda75368",
        "computeName": "test-instance-from-stack",
        "flavourId": "10",
        "virtualCpu":
        {
          "numVirtualCpu": "1",
          "virtualCpuPinning":
          {
            "cpuPinningPolicy": "dynamic"
          }
        },
        "virtualMemory":
        {
          "virtualMemSize": "64"
        },
        "virtualNetworkInterface":
        [
          {
            "ipAddress": "172.17.1.15",
            "macAddress": "fa:16:3e:aa:bb:cc",
            "networkId": "c4440f4f-66ef-4e42-9c41-7b3b449813d1",
            "operationalState": "disabled",
            "ownerId": "735ee7f9-92ce-4c00-8c7d-63eeeda75368",
```

```
            "resourceId": "658fdc3b-679e-49bd-9dbb-9d19a60f0321",
            "typeVirtualNic": "normal"
          },
          {
            "ipAddress": "20.20.20.15",
            "macAddress": "fa:16:3e:ab:ca:bc",
            "networkId": "f6a6471d-032d-4c58-9432-c17ea1f72873",
            "operationalState": "disabled",
            "ownerId": "735ee7f9-92ce-4c00-8c7d-63eeeda75368",
            "resourceId": "5b090b65-81bc-43a8-abbb-b306d1ac87c1",
            "typeVirtualNic": "normal"},
            {
              "ipAddress": "30.30.30.15",
              "macAddress": "fa:16:3e:dd:ee:ff",
              "networkId": "91652985-5fed-43f9-adae-8af0471bda03",
              "operationalState": "disabled",
              "ownerId": "735ee7f9-92ce-4c00-8c7d-63eeeda75368",
              "resourceId": "e840e0b4-4cf0-4e14-82c5-481e086abd4a",
              "typeVirtualNic": "normal"
            }
          ],
          "virtualDisks":
          [
            {
              "storageId": "735ee7f9-92ce-4c00-8c7d-63eeeda75368",
              "typeOfStorage": "disk",
              "sizeOfStorage": "10",
              "operationalState": "enable"
            },
            {
              "storageId": "735ee7f9-92ce-4c00-8c7d-63eeeda75368",
              "typeOfStorage": "ephemeral",
              "sizeOfStorage": "5",
              "operationalState": "enable"
            },
            {
              "storageId": "735ee7f9-92ce-4c00-8c7d-63eeeda75368",
              "typeOfStorage": "swap",
              "sizeOfStorage": "1",
              "operationalState": "enable"
            }
          ],
          "vcImageId": "2fe6b099-f936-429b-b644-048b96b70417",
          "zoneId": "nova",
          "hostId": "compute3",
          "operationalState": "enable",
          "metaData":
          {
            "test-key": "test-value"
          }
    },
    "output_key": "nfvComputeInfo",
    "description": "No description given"
  }
}
```


A.3.2 Example#2 Allocate Virtualised Network Resource operation

This is an example of "Create stack" in OpenStack Orchestration Service API [i.2] corresponding to the Allocate Virtualised Network Resource operation (ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]).

The input data is given as an argument and starts with `"parameters"`. The input data is expressed in JSON format, as this is determined by the HOT specification.

Parameters grouped by `"nfv"` are specified in the present document. Following the input parameter specifications in the present example (see further below), input parameters, `networkResourceName`, `networkResourceType`, `typeNetworkData`, are given as keys with their values. This covers the input parameter values part.

The following example illustrates the input parameters that can be passed to a HEAT API call for the Allocate Virtualised Network Resource operation.

```
{
    "parameters": {
        "nfv": {
            "networkResourceName": "test-network-from-stack",
            "networkResourceType": "network",
            "typeNetworkData": {
                "bandwidth": 100,
                "networkType": "flat",
                "isShared": false,
                "layer3Attributes": {
                    "ipVersion": "IPv4",
                    "gatewayIp": "10.0.0.1",
                    "cidr": "10.0.0.0/24",
                    "isDhcpEnabled": false,
                    "addressPool": [
                        {
                            "start": "10.0.0.101",
                            "end": "10.0.0.110"
                        }
                    ]
                }
            }
        }
    }
}
```

NOTE 1: `resourceGroupId` is not covered in the present example. ETSI GS NFV-IFA 005 [1], and the ETSI GS NFV-IFA 006 [2] do not specify the required operations for the management of resource groups for infrastructure tenants (e.g. creation of a resource group, etc.).

NOTE 2: The example is prepared to highlight only an allocation of virtualised network resource. The parameters of affinityOrAntiAffinityConstraintsForNetwork and locationConstraintForNetwork are not used in this example (cardinality is 0).

Below is the corresponding example of the HOT that uses the parameters provided in the example above.

The `parameters` section in the template has definitions for input data to be provided when instantiating the template. In this case, parameter, `nfv` is typed as JSON format.

When input data `"networkResourceName": "test-network-from-stack"`, is given, then `test-network-from-stack` as a string value is assigned to the input parameter `networkResourceName`. In the same way, other values are captured and assigned to the other input parameters in the template.

In this use case, `networkType` is `flat`; the cardinality of `segmentType` can be "0" to allow for the flat networks without any specific segmentation.

The `resources` section describes what type and how virtualised resources are provisioned. In the resource handling, actual values of the input parameters are assigned to parameters used by OpenStack when performing the resource handling. For example, the line:

```
name: { get_param: networkResourceName }
```

gets the value, `test-network-from-stack` (see in the example of input data above), of the input parameter, `networkResourceName`, and then assigns `test-network-from-stack` to `name`.

The `outputs` section of the template describes output data for the user, e.g. when in terms of the Allocate Virtualised Network Resource the `networkResource` is requested. The naming and structure of output parameter in the present document and the ones used and provided by default by the OpenStack Heat Orchestration can differ. Because of this, name translation and output parameter structuring are necessary.

The `template` section in the `nfvNetworkInfo` resolves such a translation. For instance, a key/value pair for `networkResourceId` is written with:

```
"networkResourceId": "$networkResourceId"
```

and the value of `networkResourceId`, which is determined by the variable `$networkResourceId`, whose value is assigned by using an intrinsic function as shown below:

```
  $networkResourceId: { get_attr: [ networkResource, resource.0.show, id ] }
```

The intrinsic function `get_attr` gets the virtualised network identifier from the `networkResource`.

NOTE 3: An alternative to putting output parameters in the template is to use API mapping, as defined in clause 4.3.

The following is an example of a HOT for the Allocate Virtualised Network Resource operation.

```
heat_template_version: pike
description: Allocate Virtualised Network Resource operation.

parameters:
  nfv:
    type: json
    description:
    default: ""

conditions:
  networkResourceType_is_network: { equals: [ { get_param: [ nfv, networkResourceType ] }, "network"
] }
  networkResourceType_is_subnet: { equals: [ { get_param: [ nfv, networkResourceType ] }, "subnet" ]
}
  networkResourceType_is_network-port: { equals: [ { get_param: [ nfv, networkResourceType ] },
"network-port" ] }
  layer3Attributes_is_null: { equals: [ { get_param: [ nfv, typeNetworkData, layer3Attributes ] },
"" ] }
  layer3Attributes_ipVersion_is_IPv4: { equals: [ { get_param: [ nfv, typeNetworkData,
layer3Attributes, ipVersion ] }, "IPv4" ] }
  typeSubnetData_ipVersion_is_IPv4: { equals: [ { get_param: [ nfv, typeSubnetData, ipVersion ] },
"IPv4" ] }

resources:
  forOutput-networkStatusResource:
    type: OS::Heat::ResourceGroup
    depends_on: networkResource
    properties:
      count: 1
      resource_def:
        type: http://controller/Allocate-Virtualised-Network-Resource-
operation/getResourceStatus.yaml
        properties:
          status: { get_attr: [ networkResource, resource.0.show, status ] }

  forOutput-networkPortStatusResource:
    type: OS::Heat::ResourceGroup
    depends_on: networkPortResource
    properties:
      count: 1
      resource_def:
        type: http://controller/Allocate-Virtualised-Network-Resource-
operation/getResourceStatus.yaml
        properties:
          status: { get_attr: [ networkPortResource, resource.0.show, status ] }

  portTypeDecidedByExistingNetwork:
    type: OS::Heat::ResourceGroup
    properties:
      count: 1
      resource_def:
        if:
          - networkResourceType_is_network-port
          - type: http://controller/Allocate-Virtualised-Network-Resource-
operation/getPortTypeDecidedByExistingNetwork.yaml
            properties:
              network_id: { get_param: [ nfv, typeNetworkPortData, networkId ] }
          - type: OS::Heat::None
```

```
  networkResource:
    type: OS::Heat::ResourceGroup
    properties:
      count: 1
      resource_def:
        if:
          - networkResourceType_is_network
          - type: OS::Neutron::ProviderNet
            properties:
              name: { get_param: [ nfv, networkResourceName ] }
              network_type: { get_param: [ nfv, typeNetworkData, networkType ] }
              shared: { get_param: [ nfv, typeNetworkData, isShared ] }
              physical_network: "provider_network"
          - type: OS::Heat::None

  subnetOfNewNetworkResource:
    type: OS::Heat::ResourceGroup
    depends_on: networkResource
    properties:
      count: 1
      resource_def:
        if:
          - layer3Attributes_is_null
          - type: OS::Heat::None
          - type: OS::Neutron::Subnet
            properties:
              network: { get_attr: [ networkResource, refs, 0 ] }
              ip_version: { if: [ layer3Attributes_ipVersion_is_IPv4, 4, 6 ] }
              gateway_ip: { get_param: [ nfv, typeNetworkData, layer3Attributes, gatewayIp ] }
              cidr: { get_param: [ nfv, typeNetworkData, layer3Attributes, cidr ] }
              enable_dhcp: { get_param: [ nfv, typeNetworkData, layer3Attributes, isDhcpEnabled ] }
              allocation_pools: { get_param: [ nfv, typeNetworkData, layer3Attributes, addressPool ]
}

  subnetResource:
    type: OS::Heat::ResourceGroup
    properties:
      count: 1
      resource_def:
        if:
          - networkResourceType_is_subnet
          - type: OS::Neutron::Subnet
            properties:
              name: { get_param: [ nfv, networkResourceName ] }
              network: { get_param: [ nfv, typeSubnetData, networkId ] }
              ip_version: { if: [ typeSubnetData_ipVersion_is_IPv4, 4, 6 ] }
              gateway_ip: { get_param: [ nfv, typeSubnetData, gatewayIp ] }
              cidr: { get_param: [ nfv, typeSubnetData, cidr ] }
              enable_dhcp: { get_param: [ nfv, typeSubnetData, isDhcpEnabled ] }
              allocation_pools: { get_param: [ nfv, typeSubnetData, addressPool ] }
          - type: OS::Heat::None

  networkPortResource:
    type: OS::Heat::ResourceGroup
    depends_on: portTypeDecidedByExistingNetwork
    properties:
      count: 1
      resource_def:
        if:
          - networkResourceType_is_network-port
          - type: OS::Neutron::Port
            properties:
              name: { get_param: [ nfv, networkResourceName ] }
              network: { get_param: [ nfv, typeNetworkPortData, networkId ] }
              binding:vnic_type: { get_attr: [ portTypeDecidedByExistingNetwork, resource.0.type ] }
          - type: OS::Heat::None

outputs:
  nfvNetworkInfo:
   value:
     if:
       - networkResourceType_is_network
       - str_replace:
           template: |
             {
                 "networkResourceId": "$networkResourceId",
                 "networkResourceName": "$networkResourceName",
                 "subnet": {
```

```
                  "resourceId": "$resourceId",
                  "networkId": "$networkId",
                  "ipVersion": "IPv$ipVersion",
                  "gatewayIp": "$gatewayIp",
                  "cidr": "$cidr",
                  "isDhcpEnabled": "$isDhcpEnabled",
                  "addressPool": "$addressPool"
                },
                "networkType": "$networkType",
                "isShared": "$isShared",
                "zoneId": "$zoneId",
                "operationalState": "$operationalState"
            }
          params:
            $networkResourceId: { get_attr: [ networkResource, resource.0.show, id ] }
            $networkResourceName: { get_attr: [ networkResource, resource.0.show, name ] }
            $resourceId: { get_attr: [ subnetOfNewNetworkResource, resource.0.show, id ] }
            $networkId: { get_attr: [ subnetOfNewNetworkResource, resource.0.show, network_id ] }
            $ipVersion: { get_attr: [ subnetOfNewNetworkResource, resource.0.show, ip_version ] }
            $gatewayIp: { get_attr: [ subnetOfNewNetworkResource, resource.0.show, gateway_ip ] }
            $cidr: { get_attr: [ subnetOfNewNetworkResource, resource.0.show, cidr ] }
            $isDhcpEnabled: { get_attr: [ subnetOfNewNetworkResource, resource.0.show, enable_dhcp
] }
            $addressPool: { get_attr: [ subnetOfNewNetworkResource, resource.0.show,
allocation_pools ] }
            $networkType: { get_attr: [ networkResource, resource.0.show, "provider:network_type" ]
}
            $isShared: { get_attr: [ networkResource, resource.0.show, shared ] }
            $zoneId: { list_concat: { get_attr: [ networkResource, show, availability_zones ] } }
            $operationalState: { get_attr: [ forOutput-networkStatusResource, resource.0.status ] }
      - str_replace:
          template:
            This is output of Network.
            Please specify output of $resourceType.
          params:
            $resourceType: { get_param: [ nfv, networkResourceType ] }

  nfvSubnetInfo:
    value:
      if:
       - networkResourceType_is_subnet
       - str_replace:
           template: |
             {
                 "resourceId": "$resourceId",
                 "networkId": "$networkId",
                 "ipVersion": "IPv$ipVersion",
                 "gatewayIp": "$gatewayIp",
                 "cidr": "$cidr",
                 "isDhcpEnabled": "$isDhcpEnabled",
                 "addressPool": "$addressPool"
               }
           params:
             $resourceId: { get_attr: [ subnetResource, resource.0.show, id ] }
             $networkId: { get_attr: [ subnetResource, resource.0.show, network_id ] }
             $ipVersion: { get_attr: [ subnetResource, resource.0.show, ip_version ] }
             $gatewayIp: { get_attr: [ subnetResource, resource.0.show, gateway_ip ] }
             $cidr: { get_attr: [ subnetResource, resource.0.show, cidr ] }
             $isDhcpEnabled: { get_attr: [ subnetResource, resource.0.show, enable_dhcp ] }
             $addressPool: { get_attr: [ subnetResource, resource.0.show, allocation_pools ] }
       - str_replace:
           template:
             This is output of Subnet.
             Please specify output of $resourceType.
           params:
             $resourceType: { get_param: [ nfv, networkResourceType ] }

  nfvNetworkPortInfo:
    value:
      if:
       - networkResourceType_is_network-port
       - str_replace:
           template: |
             {
                 "resourceId": "$resourceId",
                 "networkId": "$networkId",
                 "attachedResourceId": "$attachedResourceId",
                 "operationalState": "$portOperationalState"
```

```
            }
          params:
            $resourceId: { get_attr: [ networkPortResource, resource.0.show, id ] }
            $networkId: { get_attr: [ networkPortResource, resource.0.show, network_id ] }
            $attachedResourceId: { get_attr: [ networkPortResource, resource.0.show, id ] }
            $portOperationalState: { get_attr: [ forOutput-networkPortStatusResource,
resource.0.status ] }
      - str_replace:
          template:
            This is output of NetworkPort.
            Please specify output of $resourceType.
          params:
            $resourceType: { get_param: [ nfv, networkResourceType ] }
```

Below is an output example using template output parameters related to the allocated network resource as provided by "Show output" in OpenStack Orchestration Service API [i.2]. Attributes defined in `nfvNetworkInfo` of the HOT can be seen in the body of `output_value.`

Parameters grouped by `"nfv"` are specified in the present document.

```
{
    "output": {
        "output_value": {
                "networkResourceId": "8b6fbb50-9382-40fc-9adf-1e1ed3a6e6b0",
                "networkResourceName": "test-network-from-stack",
                "subnet": {
                    "resourceId": "bbfee0fb-e28e-465b-8982-5aaaa0ef5afe",
                    "networkId": "8b6fbb50-9382-40fc-9adf-1e1ed3a6e6b0",
                    "ipVersion": "IPv4",
                    "gatewayIp": "10.0.0.1",
                    "cidr": "10.0.0.0/24",
                    "isDhcpEnabled": false,
                    "addressPool": [
                        {
                            "end": "10.0.0.110",
                            "start": "10.0.0.101"
                        }
                    ]
                },
                "networkType": "flat",
                "isShared": false,
                "zoneId": [
                    "nova"
                ],
                "operationalState": "enable"
            },
            "output_key": "nfvNetworkInfo",
            "description": "No description given"
    }
}
```

A.3.3 Example#3: Allocate Virtualised Storage Resource operation

This is an input parameter example of "Create stack" in OpenStack Orchestration Service API [i.2] corresponding to the Allocate Virtualised Storage Resource operation (ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]).

The input data is given as an argument and starts with `"parameters"`. The input data is expressed in JSON format, as this is determined by the HOT specification.

Parameters grouped by `"nfv"` are specified in the present document. Following the input parameter specifications in the present example (see further below), input parameters `"storageName"`, `"affinityOrAntiAffinityConstraintsForStorage"`, `"storageData"`, `"locationConstraints"`, `" metaData"`, `"stack_name"` are given as keys with their values. This covers the input parameter values part.

The following example illustrates the input parameters that can be passed to a HEAT API call for the Allocate Virtualised Storage Resource operation.

```
{
  "parameters":
  {
    "nfv":
    {
      "storageName": "test-volume-from-stack",
      "affinityOrAntiAffinityConstraintsForStorage":
      {
        "typeOfAffinityOrAntiAffinityConstraintForStorage": "affinity",
        "scopeOfAffinityOrAntiAffinityConstraintForStorage": "NFVI-Node",
        "affinityAntiAffinityResourceList":
        {
          "resource": [
            "4fc36790-ce40-439e-bc72-05a821a59b2b",
            "3f26f7ac-df5c-43bb-bace-87d6ec7b5374"
            ]
        }
      },
      "storageData":
      {
        "storageAttributes":
        {
          "typeOfStorage": "volume",
          "sizeOfStorage": 1
        }
      },
      "locationConstraints": "nova",
      "metaData":
      {
        "test-key": "test-value"
      }
    }
  }
}
```

Below is the corresponding example of the HOT that uses the parameters provided in the example above.

The `parameters` section in the template has definitions for input data to be provided when instantiating the template. In this case, parameter, `nfv`, is typed as JSON format.

When the input data `"storageName": "test-volume-from-stack"` is given, then `test-volume-from-stack` as a string value is assigned to the input parameter `storageName`. In the same way, other values are captured and assigned to the other input parameters in the template.

JSON format is used for the structured data (e.g. `affinityOrAntiAffinityConstraintsForStorage`, `storageData`, `metaData`) as determined by the HOT specification. The input data is accepted as a JSON data and then used in the resource handlings written in the resource section.

When the input attribute data `"typeOfAffinityOrAntiAffinityConstraintForStorage": "affinity"` is given, and the condition `Constraints_type_is_affinity` becomes true, then `same_host` of `scheduler_hints` is selected in the if_clause. Finally, the virtualised storage resource is instantiated on an NFVI-Node, which hosts the resources specified by `affinityAntiAffinityResourceList`.

The affinity/anti-affinity attributes, e.g. `scopeOfAffinityOrAntiAffinityConstraintForStorage, affinityAntiAffinityResoruceGroup,` are passed because those parameters are specified in the present document, but are not defined in OpenStack HEAT specification. Those attribute values are therefore not used in HEAT operations, however, are used by the logic of the example template. The parameter `"flavourId"` is specified in the present document but it is not specified and used in OpenStack HEAT; it is therefore omitted from the example.

The `resources` section describes what type and how virtualised resources are provisioned. In the resource handling, actual values of the input parameters are assigned to parameters used by OpenStack when performing the resource handling. For example, the line:

```
  name: { get_param: storageName }
```

gets the value `test-volume-from-stack` (see in the example of input data above), of the input parameter `storageName`, and then assigns `test-volume-from-stack` to `name`.

The `outputs` section of the template describes output data for the user, e.g. when in terms of the Allocate Virtualised Storage Resource the `nfvStorageInfo` is requested. The naming and structure of output parameters in the present document and the ones used and provided by default by the OpenStack Heat Orchestration may differ. Because of this, name translation and output parameter structuring are necessary.

The `template` section in the `nfvStorageInfo` resolves such a translation. For instance, a key/value pair for `storageId` is written with:

```
"storageId": "$storageId"
```

and the value of `storageId`, which is determined by the variable `$storageId`, whose value is assigned by using an intrinsic function as shown below:

```
  $storageId: { get_attr: [ virtualisedStorageResource, resource.0.show, id ] }
```

The intrinsic function `get_attr` gets the virtualised storage identifier from the `virtualisedStorageResource`.

NOTE 1: An alternative to putting output parameters in the template is to use API mapping, as defined in clause 4.3.

The following is an example of a HOT for the Allocate Virtualised Storage Resource operation.

```
heat_template_version: pike
description: Allocate Virtualised Storage Resource operation

parameters:
  nfv:
    type: json
    description:
    default: {}

conditions:
  typeOfStorage_if_volume: { equals : [ { get_param: [ nfv, storageData, storageAttributes,
typeOfStorage ] }, "volume" ] }
  Constraints_type_is_affinity: { equals : [ { get_param: [ nfv,
affinityOrAntiAffinityConstraintsForStorage, typeOfAffinityOrAntiAffinityConstraintForStorage ] },
"affinity" ] }

resources:
  virtualisedStorageResource:
    type: OS::Heat::ResourceGroup
    properties:
      count: 1
      resource_def:
        if:
          - typeOfStorage_if_volume
          - type: OS::Cinder::Volume
            properties:
              name: { get_param: [ nfv, storageName ] }
              scheduler_hints:
                if:
                  - Constraints_type_is_affinity
                  - same_host:
                      repeat:
                        for_each:
                          <%Resource%>: { get_param: [ nfv,
affinityOrAntiAffinityConstraintsForStorage, affinityAntiAffinityResourceList, resource ] }
                        template:
                          <%Resource%>
                  - different_host:
                      repeat:
                        for_each:
                          <%Resource%>: { get_param: [ nfv,
affinityOrAntiAffinityConstraintsForStorage, affinityAntiAffinityResourceList, resource ] }
                        template:
                          <%Resource%>
              size: { get_param: [ nfv, storageData, storageAttributes, sizeOfStorage ] }
              availability_zone: { get_param: [ nfv, locationConstraints ] }
              metadata: { get_param: [ nfv, metaData ] }
          - type: OS::Heat::None

  forOutput-virtualisedStorageResourceStatus:
    type: OS::Heat::ResourceGroup
```

```
    depends_on: virtualisedStorageResource
    properties:
      count: 1
      resource_def:
        type: http://controller/Allocate-Virtualised-Storage-Resource-
operation/getVirtualisedStorageResourceStatus.yaml
        properties:
          status: { get_attr: [ virtualisedStorageResource, resource.0.show, status ] }

outputs:
  nfvStorageInfo:
    value:
      str_replace:
        template: |
          {
              "storageId": "$storageId",
              "storageName": "$storageName",
              "typeOfStorage": "$typeOfStorage",
              "sizeOfStorage": "$sizeOfStorage",
              "ownerId": "$ownerId",
              "zoneId": "$zoneId",
              "hostId": "$hostId",
              "operationalState": "$operationalState",
              "metadata": "$metadata"
          }
        params:
          $storageId: { get_attr: [ virtualisedStorageResource, resource.0.show, id ] }
          $storageName: { get_attr: [ virtualisedStorageResource, resource.0.show, name ] }
          $typeOfStorage: { get_param: [ nfv, storageData, storageAttributes, typeOfStorage ] }
          $sizeOfStorage: { get_attr: [ virtualisedStorageResource, resource.0.show, size ] }
          $ownerId: { get_attr: [ virtualisedStorageResource, resource.0.show, attachments, 0,
server_id ] }
          $zoneId: { get_attr: [ virtualisedStorageResource, resource.0.show, availability_zone ] }
          $hostId: { get_attr: [ virtualisedStorageResource, resource.0.show, "os-vol-host-
attr:host" ] }
          $operationalState: { get_attr: [ forOutput-virtualisedStorageResourceStatus,
resource.0.status ] }
          $metadata: { get_attr: [ virtualisedStorageResource, resource.0.show, metadata ] }
```

Below is an output example of the output parameters related to the allocated storage resource as provided by "Show output" in OpenStack Orchestration Service API [i.2]. Specified attributes defined in `nfvStorageInfo` of the HOT can be seen in the body of `output_value.`

NOTE 2: An alternative to model output parameters is API mapping, as defined in clause 4.3.

Parameters grouped by `"nfv"` are specified in the present document.

```
{
  "output":
  {
    "output_value":
    {
        "storageId": "f4e0afb5-1165-46a4-95e7-2405bfed9b5e",
        "storageName": "test-volume-from-stack",
        "typeOfStorage": "volume",
        "sizeOfStorage": 1,
        "ownerId": "",
        "zoneId": "nova",
        "hostId": "compute1@lvm#LVM",
        "operationalState": "enable",
        "metadata":
        {
          "test-key": "test-value"
        }
    },
    "output_key": "nfvStorageInfo",
    "description": "No description given"
  }
}
```

A.3.4 Example#4: Create Compute Flavour operation

This is an example of "Create stack" in OpenStack Orchestration Service API [i.2] corresponding to the Create Compute Flavour operation (ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2]).

The input data is given as an argument and starts with `"parameters"`. The input data is expressed in JSON format, as this is determined by the HOT specification.

Parameters grouped by `"nfv"` are specified in the present document. Following the input parameter specifications in the present example (see further below), input parameters `flavourId`, `accelerationCapabilityForVirtualComputeFlavor`, `virtualMemory`, `virtualCpu`, `virtualNetworkInterface` are given as keys with their corresponding values. This example covers the input parameter values part.

The following example illustrates the input parameters that can be passed to a HEAT API call for the Compute Flavour operation.

```
{
  "parameters": {
    "nfv": {
      "flavourId": 10,
      "accelerationCapabilityForVirtualComputeFlavor": "gpu",
      "virtualMemory": {
        "virtualMemSize": 100
      },
      "virtualCpu": {
        "numVirtualCpu": 8,
        "virtualCpuPinning": {
          "cpuPinningPolicy": "static",
          "cpuPinningRules": {
            "cores": 2,
            "sockets": 2,
            "threads": 2
          }
          }
      },
      "storageAttributes": [
        {
          "typeOfStorage": "disk",
          "sizeOfStorage": 10
        },
        {
          "typeOfStorage": "ephemeral",
          "sizeOfStorage": 20
        },
        {
          "typeOfStorage": "swap",
          "sizeOfStorage": 30
        }
      ],
      "virtualNetworkInterface": {
        "accelerationCapabilityForVirtualNetworkInterface": "dpdk"
      }
    }
  }
}
```

Below is the corresponding example of the HOT that uses the parameters provided in the example above.

The `parameters` section in the template has definitions for input data to be provided when instantiating the template. In this case, parameter, `nfv` is typed as JSON format.

By the function `get_param`, the value of the input parameters are taken from input parameters and paired with resource keys in the property section; `flavorid` is paired with 10 of `flavourId`, `ram` is paired with 100 of `virtualMemSize` and `vcpus` is paired with 8 of `numVirtualCpu`.

The value of `disk` is determined by the result of the if-clauses. The status of `typeOfStorage_0_is_disk`, `typeOfStorage_1_is_disk`, `typeOfStorage_2_is_disk` is determined by the comparison with each element of `typeOfStorage` in the input parameters. In this example, `typeOfStorage_0_is_disk` becomes true because the value of the element `storageAttributes[0].typeOfStorage` is equal to `"disk"`. So the value of `storageAttributes[0].sizeOfStorage` is taken and `disk` is paired with 10.

In the same way, `ephemeral` and `swap` are paired with 20 and 30 respectively.

`extra_specs` are key/value pairs in OpenStack. In this use case, input parameters:

```
"cpuPinningPolicy": "static"
"accelerationCapabilityForVirtualNetworkInterface": "dpdk"
"accelerationCapabilityForVirtualComputeFlavor": "gpu"
```

are given, so the value of `cores`, `sockets`, `threads` are paired as follows:

```
"hw:cpu_sockets": 2
"hw:cpu_cores": 2
"hw:cpu_threads": 2
```

More `extra_specs` are specified but those are not defined in the present document.

More resource handling (e.g. crypto, RDMA, packet dispatch, TCP Chimney, dynamic) can be added but this example does not propose to cover all.

```
heat_template_version: pike
description: Create Compute Flavour operation.

parameters:
  nfv:
    type: json
    description:
    default: {}

conditions:
  cpuPinningPolicy_is_static: { equals: [ { get_param: [ nfv, virtualCpu, virtualCpuPinning,
cpuPinningPolicy ] }, "static" ] }
  accelerationCapabilityForVirtualNetworkInterface_is_dpdk: { equals: [ { get_param: [ nfv,
virtualNetworkInterface, accelerationCapabilityForVirtualNetworkInterface ] }, "dpdk"] }
  accelerationCapabilityForVirtualComputeFlavor_is_gpu: { equals: [ { get_param: [ nfv,
accelerationCapabilityForVirtualComputeFlavor ] }, "gpu" ] }
  typeOfStorage_0_is_disk: { equals: [ { get_param: [ nfv, storageAttributes, 0, typeOfStorage ] },
"disk" ] }
  typeOfStorage_1_is_disk: { equals: [ { get_param: [ nfv, storageAttributes, 1, typeOfStorage ] },
"disk" ] }
  typeOfStorage_2_is_disk: { equals: [ { get_param: [ nfv, storageAttributes, 2, typeOfStorage ] },
"disk" ] }
  typeOfStorage_0_is_ephemeral: { equals: [ { get_param: [ nfv, storageAttributes, 0, typeOfStorage
] }, "ephemeral" ] }
  typeOfStorage_1_is_ephemeral: { equals: [ { get_param: [ nfv, storageAttributes, 1, typeOfStorage
] }, "ephemeral" ] }
  typeOfStorage_2_is_ephemeral: { equals: [ { get_param: [ nfv, storageAttributes, 2, typeOfStorage
] }, "ephemeral" ] }
  typeOfStorage_0_is_swap: { equals: [ { get_param: [ nfv, storageAttributes, 0, typeOfStorage ] },
"swap" ] }
  typeOfStorage_1_is_swap: { equals: [ { get_param: [ nfv, storageAttributes, 1, typeOfStorage ] },
"swap" ] }
  typeOfStorage_2_is_swap: { equals: [ { get_param: [ nfv, storageAttributes, 2, typeOfStorage ] },
"swap" ] }

resources:
  virtualisedComputeFlavour:
    type: OS::Nova::Flavor
    properties:
      flavorid: { get_param: [ nfv, flavourId ] }
      ram: { get_param: [ nfv, virtualMemory, virtualMemSize ] }
      vcpus: { get_param: [ nfv, virtualCpu, numVirtualCpu ] }
      disk:
        if:
          - typeOfStorage_0_is_disk
          - { get_param: [ nfv, storageAttributes, 0, sizeOfStorage ] }
          - if:
              - typeOfStorage_1_is_disk
              - { get_param: [ nfv, storageAttributes, 1, sizeOfStorage ] }
              - if:
```

```
                - typeOfStorage_2_is_disk
                - { get_param: [ nfv, storageAttributes, 2, sizeOfStorage ] }
                - 0
      ephemeral:
        if:
          - typeOfStorage_0_is_ephemeral
          - { get_param: [ nfv, storageAttributes, 0, sizeOfStorage ] }
          - if:
              - typeOfStorage_1_is_ephemeral
              - { get_param: [ nfv, storageAttributes, 1, sizeOfStorage ] }
              - if:
                  - typeOfStorage_2_is_ephemeral
                  - { get_param: [ nfv, storageAttributes, 2, sizeOfStorage ] }
                  - 0
      swap:
        if:
          - typeOfStorage_0_is_swap
          - { get_param: [ nfv, storageAttributes, 0, sizeOfStorage ] }
          - if:
              - typeOfStorage_1_is_swap
              - { get_param: [ nfv, storageAttributes, 1, sizeOfStorage ] }
              - if:
                  - typeOfStorage_2_is_swap
                  - { get_param: [ nfv, storageAttributes, 2, sizeOfStorage ] }
                  - 0
      extra_specs:
        if:
          - cpuPinningPolicy_is_static
          - if:
              - accelerationCapabilityForVirtualNetworkInterface_is_dpdk
              - if:
                  - accelerationCapabilityForVirtualComputeFlavor_is_gpu
                  - {
                      "hw:cpu_policy": "dedicated",
                      "hw:cpu_sockets": { get_param: [ nfv, virtualCpu, virtualCpuPinning,
cpuPinningRules, sockets ] },
                      "hw:cpu_cores": { get_param: [ nfv, virtualCpu, virtualCpuPinning,
cpuPinningRules, cores ] },
                      "hw:cpu_threads": { get_param: [ nfv, virtualCpu, virtualCpuPinning,
cpuPinningRules, threads ] },
                      "hw:mem_page_size": "large",
                      "pci_passthrough:alias": "a1:2"
                    }
                  - {} ### another pattern's process(e.g. crypto).
              - {} ### another pattern's process(e.g. RDMA, packet dispatch, TCP Chimney).
          - {} ### another pattern's process(e.g. dynamic).
```

NOTE: The unit of sizeOfStorage is GB, but the unit of swap area is MB in OpenStack.

There are no output parameters related to the flavour creation, as the flavour can be identified based on the `stackId` attribute passed as part of the input parameters.

A.3.5 Example#5: API mapping of output parameters for Allocate Virtualised Storage Resource operation

Clause A.3.3 provides an example of the Allocate Virtualised Storage Resource operation using template outputs. The present clause illustrates the alternative of API parameter mapping to allow a client to access the output information defined in the present document data model in clause 8.3.1 using the OpenStack HEAT API. Table A.3.5-1 lists the attributes that a client would obtain from invoking "resource show" in the HEAT API in order to access the output information defined in the present document data model. For the input part, this alternative uses the same approach as clause A.3.3.

Table A.3.5-1: Output attributes mapping between the present document data model and Openstack HEAT API

Attribute per the present document data model	Attribute per HEAT API
nfvStorageInfo	
>storageId	resource/attributes/id
>storageName	resource/attributes/name
>flavourId	(not supported by HEAT)
>sizeOfStorage	resource/attributes/size
>rdmaEnabled	(not supported by HEAT)
>ownerId	resource/attributes/attachments/server_id
>zoneId	resource/attributes/availability_zone
>hostId	os-vol-host-attr:host
>operationalState	resource/attributes/status
>metadata	resource/attributes/metadata

A.4 Complex templates

ETSI GS NFV-IFA 005 [1] and ETSI GS NFV-IFA 006 [2] define interfaces for the management of individual resources, and the examples in the present document are introduced to be consistent with those specifications.

On the other hand, HEAT is primarily used to manage a group of different types of virtualised resources, called "stack".

Sets of virtualised resources that are commonly used in the different stacks can be written in individual dedicated templates, so that they can be referenced in other templates. Such templates can be referred to as "nested templates".

Multiple levels of nesting of templates are allowed.

Annex B (informative): Explanations of concepts

B.1 Introduction

This annex provides explanations of certain concepts introduced in the present document.

B.2 Concept of descriptor-based virtualised resource management

In the present document, input and output parameter data models are specified using YAML [4].

The input parameter data model consists of:

- an input section defined in the virtualised resource descriptor;
- the corresponding actual input data as arguments to exchange when invoking operations over the Vi-Vnfm and Or-Vi reference points;
- the corresponding input data definitions in YAML [4], which are presented in the "parameter to be used as input" subclause of clauses 5 to 8 of the present document.

The input data to be used over the reference points in virtualised resource descriptor is compliant with input data definition.

The output parameter data model consists of:

- an output section defined in the virtualised resource descriptor;
- the corresponding actual output data as return values to exchange in a response to an operation invoked over Vi-Vnfm and Or-Vi reference points;
- the corresponding output data definitions in YAML [4], which are presented in the "parameter to be used as output" subclause of clauses 5 to 8 of the present document.

The output data to be exposed over the reference points from virtualised resource descriptor is compliant with output data definition.

Descriptor-based virtualised resource management provisions virtualised resource via virtualised resource descriptor that includes input section and output section.

Figure B.2-1 illustrates the concepts described above. The method for passing input data as arguments and output data as return values is out of the scope of the present document. The present document provides examples of virtualised resource descriptors, arguments and return values for each solution identified in Annex A.

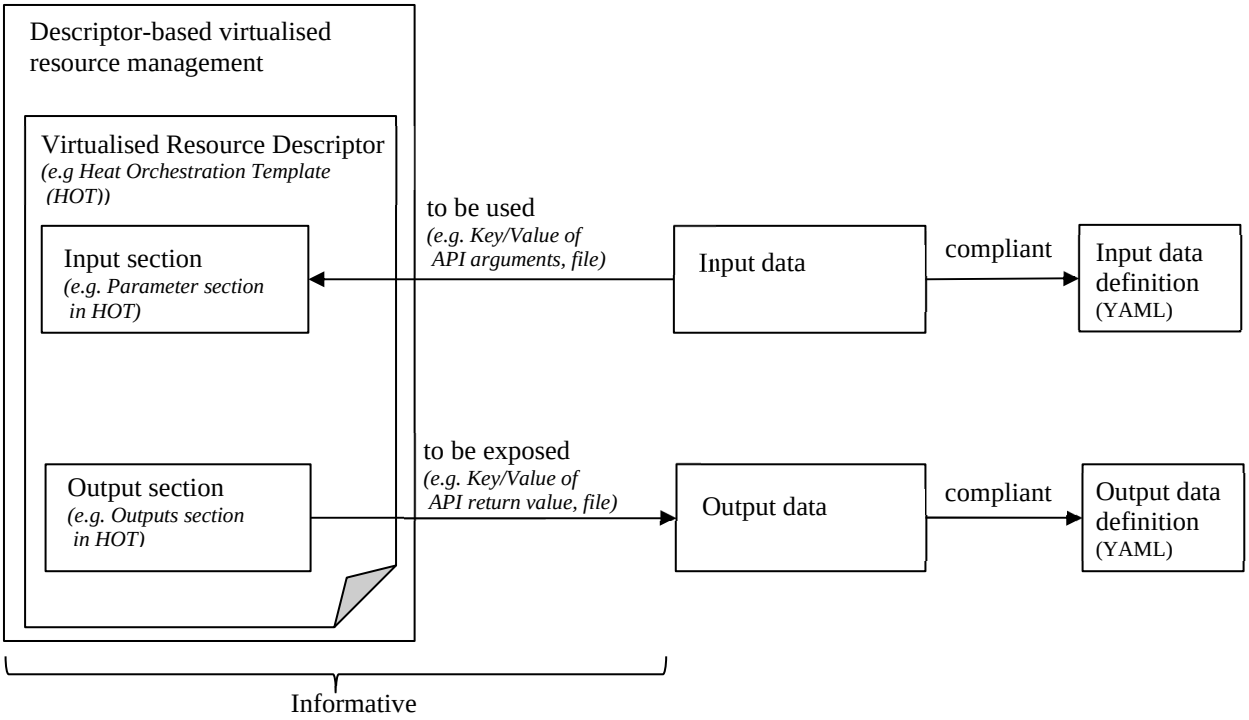

Figure B.2-1: Concept of descriptor-based virtualised resource management

Annex C (informative): Change History

Date	Version	Information about changes
Feb 2021	3.0.1	Initial draft for SOL014ed351
Mar 2021	3.0.2	Implemented - NFVSOL(21)000166 Improvements towards OpenAPI-based data models for general aspects - NFVSOL(21)000160r1 SOL014ed351 Improvements towards OpenAPI-based data models in virtualised storage resource data model - NFVSOL(21)000143r1 SOL014ed351 Improvements towards OpenAPI-based data models in virtualised compute resource data model - NFVSOL(21)000142 SOL014ed Improvements towards OpenAPI-based data models in common data model - NFVSOL(21)000119 SOL014ed351 Improvements towards OpenAPI-based data models in Virtualised Network Management - NFVSOL(21)000059 SOL014 networkResourceName in OpenAPI

History

Document history		
V3.5.1	May 2021	Publication