# ETSI GS NFV-SOL 014 V5.3.1 (2025-08)

**GROUP SPECIFICATION**

# Network Functions Virtualisation (NFV) Release 5; 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-SOL014ed531

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 IPR online database.

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™**, **LTE™** and **5G™** logo 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. The present document also focuses on data models used in the virtualised resource descriptors for the Virtualised Resources Change Notification interfaces and Virtualised Resources Fault Management interfaces. 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 in the ETSI docbox.

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 5; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification".

[2] ETSI GS NFV-IFA 006: "Network Functions Virtualisation (NFV) Release 5; 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.

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

[6] ETSI GS NFV-SOL 001: "Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; NFV descriptors based on TOSCA specification".

[7] JSON Schema.

[8] ETSI GS NFV-SOL 013: "Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; Specification of common aspects for RESTful NFV MANO APIs".

[9] ETSI GS NFV-IFA 045: "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Faults and alarms modelling specification".

## 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 may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

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

[i.2] OpenStack®: "Heat Orchestration Template (HOT) specification".

[i.3] Openstack®-heat: "Orchestration Service APIs".

NOTE: The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. ETSI is not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

[i.4] ETSI GS NFV-IFA 014: "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Network Service Templates Specification".

[i.5] ETSI GS NFV-IFA 011: "Network Functions Virtualisation (NFV) Release 5; 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 GR NFV 003 [i.1] apply.

## 3.2 Symbols

Void.

## 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GR 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.
- Virtualised Resources Change Notification interfaces.
- Virtualised Resources Fault Management 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 document 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 inlined 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
```

## 5.2.5 Parameter: stackName

The parameter used when pointing to a stack of virtual resources defined by a descriptor shall follow the indications provided in Table 5.2.5-1.

**Table 5.2.5-1: Input data model for stackName**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| stackName | String | Name of the stack, given by the consumer. |

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

```
stackName:
  type: string
  description: >
    Name of the stack, given by the consumer
  default: ""
```

## 5.2.6 Parameter: startTime

The parameter used when giving a date and time when the consumption of the resources starts shall follow the indications provided in Table 5.2.6-1.

**Table 5.2.6-1: Input data model for startTime**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| startTime | String | Specifies when the consumption of the resources starts. |

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

```
startTime:
  type: string
  description: >
    Specifies when the consumption of the resources starts
  default: ""
```

### 5.2.7 Parameter: endTime

The parameter used when giving a date and time when the reservation ends shall follow the indications provided in Table 5.2.7-1.

**Table 5.2.7-1: Input data model for endTime**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| endTime | String | Specifies when the reservation ends (when the issuer of the request expects that the resources will no longer be needed) and used by the VIM to schedule the reservation. |

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

```
  endTime:
    type: string
    description: >
      Specifies when the reservation ends (when the issuer of the request expects that the resources
will no longer be needed) and used by the VIM to schedule the reservation
    default: ""
```

### 5.2.8 Parameter: expiryTime

The parameter used when giving a date and time when the VIM can release the reservation in case no allocation request against this reservation was made shall follow the indications provided in Table 5.2.8-1.

**Table 5.2.8-1: Input data model for expiryTime**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| expiryTime | String | Specifies when the VIM can release the reservation in case no allocation request against this reservation was made. |

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

```
  expiryTime:
    type: string
    description: >
      Specifies when the VIM can release the reservation in case no allocation request against this
reservation was made
    default: ""
```

## 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.<br><br>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".<br><br>Allowed to NFVI-PoP, NFVI-Node.<br>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.<br>See note and condition. |
| >>resource | Array of Object | List of identifiers of virtualised resources. |
| >affinityAntiAffinityResourceGroup | String | Identifier of the producer-managed group of virtualised resources with which the actual resource is requested to be affine or anti-affine.<br>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, affinityAntiAffinityResourceGroup shall be supported. The mechanisms shall not be mixed in the scope of a resourceGroup (also known as 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 "inlined 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 "inlined 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.<br>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 "inlined 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. |

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| >operationalState | String | The operational state of the virtual network interface.<br>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 "inlined 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.<br>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.<br>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).<br>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. |
| >>powerStateReqs | Object | The virtual CPU power (state) requirements for the virtualised compute resource. This parameter is reserved for future use in the present document. |

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| >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. |
| >>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:<br>• typeVirtualNic;<br>• 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
          powerStateReqs:
            description: >
              The virtual CPU power (state) requirements for the virtualised compute resource.
              This parameter is reserved for future use in the present document.
            type: object
      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
```

## 6.2.11 Parameter: userData

The parameter used when providing user data to customize the virtualised compute resource at boot time shall follow the indications provided in Table 6.2.11-1.

**Table 6.2.11-1: Input data model for userData**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| userData | Object | Name of the parameter. |
| >content | String | Contains the user data to customize the virtualised compute resource at boot-time. |
| >method | String | Method used as transportation media to convey the content of the userData to the virtualised compute resource.<br>Allowed values: CONFIG_DRIVE, CONFIG_DRIVE_MIME_multi-part. |
| >certificateData | Object | Contains the additional user data to store certificate data for the VNF composed of (fully or partially) virtualised compute resource at boot-time.<br>Shall be present if delegation-mode is used. Otherwise it shall be absent. |
| >>privateKey | String | Private key paired with signed public key. VNFM shall generate both private key and public key and set this attribute. See note. |
| >>certificateFile | String | Signed certificate including public key and certificate chain. See note. |
| >>keystoreFile | String | Keystore which includes the private key, signed certificate and certificate chain, e.g. pkcs#12, pfx.<br>Credentials to read this file shall be provided to the VNF instance by outbound. See note. |
| >>certSubjectData | String | Subject to be signed. |
| >>certificateProfileName | String | Name of certificate profile to be signed. |
| NOTE: Either set of "privateKey" and "certificateFile" or "keystoreFile" but not both shall be present. | | |

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

```
  userData:
    description: >
      Contains user data to customize the virtualised compute resource at boot time
    type: object
    required:
      - content
    properties:
      content:
        type: string
      method:
        type: string
        enum:
          - CONFIG_DRIVE
          - CONFIG_DRIVE_MIME_multi-part
      certificateData:
        type: object
        properties:
          privateKey:
            type: string
          certificateFile:
            type: string
          keystoreFile:
            type: string
          certSubjectData:
            type: string
          certificateProfileName:
            type: string
    default: ""
```

## 6.2.12 Parameter: computePoolReservation

The parameter used when giving amount of compute resources to be reserved shall follow the indications provided in Table 6.2.12-1.

**Table 6.2.12-1: Input data model for computePoolReservation**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| computePoolReservation | Object | Amount of compute resources to be reserved. |
| >numCpuCores | Number | Number of CPU cores to be reserved. |
| >numVcInstances | Number | Number of virtualised container instances to be reserved. |
| >virtualMemSize | Number | Size of virtual memory to be reserved (in MB). |
| >computeAttributes | Object | Information specifying additional attributes of the compute resource to be reserved. |
| >>accelerationCapabilityForVirtualComputePoolReservation | Array of String | Selected acceleration capabilities (e.g. crypto, GPU) from the set of capabilities offered by the compute node acceleration resources. |
| >>cpuArchitecture | String | CPU architecture type. Examples are x86, ARM®. |
| >>virtualCpuOversubscriptionPolicy | String | CPU core oversubscription policy in terms of virtual CPU cores to physical CPU cores/threads on the platform. |

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

```
computePoolReservation:
  description: >
    Amount of compute resources to be reserved.
  type: object
  required:
    - numCpuCores
    - numVcInstances
    - virtualMemSize
  properties:
    numCpuCores:
      type: number
    numVcInstances:
      type: number
    virtualMemSize:
      type: number
    computeAttributes:
      type: object
      properties:
        accelerationCapabilityForVirtualComputePoolReservation
          type: array
          minItems: 0 # lower bound of cardinality
          maxItems: N # upper bound of cardinality
          items:
            type: string
        cpuArchitecture
          type: string
        virtualCpuOversubscriptionPolicy
          type: string
  default: ""
```

## 6.2.13 Parameter: minAmount

The parameter used when giving minimum amount of compute hosts to be reserved shall follow the indications provided in Table 6.2.13-1.

**Table 6.2.13-1: Input data model for minAmount**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| minAmount | Number | Minimum amount of compute hosts to be reserved. |

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

```
minAmount:
  type: number
  description: >
    Minimum amount of compute hosts to be reserved
```

### 6.2.14 Parameter: maxAmount

The parameter used when giving maximum amount of compute hosts to be reserved shall follow the indications provided in Table 6.2.14-1.

**Table 6.2.14-1: Input data model for maxAmount**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| maxAmount | Number | Maximum amount of compute hosts to be reserved. |

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

```
maxAmount:
  type: number
  description: >
    Maximum amount of compute hosts to be reserved
```

### 6.2.15 Parameter: computeHostProperties

The parameter used when giving set of properties that specify the capabilities associated to the compute hosts to be reserved shall follow the indications provided in Table 6.2.15-1.

**Table 6.2.15-1: Input data model for computeHostProperties**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| computeHostProperties | Array of String | Set of properties that specify the capabilities associated to the compute hosts to be reserved.<br>Allowed value: ssd, dpdk, sriov, gpu, fpga, cpu_pin, numa. |

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

```
computeHostProperties:
  description: >
    Set of properties that specify the capabilities associated to the compute hosts to be reserved
  type: array
  minItems: 1 # lower bound of cardinality
  maxItems: N # upper bound of cardinality
  items:
    type: string
    enum:
    - ssd
    - dpdk
    - sriov
    - gpu
    - fpga
    - cpu_pin
    - numa
```

### 6.2.16 Parameter: callbackUriForComputeHostCapacityChangeNotify

The parameter used when providing a URI of the endpoint to send compute host capacity change notifications to in a subscribe operation shall follow the indications provided in Table 6.2.16-1.

**Table 6.2.16-1: Input data model for callbackUriForComputeHostCapacityChangeNotify**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| callbackUriForComputeHostCapacityChangeNotify | String | The URI of the endpoint to send the compute host capacity change notification to. |

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

```
  callbackUriForComputeHostCapacityChangeNotify:
    description: >
      The URI of the endpoint to send the compute host capacity change notification to.
    type: string
    default: ""
```

## 6.2.17 Parameter: inputFilterForComputeHostCapacityChangeNotify

The parameter used when selecting compute host capacity change notifications to in a subscribe operation shall follow the indications provided in Table 6.2.17-1.

**Table 6.2.17-1: Input data model for inputFilterForComputeHostCapacityChangeNotify**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| inputFilterForComputeHostCapacityChangeNotify | Object | Input filter for selecting compute host capacity change notifications. |
| >zoneId | String | Identifier of the resource zone for which the capacity change notifications are requested. When not specified the total capacity managed by the VIM instance will be notified. |
| >threshold | Object | When specified this parameter indicates a capacity value which, once crossed, will trigger a notification. When not specified, notifications are issued at every change (see note). |
| >>resourceType | String | Resource type of capacity to check.<br>Allowed value: cpu, memory, disk. |
| >>capacityInformationType | String | Information type of capacity to check.<br>Allowed value: available, total, reserved, allocated. |
| >>valueOfThreshold | Number | Value to check. |
| >>crossingDirection | String | Direction to check.<br>Allowed value: UP, DOWN.<br>In case of "UP", notification is triggered when the value is greater than "valueOfThreshold".<br>In case of "DOWN", notification is triggered when the value is less than "valueOfThreshold". |
| >capacityInformationType | Array of String | Input parameter for selecting which capacity information the subscription refers to. When not present, all four values are requested.<br>Allowed value: available, total, reserved, allocated. |
| NOTE: The VIM may still implement a minimum-delta threshold in order to avoid an excessive notification flow. | | |

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

```
  inputFilterForComputeHostCapacityChangeNotify:
    description: >
      Input filter for selecting compute host capacity change notifications.
    type: object
    properties:
      zoneId:
        type: string
      threshold:
        type: object
        properties:
          resourceType:
            type: string
            enum:
              - cpu
              - memory
              - disk
          capacityInformationType:
            type: string
            enum:
              - available
              - total
              - reserved
              - allocated
          valueOfThreshold:
            type: number
          crossingDirection:
            type: string
            enum:
```

```
            - UP
            - DOWN
      capacityInformationType:
        type: array
        minItems: 1  # lower bound of cardinality
        maxItems: 4  # upper bound of cardinality
        items:
          type: string
          enum:
            - available
            - total
            - reserved
            - allocated
```

## 6.2.18 Parameter: changeIdForComputeHostCapacityChangeNotify

The parameter used when providing an identifier of the change in the compute host capacity change notification shall follow the indications provided in Table 6.2.18-1.

**Table 6.2.18-1: Input data model for changeIdForComputeHostCapacityChangeNotify**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| changeIdForComputeHostCapacityChangeNotify | String | Identifies a change in the capacity. |

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

```
  changeIdForComputeHostCapacityChangeNotify:
    description: >
      Identifies a change in the capacity.
    type: string
    default: ""
```

## 6.2.19 Parameter: zoneIdForComputeHostCapacityChangeNotify

The parameter used when providing an identifier of the resource zone in the compute host capacity change notification shall follow the indications provided in Table 6.2.19-1.

**Table 6.2.19-1: Input data model for zoneIdForComputeHostCapacityChangeNotify**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| zoneIdForComputeHostCapacity ChangeNotify | String | Identifies the resource zone for which the capacity has changed. When omitted the total capacity managed by the VIM is reported. |

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

```
  zoneIdForComputeHostCapacityChangeNotify:
    description: >
      Identifies the resource zone for which the capacity has changed. When omitted the total
capacity managed by the VIM is reported.
    type: string
    default: ""
```

## 6.2.20 Parameter: resourceDescriptorForComputeHostCapacityChangeNotify

The parameter used when providing a resource type in the compute host capacity change notification shall follow the indications provided in Table 6.2.20-1.

**Table 6.2.20-1: Input data model for resourceDescriptorForComputeHostCapacityChangeNotify**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| resourceDescriptorForComputeH ostCapacityChangeNotify | String | Resource type for which the capacity is changed.<br>Allowed value: cpu, memory, disk. |

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

```
  resourceDescriptorForComputeHostCapacityChangeNotify:
    description: >
      Resource type for which the capacity is changed.
    type: string
    enum:
      - cpu
      - memory
      - disk
```

## 6.2.21 Parameter: capacityInformationForComputeHostCapacityChangeNotify

The parameter used when providing a resource type in the compute host capacity change notification shall follow the indications provided in Table 6.2.21-1.

**Table 6.2.21-1: Input data model for capacityInformationForComputeHostCapacityChangeNotify**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| capacityInformationForComputeHostCapacityChangeNotify | Object | Available, total, reserved and/or allocated/used capacity of the Resource Zone, or the available, total, reserved and/or allocated/used capacity of the VIM in case the zoneIdForComputeHostCapacityChangeNotify is omitted. |
| >availableCapacity | Number | Free capacity available for allocation and reservation. |
| >reservedCapacity | Number | Reserved capacity. |
| >totalCapacity | Number | The total capacity is usually specified as a fixed capacity without variations in time (see note 1). |
| >allocatedCapacity | Number | The allocated capacity is usually specified as the current allocated capacity (see note 2). |
| NOTE 1: VIM does not keep schedules for equipment build-out.<br>NOTE 2: The allocated capacity is given without time variation since the VIM does not have a schedule of future allocations and de-allocations. | | |

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

```
  capacityInformationForComputeHostCapacityChangeNotify:
    description: >
      Available, total, reserved and/or allocated/used capacity of the Resource Zone, or the
available, total, reserved and/or allocated/used capacity of the VIM in case the
zoneIdForComputeHostCapacityChangeNotify is omitted.
    type: object
    properties:
      availableCapacity:
        type: number
      reservedCapacity:
        type: number
      totalCapacity:
        type: number
      allocatedCapacity:
        type: number
```

## 6.2.22 Parameter: existingVirtualComputeResourcesTermination

The parameter used when pointing to identifiers of the existing virtual compute resources that are planned to be terminated by the consumer, and whose capacities information can be considered for computing the requested resource reservation, shall follow the indications provided in Table 6.2.22-1.

**Table 6.2.22-1: Input data model for existingVirtualComputeResourcesTermination**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| existingVirtualComputeResourcesTermination | Array of String | Identifiers of the existing virtual compute resources that are planned to be terminated by the consumer, and whose capacities information can be considered for computing the requested resource reservation. |

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

```
  existingVirtualComputeResourcesTermination:
    description: >
      Identifiers of the existing virtual compute resources that are planned to be terminated by the
consumer, and whose capacities information can be considered for computing the requested resource
reservation
    type: array
    minItems: 1 # lower bound of cardinality
    maxItems: N # upper bound of cardinality
    items:
      type: string
```

# 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®.<br>See note 1. |
| >>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).<br>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. |
| >>powerState | Object | The virtual CPU power state information of the virtualised compute resource. This parameter is reserved for future use in the present document. |
| >virtualMemory | Object | The virtual memory of the compute as VirtualMemory. |
| >>virtualMemSize | Number | Amount of virtual memory in byte. |

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| >>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 VirtualNetwork Port) | 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. |
| >>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.<br>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. See note 2. |
| >>>hostId | String | Identifier of the host where the virtualised storage resource is allocated. |
| >>>operationalState | String | Operational state of the resource.<br>Allowed value: enabled, 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. |
| >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. See note 2. |
| >hostId | String | Identifier of the host the virtualised compute resource is allocated on. |
| >operationalState | String | Operational state of the compute resource.<br>Possible values are: "enabled" or "disabled". See note 3. |
| >runningState | String | Running state of the compute resource.<br>Possible values are: "started", "stopped", "paused", "suspended" or "rebooting". See note 3. |
| >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 1: See "cpu_architecture" in tosca.datatypes.nfv.VirtualCpu, ETSI GS NFV-SOL 001 [6].<br>NOTE 2: This identifier is unique within an NFVI-PoP and is assigned by the PIM during resource zone creation. The VIM uses the same identifier for a resource zone that is assigned by the PIM.<br>NOTE 3: Values of the operationalState attribute are rather administrative states, while the runningState attribute provides operational state information. | | |

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
    - runningState
  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
          powerState:
            description: >
              The virtual CPU power state information of the virtualised compute resource.
              This parameter is reserved for future use in the present document.
            type: object
      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
        runningState:
          type: string
          enum:
            - started
            - stopped
            - paused
            - suspended
            - rebooting
        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.<br>Possible values are: "network", "subnet", "network-port", or "routing-resource". |

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
  - routing-resource
  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 | 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.<br>See note. |
| >>>end | String | The last IP address in the addressPool.<br>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.<br>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.<br>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. |

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| >>start | String | The first IP address in the addressPool.<br>See note. |
| >>end | String | The last IP address in the addressPool.<br>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.<br>Allowed_values: affinity, anti-affinity. |

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| >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 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 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 ETSI GS NFV-IFA 006 [2]) are supported, affinityAntiAffinityResourceGroup shall be supported. The mechanisms shall not be mixed in the scope of a resourceGroup (also known as 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.<br>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.<br>See note. |
| >>>end | String | The last IP address in the addressPool.<br>See note. |

| 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. |
| >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.<br>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.<br>See note. |
| >>end | String | The last IP address in the addressPool.<br>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
```

## 7.2.15 Parameter: typeRoutingResourceData

The parameter used when setting the routing resource data provides information about the particular routing resource shall follow the indications in Table 7.2.15-1.

**Table 7.2.15-1: Input data model for typeRoutingResourceData**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| typeRoutingResourceData | Object | The routing resource data provides information about the particular routing resource. |
| >name | String | Name of the routing resource. |
| >distributed | Boolean | If present, it specifies whether the routing resource is requested to be distributed. |
| >operationalState | String | Operational state of the routing resource.<br>Possible values are: "enabled", "disabled". |
| >ha | Boolean | If present, it specifies whether the routing resource is requested to have high availability. |
| >zoneId | String | If present, it identifies the Resource Zone where the routing resource is requested to be allocated. |
| >subnet | Array of String | Identifier of the subnet(s) that the routing resource is associated with. |
| >routingSet | Array of Object | Specifies the information used to set the routing resource. |
| >>destination | String | The destination address of one route. |
| >>nexthop | String | The nexthop IP address of one route. |
| >>ipVersion | String | Specifies IP version of the destination and nexthop addresses.<br>Possible values are: "IPv4", "IPv6". |
| >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 typeRoutingResourceData shall comply with the following definition:

```
typeRoutingResourceData:
  description: >
    The routing resource data provides information about the particular routing resource
  required:
```

```
      - name
      - operationalState
    type: object
    properties:
      name:
        type: string
      distributed:
        type: Boolean
      operationalState:
        type: string
        enum:
          - enabled
          - disabled
      ha:
        type: Boolean
      zoneId:
        type: string
      subnet:
        type: array
        minItems: 1  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: string
      routingSet:
        type: array
        minItems: 1  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          required:
            - destination
            - nexthop
            - ipVersion
          type: object
          properties:
            destination:
              type: string
            nexthop:
              type: string
            ipVersion:
              type: string
              enum:
                - IPv4
                - IPv6
      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
    default: ""
```

## 7.2.16 Parameter: mirroringJobName

The parameter used when providing a name of a data flow mirroring job shall follow the indications provided in Table 7.2.16-1.

**Table 7.2.16-1: Input data model for mirroringJobName**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| mirroringJobName | String | Name of Data Flow Mirroring Job. |

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

```
  mirroringJobName:
    description: >
      Name of Data Flow Mirroring Job
    type: string
    default: ""
```

## 7.2.17 Parameter: descriptionForDataFlowMirroringJob

The parameter used when providing a description of a data flow mirroring job shall follow the indications provided in Table 7.2.17-1.

**Table 7.2.17-1: Input data model for descriptionForDataFlowMirroringJob**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| descriptionForDataFlowMirroringJob | String | Information description of Data Flow Mirroring Job. |

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

```
descriptionForDataFlowMirroringJob:
  description: >
    Information description of Data Flow Mirroring Job
  type: string
  default: ""
```

## 7.2.18 Parameter: collectorDetails

The parameter used when providing information about where the mirrored flow is to be delivered shall follow the indications provided in Table 7.2.18-1.

**Table 7.2.18-1: Input data model for collectorDetails**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| collectorDetails | Object | Information about where the mirrored flow is to be delivered. |
| >networkPortId | String | Identifier of the collector network port managed by VIM where the mirrored flow is to be delivered. |

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

```
collectorDetails:
  description: >
    Information about where the mirrored flow is to be delivered
  type: object
  required:
    - networkPortId
  properties:
    networkPortId:
      type: string
  default: ""
```

## 7.2.19 Parameter: dataFlowDetails

The parameter used when providing information about the data flows that need to be mirrored shall follow the indications provided in Table 7.2.19-1.

**Table 7.2.19-1: Input data model for dataFlowDetails**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| dataFlowDetails | Object | Information about the data flows that need to be mirrored. |
| >networkPortId | String | Identifier of the network port which acts as the mirroring point of the data flow to be mirrored. |
| >direction | String | The direction of the data flow that are requested to be mirrored. Possible values are: "in", "out", "both". |
| >targetIpAddress | String | If present, only the data flow to or/and from the target IP address shall be mirrored. |

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

```
dataFlowDetails:
  description: >
    Information about the data flows that need to be mirrored
```

```
    type: object
    required:
      - networkPortId
    properties:
      networkPortId:
        type: string
      direction:
        type: string
        enum:
          - in
          - out
          - both
      targetIpAddress:
        type: string
    default: ""
```

# 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 1) | 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. These 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. See note 2. |
| >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. |
| NOTE 1: 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. | | |
| NOTE 2: This identifier is unique within an NFVI-PoP and is assigned by the PIM during resource zone creation. The VIM uses the same identifier for a resource zone that is assigned by the PIM. | | |

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.<br>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.<br>See note. |
| >>end | String | The last IP address in the addressPool.<br>See note. |
| >operationalState | String | The operational state of the virtualised subnetwork.<br>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.<br>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.<br>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
```

## 7.3.4 Parameter: nfvRoutingResourceInfo

The parameter is used when returning information for a routing resource, and its output data model shall follow the indications provided in Table 7.3.4-1. This parameter maps to the "routingResourceData" parameter defined in ETSI GS NFV-IFA 005 [1].

**Table 7.3.4-1: Output data model for nfvRoutingResourceInfo**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| nfvRoutingResourceInfo | Object | If routing resource types are created satisfactorily, it contains the data relative to the allocated routing resource. |
| >resourceId | String | Identifier of the routing resource. |
| >name | String | Name of the routing resource. |
| >subnet | Array of String | References the network subnet(s). |
| >operationalState | String | Operational state of the routing resource.<br>Possible values are: "enabled", "disabled". |
| >ha | Boolean | Specifies whether the routing resource has high availability. It shall be present when support for high availability is enabled. |
| >zoneId | String | If present, it identifies the Resource Zone where the routing resource has been allocated. See note. |
| >distributed | Boolean | Specifies whether the routing resource is distributed. It shall be present when support for distributed routing resource is enabled. |
| >routingSet | Array of Object | Specifies the information used to set the routing resource. |
| >>destination | String | The destination address of one route. |
| >>nexthop | String | The nexthop IP address of one route. |
| >>ipVersion | String | Specifies IP version of the destination and nexthop addresses.<br>Possible values are: "IPv4", "IPv6". |
| >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: This identifier is unique within an NFVI-PoP and is assigned by the PIM during resource zone creation. The VIM uses the same identifier for a resource zone that is assigned by the PIM. | | |

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

```
  nfvRoutingResourceInfo:
    type: object
    required:
      - resourceId
      - name
      - operationalState
    properties:
      resourceId:
        type: string
      name:
        type: string
```

```
      subnet:
        type: array
        minItems: 1  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: string
      operationalState:
        type: string
        enum:
          - enabled
          - disabled
      ha:
        type: Boolean
      zoneId:
        type: string
      distributed:
        type: Boolean
      routingSet:
        type: array
        minItems: 1  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          required:
            - destination
            - nexthop
            - ipVersion
          type: object
          properties:
            destination:
              type: string
            nexthop:
              type: string
            ipVersion:
              type: string
              enum:
                - IPv4
                - IPv6
      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
```

## 7.3.5 Parameter: nfvMirroringJob

The parameter is used returning information for a data flow mirroring job, and its output data model shall follow the indications provided in Table 7.3.5-1. This parameter maps to the "mirroringJob" parameter defined in ETSI GS NFV-IFA 005 [1].

**Table 7.3.5-1: Output data model for nfvMirroringJob**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| nfvMirroringJob | Object | Information about the Data Flow Mirroring Job that has been created. |
| >mirroringJobId | String | Unique identifier of the Data Flow Mirroring Job. |
| >mirroringJobName | String | Name of the Data Flow Mirroring Job. |
| >description | String | Information description of the Data Flow Mirroring Job. |
| >collectorDetails | Object | Information about where the mirrored flow is to be delivered. |
| >>networkPortId | String | Identifier of the collector network port managed by VIM where the mirrored flow is to be delivered. |
| >dataFlowDetails | Object | Information about the data flows that need to be mirrored. |
| >>networkPortId | String | Identifier of the network port which acts as the mirroring point of the data flow to be mirrored. |
| >>direction | String | The direction of the data flow that are requested to be mirrored. Possible values are: "in", "out", "both". |
| >>targetIpAddress | String | If present, only the data flow to or/and from the target IP address shall be mirrored. |

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

```
  nfvMirroringJob:
    type: object
    required:
      - mirroringJobId
      - mirroringJobName
      - description
      - collectorDetails
      - dataFlowDetails
    properties:
      mirroringJobId:
        type: string
      mirroringJobName:
        type: string
      description:
        type: string
      collectorDetails:
        type: object
        required:
          - networkPortId
        properties:
          networkPortId:
            type: string
      dataFlowDetails:
        type: object
        required:
          - networkPortId
        properties:
          networkPortId:
            type: string
          direction:
            type: string
            enum:
              - in
              - out
              - both
          targetIpAddress:
            type: string
```

# 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.<br>Allowed to affinity and anti-affinity. |
| >scopeOfAffinityOrAntiAffinityConstraintForStorage | String | Qualifies the scope of the constraint for the virtualised storage resource.<br>In case of storage resource:<br>e.g. NFVI-Node.<br>Persistent storage node is a type of NFVI-Node which supports, for example, Object, Block or File-based storage service.<br>Ephemeral storage service is supported in a compute node. So this is not included in this attribute.<br>Allowed to NFVI-Node.<br>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.<br>See note and condition. |
| >>resource | Array of String | List of identifiers of virtualised resources. |
| >affinityAntiAffinityResourceGroup | String | Identifier of the producer-managed group of virtualised resources with which the actual resource is requested to be affine or anti-affine.<br>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, affinityAntiAffinityResourceGroup shall be supported. The mechanisms shall not be mixed in the scope of a resourceGroup (also known as 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. See note. |
| >hostId | String | Identifier of the host where the virtualised storage resource is allocated. |
| >operationalState | String | Operational state of the resource.<br>Allowed value: enabled, 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: This identifier is unique within an NFVI-PoP and is assigned by the PIM during resource zone creation. The VIM uses the same identifier for a resource zone that is assigned by the PIM. | | |

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
```

# 9 Data model for Virtualised Resources Change Notification

## 9.1 Description

This clause specifies data models for input and output parameters for Virtualised Resources Change Notification.

## 9.2 Parameters to be used as input

### 9.2.1 Parameter: callbackUriForChangeNotify

The parameter used when providing a URI of the endpoint to send change notifications to in a subscribe operation shall follow the indications provided in Table 9.2.1-1.

**Table 9.2.1-1: Input data model for callbackUriForChangeNotify**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| callbackUriForChangeNotify | String | The URI of the endpoint to send the change notification to. |

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

```
  callbackUri:
    description: >
      The URI of the endpoint to send the change notification to.
    type: string
    default: ""
```

### 9.2.2 Parameter: inputFilter

The parameter used when selecting change notifications to in a subscribe operation shall follow the indications provided in Table 9.2.2-1.

**Table 9.2.2-1: Input data model for inputFilter**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| inputFilter | Object | Input filter for selecting change notifications. |
| >hostId | String | Identifier of the host. When selected host will change (e.g. maintenance), VIM will send the change notification. |

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

```
  inputFilter:
    description: >
      Input filter for selecting change notifications.
    type: object
    properties:
      hostId:
        description: >
          Identifier of the host. When selected host will change (e.g. maintenance), VIM will send
the change notification.
        type: string
```

### 9.2.3 Parameter: changeId

The parameter used when providing an identifier of the change on the virtualised resource in a change notification shall follow the indications provided in Table 9.2.3-1.

**Table 9.2.3-1: Input data model for changeId**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| changeId | String | Unique identifier of the change on the virtualised resource. |

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

```
  changeId:
    description: >
      Unique identifier of the change on the virtualised resource.
    type: string
    default: ""
```

### 9.2.4 Parameter: virtualisedResourceId

The parameter used when providing the identifier of the instantiated virtualised resource for which the change notification is issued shall follow the indications provided in Table 9.2.4-1.

**Table 9.2.4-1: Input data model for virtualisedResourceId**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| virtualisedResourceId | String | Identifier of the instantiated virtualised resource for which the change notification is issued. |

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

```
  virtualisedResourceId:
    description: >
      Identifier of the instantiated virtualised resource for which the change notification is
issued.
    type: string
    default: ""
```

### 9.2.5 Parameter: virtualisedResourceGroupId

The parameter used when providing the identifier of the affinity or anti-affinity group of the virtualised resource for which the change notification is issued shall follow the indications provided in Table 9.2.5-1.

**Table 9.2.5-1: Input data model for virtualisedResourceGroupId**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| virtualisedResourceGroupId | String | Identifier of the affinity or anti-affinity group of the virtualised resource for which the change notification is issued. |

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

```
  virtualisedResourceGroupId:
    description: >
      Identifier of the affinity or anti-affinity group of the virtualised resource for which the
change notification is issued.
    type: string
    default: ""
```

### 9.2.6 Parameter: endOfChange

The parameter used when providing whether this change notification is the end of the changes shall follow the indications provided in Table 9.2.6-1.

**Table 9.2.6-1: Input data model for endOfChange**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| endOfChange | Boolean | If the value is True it indicates the end of the changes of virtualised resources for which the notification of type of change is issued. |

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

```
  endOfChange:
    description: >
      If the value is True it indicates the end of the changes of virtualised resources for which
the the notification of type of change is issued.
    type: boolean
    default: "true"
```

### 9.2.7 Parameter: changeTime

The parameter used when providing the time of changes in a change notification shall follow the indications provided in Table 9.2.7-1.

**Table 9.2.7-1: Input data model for changeTime**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| changeTime | String | Specifies the anticipated time of change of the virtualised resource for which the change notification is issued or the ending time of changes of virtualised resources if the value of the endOfChange is "true". |

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

```
  changeTime:
    description: >
      Specifies the anticipated time of change of the virtualised resource for which the change
notification is issued or the ending time of changes of virtualised resources if the value of the
endOfChange is "true".
    type: string
    default: ""
```

### 9.2.8 Parameter: vimId

The parameter used when providing the identifier of the VIM reporting the change notification shall follow the indications provided in Table 9.2.8-1.

**Table 9.2.8-1: Input data model for vimId**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| vimId | String | Identifier of the VIM reporting the change. |

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

```
  vimId:
    description: >
      Identifier of the VIM reporting the change.
    type: string
    default: ""
```

### 9.2.9 Parameter: changeType

The parameter used when providing the type of change notification shall follow the indications provided in Table 9.2.9-1.

**Table 9.2.9-1: Input data model for changeType**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| changeType | String | Categorizes the type of change. Possible values can be related to maintenance and operation of the NFVI.<br>Allowed value: normal, maintenance, evacuation, optimization. |

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

```
  changeType:
    description: >
      Categorizes the type of change. Possible values can be related to maintenance and operation of
the NFVI.
    type: string
    enum:
      - normal
      - maintenance
      - evacuation
      - optimization
    default: ""
```

### 9.2.10 Parameter: changedResourceData

The parameter used when providing details of the changes of the resource in a change notification shall follow the indications provided in Table 9.2.10-1.

**Table 9.2.10-1: Input data model for changedResourceData**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| changedResourceData | String | Details of the changes of the resource. Its content can differ based on the different values of the attribute changeType. |

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

```
  changedResourceData:
    description: >
      Details of the changes of the resource. Its content can differ based on the different values
of the attribute changeType.
    type: string
    default: ""
```

## 9.3 Parameters to be used as output

None.

# 10 Data model for Virtualised Resources Fault Management

## 10.1 Description

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

# 10.2 Parameters to be used as input

## 10.2.1 Parameter: callbackUriForFaultNotify

The parameter used when providing a URI of the endpoint to send fault notifications to in a subscribe operation shall follow the indications provided in Table 10.2.1-1.

**Table 10.2.1-1: Input data model for callbackUriForFaultNotify**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| callbackUriForFaultNotify | String | The URI of the endpoint to send the fault notification to. |

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

```
  callbackUriForFaultNotify:
    description: >
      The URI of the endpoint to send the fault notification to.
    type: string
    default: ""
```

## 10.2.2 Parameter: filter

The parameter used when selecting fault notifications to in a subscribe operation shall follow the indications provided in Table 10.2.2-1.

**Table 10.2.2-1: Input data model for filter**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| filter | Object | Input filter for selecting fault notifications. |
| >computeId | String | Identifier of the virtualised compute resource. When selected virtualised compute resource was affected by faults, VIM will send notifications. |
| >networkResourceId | String | Identifier of the virtualised network resource. When selected virtualised network resource was affected by faults, VIM will send notifications. |
| >storageId | String | Identifier of the virtualised storage resource. When selected virtualised storage resource was affected by faults, VIM will send notifications. |

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

```
  filter:
    description: >
      Input filter for selecting fault notifications.
    type: object
    properties:
      computeId:
        description: >
          Identifier of the virtualised compute resource. When selected virtualised compute resource
was affected by faults, VIM will send notifications.
        type: string
      networkResourceId:
        description: >
          Identifier of the virtualised network resource. When selected virtualised network resource
was affected by faults, VIM will send notifications.
        type: string
      storageId:
        description: >
          Identifier of the virtualised storage resource. When selected virtualised storage resource
was affected by faults, VIM will send notifications.
        type: string
```

## 10.2.3 Parameter: alarm

The parameter used when providing Information about an alarm in a fault notification shall follow the indications provided in Table 10.2.3-1.

**Table 10.2.3-1: Input data model for alarm**

| Parameter Name and Attributes | Type | Description |
|---|---|---|
| alarm | Object | Information about an alarm. |
| >alarmId | String | Alarm identifier. |
| >managedObjectId | String | Identifier of the affected managed Object. The Managed Objects for this information element will be virtualised resources.<br>A virtualised resource can have fault monitored sub-object types and identification information is carried as defined in the respective Alarm definition as defined in clause 7.2 of ETSI GS NFV-IFA 045 [9], e.g. using the "faultDetails" attribute. |
| >alarmRaisedTime | String | Timestamp indicating when the alarm was first raised by the managed object. |
| >alarmChangedTime | String | Timestamp indicating when the alarm was last changed. It shall be present if the alarm has been updated. |
| >alarmClearedTime | String | Timestamp indicating when the alarm was cleared. It shall be present if the alarm has been cleared. |
| >state | String | State of the alarm.<br>Allowed value: FIRED, UPDATED, CLEARED. |
| >perceivedSeverity | String | Perceived severity of the virtualised managed object failure.<br>Allowed value: CRITICAL, MAJOR, MINOR, WARNING, INDETERMINATE, CLEARED.<br>Valid values applicable to specific Alarms are specified as "Perceived severity" values of the Alarm applicable to Vi-Vnfm or Or-Vi reference point, as defined in clause 7.2 of ETSI GS NFV-IFA 045 [9]. |
| >eventTime | String | Timestamp indicating when the fault was observed. |
| >eventType | String | Type of the event.<br>Allowed value: COMMUNICATION_ALARM, PROCESSING_ALARM, ENVIRONMENT_ALARM, QOS_ALARM, EQUIPMENT_ALARM.<br>Valid values applicable to specific Alarms are specified as "Event type" values of the Alarm applicable to Vi-Vnfm or Or-Vi reference point, as defined in clause 7.2 of ETSI GS NFV-IFA 045 [9]. |
| >faultType | String | Information related to the type of the fault. The allowed values for the faultType attribute depend on the type of the related managed object.<br>Valid values applicable to specific Alarms are specified as "Alarm definition identifier" values of the Alarm applicable to Vi-Vnfm or Or-Vi reference point, as defined in clause 7.2 of ETSI GS NFV-IFA 045 [9]. |
| >probableCause | String | Information about the probable cause of the fault.<br>Valid values applicable to specific Alarms are specified as "Probable cause" values of the Alarm applicable to Vi-Vnfm or Or-Vi reference point, as defined in clause 7.2 of ETSI GS NFV-IFA 045 [9]. |
| >isRootCause | Boolean | Parameter indicating if this fault is the root for other correlated alarms. If "true", then the alarms listed in the parameter correlatedAlarmId are caused by this fault. |
| >correlatedAlarmId | Array of String | List of other alarms correlated to this fault. |
| >faultDetails | Array of String | Provides additional information about the fault.<br>Valid values applicable to specific Alarms are specified as "Fault details" values of the Alarm applicable to Vi-Vnfm or Or-Vi reference point, as defined in clause 7.2 of ETSI GS NFV-IFA 045 [9]. |

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

```
alarm:
  description: >
    Information about an alarm.
  type:  object
  properties:
    alarmId:
      type: string
    managedObjectId:
      type: string
    alarmRaisedTime:
      type: string
    alarmChangedTime:
      type: string
    alarmClearedTime:
      type: string
    state:
```

```
        type: string
        enum:
          - FIRED
          - UPDATED
          - CLEARED
      perceivedSeverity:
        type: string
        enum:
          - CRITICAL
          - MAJOR
          - MINOR
          - WARNING
          - INDETERMINATE
          - CLEARED
      eventTime:
        type: string
      eventType:
        type: string
        enum:
          - COMMUNICATION_ALARM
          - PROCESSING_ALARM
          - ENVIRONMENT_ALARM
          - QOS_ALARM
          - EQUIPMENT_ALARM
      faultType:
        type: string
      probableCause:
        type: string
      isRootCause:
        type: boolean
      correlatedAlarmId:
        type: array
        minItems: 1  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: string
      faultDetails:
        type: array
        minItems: 1  # lower bound of cardinality
        maxItems: N  # upper bound of cardinality
        items:
          type: string
```

## 10.3 Parameters to be used as output

None.

# 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 [i.2].

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": {
        "content": "test-userData",
        "method": "CONFIG_DRIVE_MIME_multi-part",
        "certificateData": {
          "privateKey": "-----BEGIN RSA PRIVATE KEY-----HZLFmek5Kb...",
          "certificateFile": "-----BEGIN CERTIFICATE-----eTDGgITr6a..."
        }
      }
    },
    "gap": {
      "networkId": {
        "0": "c4440f4f-66ef-4e42-9c41-7b3b449813d1",
        "1": "f6a6471d-032d-4c58-9432-c17ea1f72873",
        "2": "91652985-5fed-43f9-adae-8af0471bda03"
       }
    }
  },
  "stack_name": "compute-test-stack"
}
```

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 `10` (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 "`stack_name`" is a required parameter to generate an identifier that points to a stack of the virtualised resources, as shown in clause A.3.6.

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

```
heat_template_version: 2021-04-16
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" ] }
  userData_method_is_CONFIG_DRIVE: { equals : [ { get_param: [ nfv, userData, method ] },
"CONFIG_DRIVE" ] }
  userData_method_is_CONFIG_DRIVE_MIME_multi-part: { equals : [ { get_param: [ nfv, userData, method
] }, "CONFIG_DRIVE_MIME_multi-part" ] }

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 ] }

  userDataContentResource:
    type: OS::Heat::SoftwareConfig
    properties:
      config: { get_param: [ nfv, userData, content ] }

  userDataCertificateDataResource:
    type: OS::Heat::SoftwareConfig
    properties:
      config: { get_param: [ nfv, userData, certificateData ] }

  configDriveMIMEMultiPartResource:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: { get_resource: userDataContentResource }
      - config: { get_resource: userDataCertificateDataResource }

  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 ] }
      config_drive:
        if:
          - userData_method_is_CONFIG_DRIVE
          - true
          - if:
            - userData_method_is_CONFIG_DRIVE_MIME_multi-part
            - true
            - false
      user_data:
        if:
          - userData_method_is_CONFIG_DRIVE_MIME_multi-part
          - { get_resource: configDriveMIMEMultiPartResource }
          - { get_resource: userDataContentResource }
      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: [ forOutput-flavorResource, show, 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.3]. 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.3] 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 [i.2].

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"
                        }
                    ]
                }
            }
        }
    },
    "stack_name": "network-test-stack"
}
```

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 "`stack_name`" is a required parameter to generate an identifier that points to a stack of the virtualised resources, as shown in clause A.3.6.

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

```
heat_template_version: 2021-04-16
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" ] }
  networkResourceType_is_routing-resource: { equals: [ { get_param: [ nfv, networkResourceType ] },
"routing-resource" ] }
  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" ] }
```

```
  typeRoutingResourceData_operationalState_is_ENABLED: { equals: [ { get_param: [ nfv,
typeRoutingResourceData, operationalState ] }, "ENABLED" ] }

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-routingResourceStatusResource:
    type: OS::Heat::ResourceGroup
    depends_on: routerResource
    properties:
      count: 1
      resource_def:
        type: http://controller/Allocate-Virtualised-Network-Resource-
operation/getResourceStatus.yaml
        properties:
          status: { get_attr: [ routerResource, 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

  routerResource:
    type: OS::Heat::ResourceGroup
    properties:
      count: 1
      resource_def:
        if:
          - networkResourceType_is_routing-resource
          - type: OS::Neutron::Router
            properties:
              name: { get_param: [ nfv, typeRoutingResourceData, name ] }
              distributed: { get_param: [ nfv, typeRoutingResourceData, distributed ] }
              admin_state_up:
                if:
                  - typeRoutingResourceData_operationalState_is_ENABLED
                  - true
                  - false
              ha: { get_param: [ nfv, typeRoutingResourceData, ha ] }
              tags: [ { get_param: [ nfv, typeRoutingResourceData, metadata ] } ]
          - type: OS::Heat::None

  routerInterfaceResources:
    type: OS::Heat::ResourceGroup
    depends_on: routerResource
    properties:
      count: 1
      index_var: "none"
      resource_def:
        if:
          - networkResourceType_is_routing-resource
          - type: OS::Heat::ResourceGroup
            properties:
              count: { get_param: [ nfv, typeRoutingResourceData, subnet, count ] }
              resource_def:
                type: http://controller/Allocate-Virtualised-Network-Resource-
operation/createRouterInterface.yaml
                properties:
                  subnet: { get_param: [ nfv, typeRoutingResourceData, subnet ] }
                  routerId: { get_attr: [ routerResource, refs, 0 ] }
                  index: "%index%"
```

```
          - 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 ] }

  nfvRoutingResourceInfo:
    value:
      if:
       - networkResourceType_is_routing-resource
       - str_replace:
           template: |
             {
               "resourceId": "$resourceId",
               "name": "$name",
               "operationalState": "$routingResourceOperationalState",
               "ha": "$ha",
               "zoneId": "$zoneId",
               "distributed": "$distributed",
               "metadata": "$metadata"
             }
           params:
             $resourceId: { get_attr: [ routerResource, resource.0.show, id ] }
             $name: { get_attr: [ routerResource, resource.0.show, name ] }
             $routingResourceOperationalState: { get_attr: [ forOutput-
routingResourceStatusResource, resource.0.status ] }
             $ha: { get_attr: [ routerResource, resource.0.show, ha ] }
             $zoneId: { get_attr: [ routerResource, resource.0.show, availability_zones ] }
             $distributed: { get_attr: [ routerResource, resource.0.show, distributed ] }
             $metadata: { get_attr: [ routerResource, resource.0.show, tags ] }
       - str_replace:
           template:
             This is output of RoutingResource.
             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.3]. 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.3] 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 [i.2].

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"
      }
    }
  },
  "stack_name": "storage-test-stack"
}
```

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 [i.2]. 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, affinityAntiAffinityResourceGroup, 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 "stack_name" is a required parameter to generate an identifier that points to a stack of the virtualised resources, as shown in clause A.3.6.

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

```
heat_template_version: 2018-08-31
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.3]. 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.3] 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 [i.2].

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"
      }
    }
  },
  "stack_name": "flavour-test-stack"
}
```

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 values 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.

The "stack_name" is a required parameter to generate an identifier that points to a stack of the virtualised resources, as shown in clause A.3.6.

```
heat_template_version: 2018-08-31
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.3.6 Example#6: OpenStack Heat API sequence

Clause A.3 provides examples of resource management operation interfaces using OpenStack Orchestration Service API [i.3] and templates. The present clause illustrates sequence diagrams which show how to use the APIs.

Figure A.3.6-1 illustrates the sequence diagram of Allocate Virtualised Compute/Network/Storage Resource operation, Create Virtualised Compute Resource Affinity Or AntiAffinity Constraints Group operation and Attach Virtualised Storage Resource operation using "create stack" and "show stack".

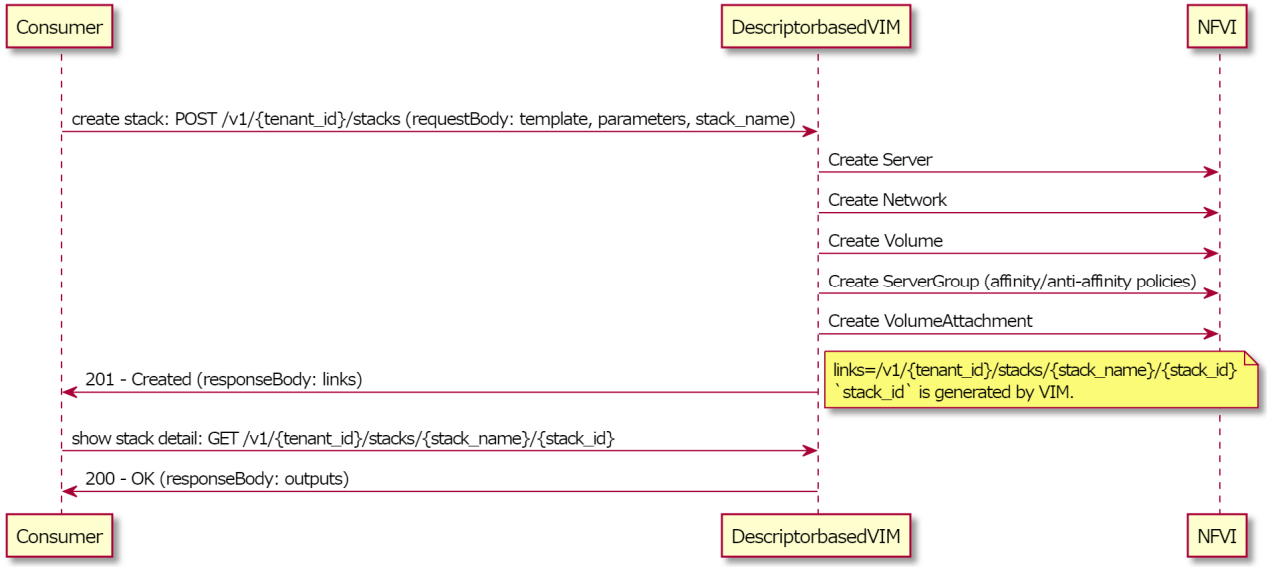


**Figure A.3.6-1**

Figure A.3.6-2 illustrates the sequence diagram of Query Virtualised Compute/Network/Storage Resource operation using "find stack" and "show stack".

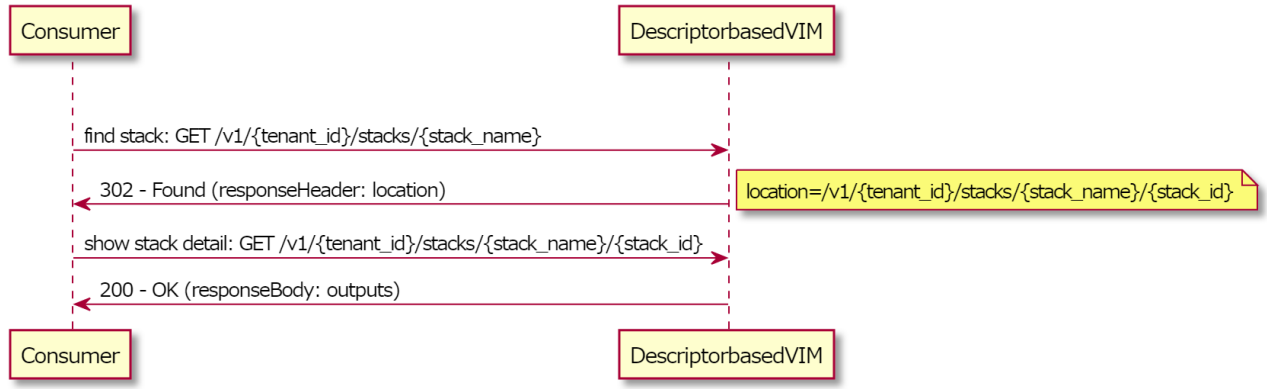


**Figure A.3.6-2**

Figure A.3.6-3 illustrates the sequence diagram of Update Virtualised Compute/Network/Storage Resource operation using "find stack", "update stack" and "show stack".

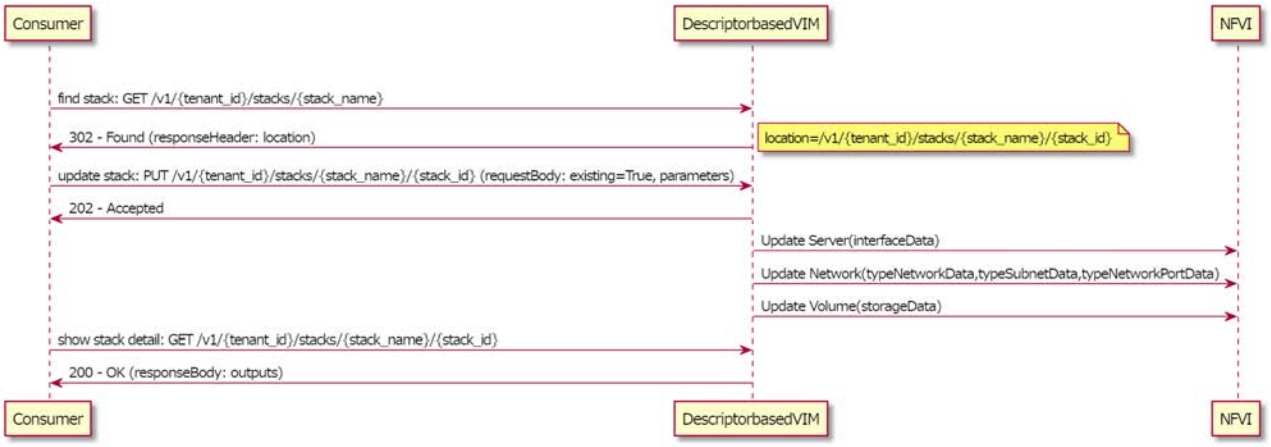


**Figure A.3.6-3**

Figure A.3.6-4 illustrates the sequence diagram of Scale Virtualised Compute/Storage Resource operation using "find stack", "update stack" and "show stack". In the case of Storage, only increase operation is supported.

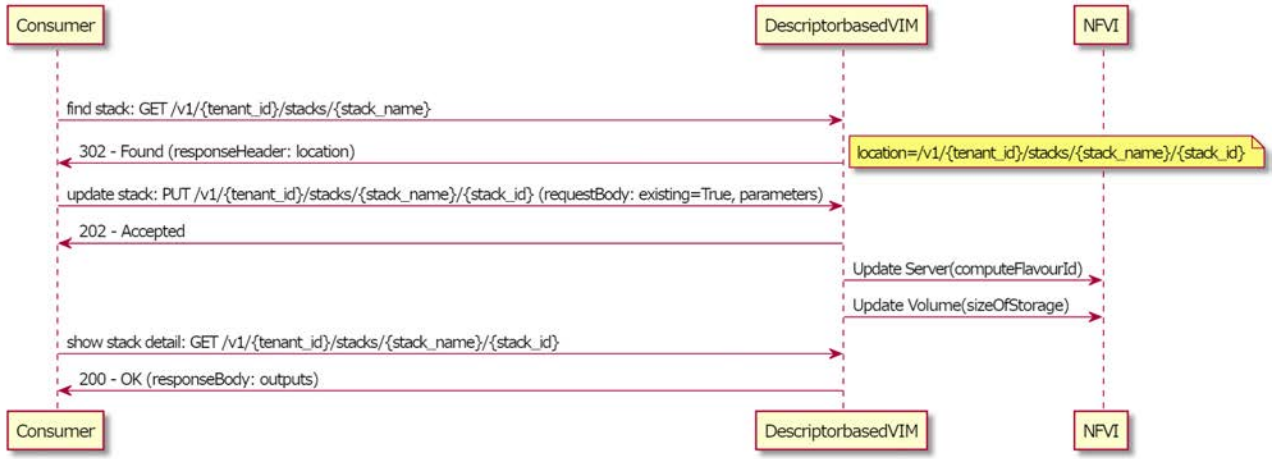


**Figure A.3.6-4**

Figure A.3.6-5 illustrates the sequence diagram of Terminate Virtualised Compute/Network/Storage Resource operation and Detach Virtualised Storage Resource operation using "find stack" and "delete stack". The consumer gets stack_name associated with identifier of the resource using Query Virtualised Compute/Network/Storage Resource operation. This sequence terminates all resources associated with stack_name.

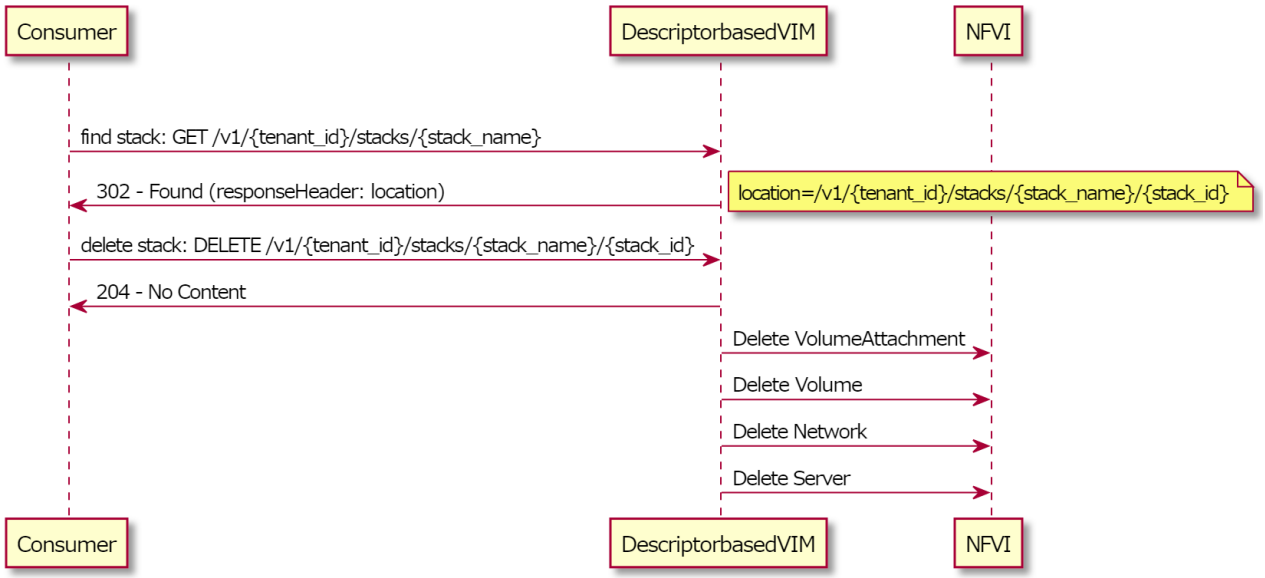


**Figure A.3.6-5**

In the case of Heat, parameter "stackName" in clause 5.2.5 is passed as parameter "stack_name". That is, Consumer manages virtualised resources by generating an identifier that points to a stack of the virtualised resources defined in a descriptor and passing the identifier as input parameter "stack_name".

## A.3.7 Example#7: Virtualised Resources Change Notification Interface Subscribe operation

This is an example of "Create stack" in OpenStack Orchestration Service API [i.3] corresponding to the Subscribe operation in the Change Notification Interface (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 [i.2].

Parameters grouped by "nfv" are specified in the present document. Following the input parameter specifications in the present example (see further below), input parameters callbackUriForChangeNotify, inputFilter 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 Subscribe operation.

```
{
  "parameters": {
    "nfv": {
      "callbackUriForChangeNotify": "http://10.0.0.56",
      "inputFilter": {
        "hostId": "ussuri-compute1"
      }
    }
  },
  "stack_name": "change-notify-compute"
}
```

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 values of the input parameters are taken from input parameters and paired with resource keys in the property section; alarm_actions is paired with "http://10.0.0.56" of callbackUriForChangeNotify, value of traits.host in query is paired with " ussuri-compute1" of hostId of inputFilter.

This HOT is intended to subscribe for change notifications of compute hosts. The conditions for change notifications should be matched to the operation rule for change notifications. This HOT assumes the following operation rule:

When maintenance a compute host, operators allocate a virtualised compute resource on the maintenance target compute host , and this is executed by the VIM user whose user_id is "c794077f19f142819716e3116724ccfe".

```
heat_template_version: 2018-08-31
description: Virtualised Compute Resources Change Notification Interface Subscribe operation.

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

resources:
  maintenance:
    type: OS::Aodh::EventAlarm
    properties:
      event_type: compute.instance.create.end
      alarm_actions:
      - { get_param: [ nfv, callbackUriForChangeNotify ] }
      query:
      - field: traits.host
        op: eq
        value: { get_param: [ nfv, inputFilter, hostId ] }
      - field: traits.user_id
        op: eq
        value: c794077f19f142819716e3116724ccfe
      repeat_actions: true
```

The following is an example of a change notification by `OpenStack` Aodh.

```
{
  "alarm_name": "change-notify-compute-maintenance-dricxhhg53iq",
  "alarm_id": "a76dd860-daad-456c-8ff3-0e7cbed12117",
  "severity": "low",
  "previous": "insufficient data",
  "current": "alarm",
  "reason": "Event <id=1cafb631-0c9a-4fdc-8d3c-d4ff9b0eba43,event_type=compute.instance.create.end>
hits the query <query=[{field: traits.host, op: eq, type: string, value: ussuri-compute1}, {field:
traits.user_id, op: eq, type: string, value: c794077f19f142819716e3116724ccfe}]>.",
  "reason_data": {
    "type": "event",
    "event": {
      "message_id": "1cafb631-0c9a-4fdc-8d3c-d4ff9b0eba43",
```

```
    "event_type": "compute.instance.create.end",
    "generated": "2021-08-19T05:11:35.822867",
    "traits": [
      [
        "service",
        1,
        "compute"
      ],
      [
        "request_id",
        1,
        "req-462d2144-88e7-4307-bbec-cd972a07bfa0"
      ],
      [
        "project_id",
        1,
        "f4453d4fab274ff4a1e9cfe76a82d928"
      ],
      [
        "user_id",
        1,
        "c794077f19f142819716e3116724ccfe"
      ],
      [
        "tenant_id",
        1,
        "f4453d4fab274ff4a1e9cfe76a82d928"
      ],
      [
        "instance_id",
        1,
        "c33af9b4-7ebd-4894-944c-e16880630d5b"
      ],
      [
        "display_name",
        1,
        "vm-for-maintenance"
      ],
      [
        "resource_id",
        1,
        "c33af9b4-7ebd-4894-944c-e16880630d5b"
      ],
      [
        "cell_name",
        1,
        ""
      ],
      [
        "host",
        1,
        "ussuri-compute1"
      ],
      [
        "memory_mb",
        2,
        64
      ],
      [
        "disk_gb",
        2,
        15
      ],
      [
        "root_gb",
        2,
        10
      ],
      [
        "ephemeral_gb",
        2,
        5
      ],
      [
        "vcpus",
        2,
        1
      ],
```

```
        [
          "instance_type_id",
          1,
          "11"
        ],
        [
          "instance_type",
          1,
          "m1.nano"
        ],
        [
          "state",
          1,
          "active"
        ],
        [
          "launched_at",
          4,
          "2021-08-19T05:11:35.661283"
        ],
        [
          "availability_zone",
          1,
          "nova"
        ]
      ],
      "raw": {},
      "message_signature": "86cfec1c900658c76dd0e5efe9ed8e4e074c97d76718c898d36ae43aef269630"
    }
  }
}
```


Table A.3.7-1 illustrates the mapping between the input parameters for change notifications defined in the present document data model and the information in change notifications by OpenStack Aodh.

**Table A.3.7-1: Mapping between the present document change notification parameters and notification from OpenStack Aodh**

| Parameter in the present document | Notification from OpenStack Aodh |
|---|---|
| changeId | reason_data/event/message_id |
| virtualisedResourceId | (not supported by Aodh) |
| virtualisedResourceGroupId | (not supported by Aodh) |
| endOfChange | (not supported by Aodh) |
| changeTime | reason_data/event/generated |
| vimId | (not supported by Aodh) |
| changeType | alarm_name (see note) |
| changedResourceData | (not supported by Aodh) |
| NOTE: Due to current specifications of OpenStack Heat and OpenStack Aodh, "alarm_name" is determined to be a concatenation of "stack_name", "resource name in HOT" and "random string generated by Heat". This is not matched to allowed value of changeType, but like changeType, "alarm_name" is used to distinguish the type of notification. | |

## A.3.8 Example#8: Virtualised Resources Fault Management Interface Subscribe operation

This is an example of "`Create stack`" in OpenStack Orchestration Service API [i.3] corresponding to the Subscribe operation in the Fault Management Interface (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 [i.2].

Parameters grouped by "`nfv`" are specified in the present document. Following the input parameter specifications in the present example (see further below), input parameters `callbackUriForFaultNotify`, filter is 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 Subscribe operation.

```
{
  "parameters": {
    "nfv": {
      "callbackUriForFaultNotify": "http://10.0.0.56",
      "filter": {
        "computeId": "3e5cb24b-044d-4c65-8488-60ebd2c797cc"
      }
    }
  },
  "stack_name": "fault-notify-compute"
}
```

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 values of the input parameters are taken from input parameters and paired with resource keys in the property section; alarm_actions is paired with "http://10.0.0.56" of callbackUriForFaultNotify, value of traits.instance_id in query is paired with "3e5cb24b-044d-4c65-8488-60ebd2c797cc" of computeId of filter.

This HOT is intended to subscribe for fault notifications of virtualised compute resources. The conditions for fault notifications should be matched to the operation when faults were observed. This HOT assumes the following operation:

When the monitoring system observes faults relating a virtualised compute resource, the monitoring system changes the state of the virtualised compute resource to "error" or shutdown the virtualised compute resource.

```
heat_template_version: 2018-08-31
description: Virtualised Compute Resources Fault Management Interface Subscribe operation.

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

resources:
  errorState:
    type: OS::Aodh::EventAlarm
    properties:
      alarm_actions:
      - { get_param: [ nfv, callbackUriForFaultNotify ] }
      query:
      - field: traits.instance_id
        op: eq
        value: { get_param: [ nfv, filter, computeId ] }
      - field: traits.state
        op: eq
        value: error
      repeat_actions: true

  stoppedState:
    type: OS::Aodh::EventAlarm
    properties:
      event_type: compute.instance.power_off.*
      alarm_actions:
      - { get_param: [ nfv, callbackUriForFaultNotify ] }
      query:
      - field: traits.instance_id
        op: eq
        value: { get_param: [ nfv, filter, computeId ] }
      repeat_actions: true
```

The following is an example of a fault notification by OpenStack Aodh.

```
{
  "alarm_name": "fault-notify-compute-stoppedState-bla6fvjo5nrf",
  "alarm_id": "5429876f-c8c4-42e6-a8be-5a5dae68b5c4",
  "severity": "low",
  "previous": "alarm",
  "current": "alarm",
  "reason": "Event <id=4a55319f-dc22-413d-b21a-
34d3728c72c0,event_type=compute.instance.power_off.end> hits the query <query=[{field:
traits.instance_id, op: eq, type: string, value: 3e5cb24b-044d-4c65-8488-60ebd2c797cc}]>.",
```

```
  "reason_data": {
    "type": "event",
    "event": {
      "message_id": "4a55319f-dc22-413d-b21a-34d3728c72c0",
      "event_type": "compute.instance.power_off.end",
      "generated": "2021-08-19T08:12:20.805018",
      "traits": [
        [
          "service",
          1,
          "compute"
        ],
        [
          "request_id",
          1,
          "req-72fd8bda-9444-4e65-8b02-8485cdbb843c"
        ],
        [
          "project_id",
          1,
          "f4453d4fab274ff4a1e9cfe76a82d928"
        ],
        [
          "user_id",
          1,
          "4d75edbddee64505878e5c59c042b70c"
        ],
        [
          "tenant_id",
          1,
          "f4453d4fab274ff4a1e9cfe76a82d928"
        ],
        [
          "instance_id",
          1,
          "3e5cb24b-044d-4c65-8488-60ebd2c797cc"
        ],
        [
          "display_name",
          1,
          "test-vm"
        ],
        [
          "resource_id",
          1,
          "3e5cb24b-044d-4c65-8488-60ebd2c797cc"
        ],
        [
          "cell_name",
          1,
          ""
        ],
        [
          "host",
          1,
          "ussuri-compute2"
        ],
        [
          "memory_mb",
          2,
          64
        ],
        [
          "disk_gb",
          2,
          1
        ],
        [
          "root_gb",
          2,
          1
        ],
        [
          "ephemeral_gb",
          2,
          0
        ],
        [
```

```
          "vcpus",
          2,
          1
        ],
        [
          "instance_type_id",
          1,
          "9"
        ],
        [
          "instance_type",
          1,
          "m1.nano"
        ],
        [
          "state",
          1,
          "stopped"
        ],
        [
          "launched_at",
          4,
          "2021-08-06T01:30:56"
        ]
      ],
      "raw": {},
      "message_signature": "e01fc1dc4fbba2f50168a99da48973dc4964d50bb7ee6c5c5be9f8101a10bd94"
    }
  }
}
```


Table A.3.8-1 illustrates the mapping between the input parameter and attributes for fault notifications defined in the present document data model and the information in fault notifications by OpenStack Aodh.

**Table A.3.8-1: Mapping between the present document fault notification parameters and notification from OpenStack Aodh**

| Attribute in the present document | Notification from OpenStack Aodh |
|---|---|
| alarm | |
| >alarmId | reason_data/event/message_id |
| >managedObjectId | reason_data/event/traits/instance_id |
| >alarmRaisedTime | reason_data/event/generated |
| >alarmChangedTime | (not supported by Aodh) |
| >alarmClearedTime | (not supported by Aodh) |
| >state | (not supported by Aodh) |
| >perceivedSeverity | severity |
| >eventTime | (not supported by Aodh) |
| >eventType | (not supported by Aodh) |
| >faultType | alarm_name (see note) |
| >probableCause | (not supported by Aodh) |
| >isRootCause | (not supported by Aodh) |
| >correlatedAlarmId | (not supported by Aodh) |
| >faultDetails | (not supported by Aodh) |
| NOTE: Due to current specifications of OpenStack Heat and OpenStack Aodh, "alarm_name" is determined to be a concatenation of "stack_name", "resource name in HOT" and "random string generated by Heat". Like faultType, "alarm_name" is used to distinguish the type of notification. | |

## A.3.9 Example#9: Create Compute Resource Reservation operation

This is an example of "Create stack" in OpenStack Orchestration Service API [i.3] corresponding to the Create Compute Resource Reservation operation (ETSI GS NFV-IFA 005 [1]).

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 [i.2].

Parameters grouped by "nfv" are specified in the present document. Following the input parameter specifications in the present example (see further below), input parameters startTime, endTime, computePoolReservation 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 Create Compute Resource Reservation operation.

```
{
  "parameters": {
    "nfv": {
      "startTime": "2022-03-28 06:00",
      "endTime": "2023-03-28 06:00",
      "computePoolReservation": {
        "numCpuCores": 1,
        "numVcInstances": 1,
        "virtualMemSize": 1024
      }
    }
  },
  "stack_name": "compute-reservation-test-stack"
}
```

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 values of the input parameters are taken from input parameters and paired with resource keys in the property section; start_date is paired with "2022-03-28 06:00" of startTime, memory_mb is paired with 1024 of virtualMemSize of computePoolReservation.

The outputs section of the template describes output data for the user, e.g. when in terms of the Create Compute Resource Reservation the nfvComputeReservationInfo 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 "stack_name" is a required parameter to generate an identifier that points to a stack of the virtualised resources, as shown in clause A.3.6.

```
heat_template_version: 2018-08-31
description: Create Compute Resource Reservation operation.

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

resources:
  computeResourceReservation:
    type: OS::Blazar::Lease
    properties:
      name: "computeResourceReservation"
      start_date: { get_param: [ nfv, startTime ] }
      end_date: { get_param: [ nfv, endTime ] }
      reservations:
      - "vcpus": { get_param: [ nfv, computePoolReservation, numCpuCores ] }
        "amount": { get_param: [ nfv, computePoolReservation, numVcInstances ] }
        "memory_mb": { get_param: [ nfv, computePoolReservation, virtualMemSize ] }
        "disk_gb": 0
        "resource_type": "virtual:instance"

outputs:
  nfvComputeReservationInfo:
    value:
      str_replace:
        template: |
          {
            "reservationId": "$reservationId"
          }
        params:
          $reservationId: { get_attr: [ computeResourceReservation, show, reservations, 0, id ] }
```

Below is an output example of the output parameters related to the compute resource reservation resource as provided by "Show output" in OpenStack Orchestration Service API [i.3]. Specified attributes defined in nfvComputeReservationInfo 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": {
      "reservationId": "e0bd181d-0d82-466b-bd26-30dd9c750175"
    },
    "output_key": "nfvComputeReservationInfo",
    "description": "No description given"
  }
}
```

## A.3.10 Example#10: Create Compute Host Reservation operation

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

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 minAmount, maxAmount, startTime, endTime, computeHostProperties 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 Create Compute Host Reservation operation.

```
{
  "parameters": {
    "nfv": {
      "minAmount": "1",
      "maxAmount": "1",
      "startTime": "2023-06-01 00:00",
      "endTime": "2024-06-01 00:00",
      "computeHostProperties": [
        "ssd",
        "gpu"
      ]
    }
  },
  "stack_name": "compute-host-reservation-test-stack"
}
```

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 values of the input parameters are taken from input parameters and paired with resource keys in the property section; start_date is paired with "2023-06-01 00:00" of startTime, computeHostProperty0 is paired with "`ssd`" of the first element of computeHostProperties.

The outputs section of the template describes output data for the user, e.g. when in terms of the Create Compute Host Reservation the nfvComputeHostReservationInfo 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 "`stack_name`" is a required parameter to generate an identifier that points to a stack of the virtualised resources, as shown in clause A.3.6.

```
heat_template_version: 2021-04-16
description: Create Compute Host Reservation operation.

parameters:
```

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

resources:
  computeHostReservation:
    type: OS::Blazar::Lease
    properties:
      name: "computeHostReservation"
      start_date: { get_param: [ nfv, startTime ] }
      end_date: { get_param: [ nfv, endTime ] }
      reservations:
      - "min": { get_param: [ nfv, minAmount ] }
        "max": { get_param: [ nfv, maxAmount ] }
        "resource_properties":
          str_replace:
            template: '[ "and", [ "=", "$computeHostProperty0", "true"], ["=",
"$computeHostProperty1", "true"] ]'
            params:
              computeHostProperty0: { get_param: [ nfv, computeHostProperties, 0 ] }
              computeHostProperty1: { get_param: [ nfv, computeHostProperties, 1 ] }
        "resource_type": "physical:host"

outputs:
  nfvComputeHostReservationInfo:
    value:
      str_replace:
        template: |
          {
            "reservationId": "$reservationId"
          }
        params:
          $reservationId: { get_attr: [ computeHostReservation, show, reservations, 0, id ] }
```

Below is an output example of the output parameters related to the compute host reservation resource as provided by "Show output" in OpenStack Orchestration Service API [i.2]. Specified attributes defined in nfvComputeHostReservationInfo 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": {
            "reservationId": "d090a8af-7e0b-4432-9a54-3646e26c2eff"
        },
        "output_key": "nfvComputeHostReservationInfo",
        "description": "No description given"
    }
}
```

# 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" 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" 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.

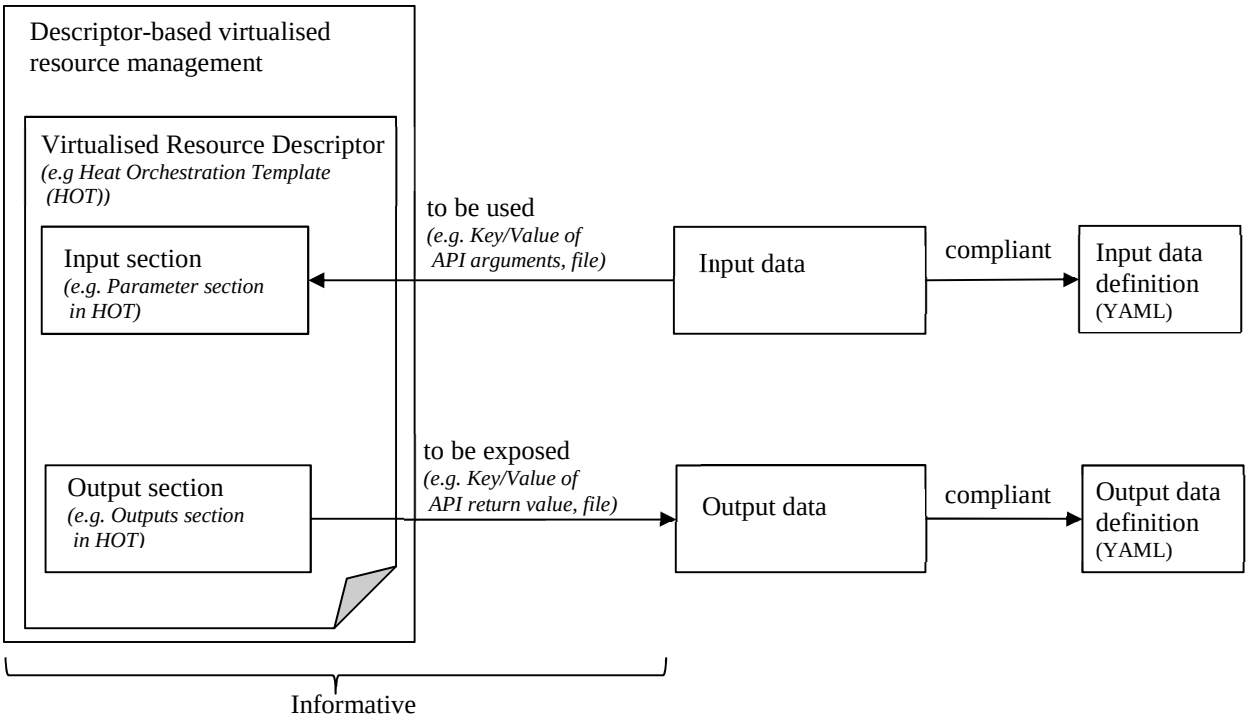


**Figure B.2-1: Concept of descriptor-based virtualised resource management**

# Annex C (informative): Change history

| Date | Version | Information about changes |
|---|---|---|
| February 2021 | 3.0.1 | Initial draft for SOL014ed351 |
| March 2021 | 3.0.2 | Implements<br>- NFVSOL(21)000166 Improvements towards OpenAPI-based data models for general aspects<br>- NFVSOL(21)000160r1 SOL014ed351 Improvements towards OpenAPI-based data models in virtualised storage resource data model<br>- NFVSOL(21)000143r1 SOL014ed351 Improvements towards OpenAPI-based data models in virtualised compute resource data model<br>- NFVSOL(21)000142 SOL014ed Improvements towards OpenAPI-based data models in common data model<br>- NFVSOL(21)000119 SOL014ed351 Improvements towards OpenAPI-based data models in Virtualised Network Management<br>- NFVSOL(21)000059 SOL014 networkResourceName in OpenAPI |
| September 2021 | 3.5.2 | Implements<br>- NFVSOL(21)000452r1 - SOL014ed361 OpenStack Heat API sequence examples and new parameter stackName<br>- NFVSOL(21)000453 - SOL014e361 Example of Change Notification Interface<br>- NFVSOL(21)000462 - SOL014e361 New data models for Change and Fault management<br>- NFVSOL(21)000485 - SOL014ed361 Update Heat template and parameter examples to upgrade version and support required parameter |
| October 2021 | 3.5.3 | Implements<br>- NFVSOL(21)0000486r4 - SOL014ed361 Profiling OpenStack Orchestration API to map IFA operations to resource declarations |
| November 2021 | 3.5.4 | Implements<br>- NFVSOL(21)000571 - SOL014ed361 new parameter userData |
| April 2022 | 4.0.1 | Initial draft for SOL014ed431 based on SOL014ed361<br>Implements<br>- NFVSOL(22)000185 - SOL014e431 mirror of CR 165 New data models for Compute Resource Reservation management |
| June 2022 | 4.0.2 | Implements<br>- NFVSOL(22)000284 - SOL014ed431-Changing the term "inline" to "inlined" |
| January 2023 | 4.3.2 | Initial draft for SOL014ed441 based on the published version 4.3.1 of ETSI GS NFV-SOL 014 |
| January 2023 | 4.3.3 | Update RFC reference according to the new format |
| July 2023 | 4.4.2 | Implements<br>- NFVSOL(23)000196 - SOL014e451 New data models for Compute Host Reservation management |
| September 2023 | 4.4.3 | Implements<br>- NFVSOL(23)000286 - Enh02.01 SOL014ed451 Add Routing Resource<br>- NFVSOL(23)000292 - Enh02.03 SOL014ed451 Add Data Flow Mirroring Job |
| October 2023 | 4.4.4 | Implements<br>- NFVSOL(23)000266 - SOL014e451 New data models for Compute Host Capacity Management |
| October 2023 | 4.4.5 | Implements<br>- NFVSOL(23)000218r1 - Enh01.01 SOL014ed451 Add certificateData |
| February 2024 | 4.5.2 | Implements<br>- NFVSOL(23)000392r1 - SOL014ed461 FEAT28 add reference to IFA045 (Mirror of 350) |
| April 2024 | 5.0.1 | Initial draft for SOL014ed511 based on published version v4.5.1 |
| April 2024 | 5.0.2 | Implements<br>- NFVSOL(24)000036 - SOL014ed511 Rel.5 mirror of (23)392r1 |
| July 2024 | 5.1.2 | Initial draft for SOL014ed521 based on published version v5.1.1 |
| October 2024 | 5.0.2 | Implements<br>- NFVSOL(24)000349 - FEAT29 SOL014ed521 Enhancement Compute Resource reservation<br>- NFVSOL(24)000365 - SOL014ed521 Clarification about resource zones |
| June 2025 | 5.2.2 | Initial draft for SOL014ed531 based on published version v5.2.1 |
| June 2025 | 5.2.3 | Implements<br>- NFVSOL(25)000140 - FEAT29 SOL014ed531 GreenNFV interface enhancements<br>- Rapporteur change to align the text in clause 2.2 as per the new ETSI GS skeleton |

# History

| Document history | | |
|---|---|---|
| V5.1.1 | July 2024 | Publication |
| V5.2.1 | December 2024 | Publication |
| V5.3.1 | August 2025 | Publication |
| | | |
| | | |