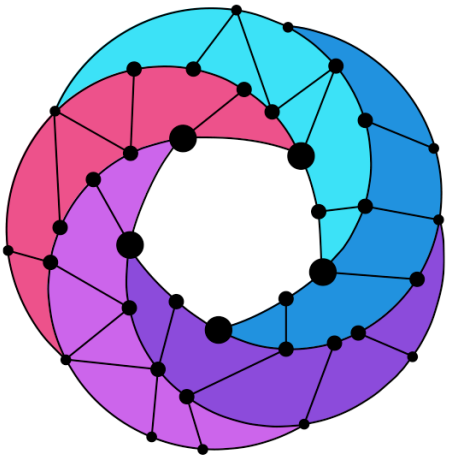

Open Source MANO

SO REST API (OSM RELEASE ONE)

March 1, 2017

CONTENTS

REST API—Introduction

This guide describes the client interface of the REST Application Programming Interface. This guide also provides examples on how to communicate with the REST server.

Terminology

- anyxml: A data node that can contain an unknown chunk of XML data.
- augment: Adds new schema nodes to a previously defined schema node. augment is useful for adding vendor-specific parameters to standard data models.
- base type: The type from which a derived type was derived, which may be either a built-in type or another derived type.
- built-in type: A YANG data type defined in the YANG language, such as uint32 or string.
- choice: A schema node where only one of a number of identified alternatives is valid.
- configuration data: The set of writable data that is required to transform a system from its initial default state into its current state [RFC4741].
- conformance: A measure of how accurately a device follows a data model.
- container: An interior data node that exists in at most one instance in the data tree. A container has no value, but rather a set of child nodes.
- data definition statement: A statement that defines new data nodes. One of container, leaf, leaf-list, list, choice, case, augment, uses, and anyxml.
- data model: A data model describes how data is represented and accessed.
- data node: A node in the schema tree that can be instantiated in a data tree. One of container, leaf, leaf-list, list, and anyxml.
- data tree: The instantiated tree of configuration and state data on a device.
- derived type: A type that is derived from a built-in type (such as uint32), or another derived type.
- device deviation: A failure of the device to implement the module faithfully.
- extension: An extension attaches non-YANG semantics to statements. The extension statement defines new statements to express these semantics.
- feature: A mechanism for marking a portion of the model as optional. Definitions can be tagged with a feature name and are only valid on devices that support that feature.
- grouping: A reusable set of schema nodes, which may be used locally in the module, in modules that include it, and by other modules that import from it. The grouping statement is not a data definition statement and, as such, does not define any nodes in the schema tree.
- identifier: Used to identify different kinds of YANG items by name.
- instance identifier: A mechanism for identifying a particular node in a data tree.
- interior node: Nodes within a hierarchy that are not leaf nodes.
- leaf: A data node that exists in at most one instance in the data tree. A leaf has a value but no child nodes.

- leaf-list: Like the leaf node but defines a set of uniquely identifiable nodes rather than a single node. Each node has a value but no child nodes.
- leaf node: Contains simple data like an integer or a string. It has exactly one value of a particular type and no child nodes.
- list: An interior data node that may exist in multiple instances in the data tree. A list has no value, but rather a set of child nodes.
- module: A YANG module defines a hierarchy of nodes that can be used for NETCONF-based operations. With its definitions and the definitions it imports or includes from elsewhere, a module is self-contained and "compilable".
- RPC: A Remote Procedure Call, as used within the NETCONF protocol.
- RPC operation: A specific Remote Procedure Call, as used within the NETCONF protocol. It is also called a protocol operation.
- schema node: A node in the schema tree. One of container, leaf, leaf-list, list, choice, case, rpc, input, output, notification, and anyxml.
- schema node identifier: A mechanism for identifying a particular node in the schema tree.
- schema tree: The definition hierarchy specified within a module.
- state data: The additional data on a system that is not configuration data such as read-only status information and collected statistics [RFC4741].
- submodule: A partial module definition that contributes derived types, groupings, data nodes, RPCs, and notifications to a module. A YANG module can be constructed from a number of submodules.
- top-level data node: A data node where there is no other data node between it and a module or submodule statement.
- uses: The "uses" statement is used to instantiate the set of schema nodes defined in a grouping statement. The instantiated nodes may be refined and augmented to tailor them to any specific needs.

Resource: YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)

Normal Operations

URL targets

RW.REST URL targets can be one of the following options:

URL Target	Description
`/api/config`	Targets the config data store.
`/api/operational`	Targets operational data.
`/api/operations`	Targets remote procedure calls (RPCs).
`/api/schema`	Convert YANG node described by the URL path into its JSON representation

The subsequent parts of the URL are used to select the YANG object. For example, to associate a cloud account with your Orchestrator instance, append target `/api/config` with the YANG object `cloud`:

```
https://<orchestrator_ip/fqdn>:8008/api/config/cloud
```

Authentication

All requests to the REST API require you to authenticate.

Provide a username/password pair encoded using HTTP basic auth access in the request header.

Query parameters

Optionally use the following query parameters with GET operations to filter the results of a RW.REST API request:

- `deep` – Use to select an entire sub-tree. To retrieve all elements, use the deep query parameter.

  For example, GET a top-level container with `?deep`:

  ```
  /api/running/top-container-deep?deep
  ```

- `select` – Use to select specific leaves of a list or container.

Mime types

The API supports the following MIME types:

MIME Type	Description
`application/vnd.yang.data+json`	Data being given or received is encoded in JSON.
`application/vnd.yang.collection+json`	Data being given or received is encoded in JSON, with a special element aggregating the top-level list objects.
`application/vnd.yang.data+xml`	Data being given or received is encoded in XML.
`application/vnd.yang.collection+xml`	Data being given or received is encoded in XML, with a special element aggregating the top-level list objects.

Example request header

```
GET /api/running/misc/int-leaf HTTP/1.1
 Host: example.example.com
 Accept: application/yang.api+xml
```

Operations

The API supports the following operations:

Operation	Description
`GET`	Retrieves data. GET is equivalent to a NETCONF `get`, or `get_config` operation, depending on the target. Note: When you perform GET operations on containers, the results are pruned if the container is a top-level object in its module. The pruning removes all non-key leafs from the result.

Operation	Description
`POST`	Creates new data. POST is equivalent to a NETCONF `create` operation, or if the target is an RPC, they are a RPC dispatch.
`PUT`	Updates data. PUT is equivalent to a NETCONF `replace` operation.
`DELETE`	Deletes data. DELETE is equivalent to a NETCONF `delete` operation.

See also

Netconf Central Documentation and RPC Methods

REST Examples

This topic provides a sample YANG scheme and examples for common calls to the REST API.

REST YANG Model

The YANG used for the examples in this guide is as follows.

Module example

```
{
   namespace "http://example.com/ns/example";
   prefix "example";

   container top-container-shallow {
     leaf a {
       type string;
     }
   }

   list top-list-shallow {
     key "k";
     leaf k {
       type string;
     }
   }

   container top-container-deep {
     leaf a {
       type string;
     }

     list inner-list-shallow {
       key "k";
       leaf k {
         type string;
       }
     }

     container inner-container {
       leaf a{
         type string;
       }
       list inner-list {
         key "k";
         leaf k {
           type string;
         }
       }

     }

     list inner-list-deep {
       key "k";
```

```
      leaf k {
        type string;
      }
      container inner-container-shallow {
        leaf a {
          type string;
        }
      }
      container inner-container-deep {
        list bottom-list-shallow {
          key "k";
          leaf k {
            type string;
          }
        }
      }
    }

  }

  list top-list-deep {
      key "k";
      leaf k {
        type string;
      }
      list inner-list {
        key "k";
        leaf k {
          type string;
        }
        leaf a {
          type string;
        }
        container inner-container {
          leaf a{
            type string;
          }
        }
      }

      container inner-container-shallow {
        leaf a{
          type string;
        }
      }
      container inner-container-deep {
        list bottom-list-shallow {
          key "k";
          leaf k {
            type string;
          }
        }
      }
    }

  list multi-key {
```

```
    key "foo bar";
    leaf foo {
      type string;
    }
    leaf bar {
      type string;
    }

    leaf treasure {
      type uint32;
    }

  }

  rpc in-and-out {
    input {
      leaf in {
        type string;
      }
    }
    output {
      leaf out {
        type string;
      }
    }
  }

  rpc in-no-out {
    input {
      leaf in {
        type string;
      }
    }
  }

  identity identity-base {
    description "Testing base identity";
  }

  identity identity-sub {
    base "identity-base";
  }

  container misc {

    leaf binary-leaf {
      type binary {
        length "1..20";
      }
    }

    leaf bool-leaf {
      type boolean;
    }

    leaf decimal-leaf {
```

```
      type decimal64 {
        fraction-digits 2;
      }
    }

    leaf empty-leaf {
      type empty;
    }

    leaf enum-leaf {
      type enumeration {
        enum a;
        enum b;
      }
    }

    leaf identityref-leaf {
      type identityref {
        base "identity-base";
      }
    }

    leaf int-leaf {
      type int32;
    }

    leaf instance-identifier-leaf {
      type instance-identifier;
    }

    list list-a {
      key "id";
      leaf id {
        type uint8;
      }
      leaf foo {
        type string;
      }
    }

    list list-b {
      key "id";
      leaf id {
        type uint8;
      }
      leaf leafref-to-list-a {
        type leafref {
          path "../../list-a[id=current()/../id]/foo";
        }
      }
    }
  }
}
```

GET examples

GET a leaf

URL

```
/api/running/misc/int-leaf
```

JSON response

```
{
   "example:int-leaf": 42
 }
```

XML response

```
<misc xmlns="http://example.com/ns/example"
         xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
     <int-leaf>42</int-leaf>
 </misc>
```

GET a top-level container

URL

```
/api/running/top-container-deep
```

JSON response

```
{
   "example:top-container-deep": {
     "inner-container": {
       "inner-list": [
         {
           "k": "another key thing"
         }
       ]
     }
   }
 }
```

XML response

```
<top-container-deep xmlns="http://example.com/ns/example"
         xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
     <inner-container>
         <inner-list>
             <k>another key thing</k>
         </inner-list>
     </inner-container>
 </top-container-deep>
```

GET a top-level container with ?deep

URL

```
/api/running/top-container-deep?deep
```

JSON response

```
{
   "example:top-container-deep": {
     "inner-container": {
       "inner-list": [
         {
           "k": "another key thing"
         }
       ],
       "a": "another string"
     }
   }
 }
```

XML response

```
<top-container-deep xmlns="http://example.com/ns/example"
         xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
     <inner-container>
         <a>another string</a>
         <inner-list>
             <k>another key thing</k>
         </inner-list>
     </inner-container>
 </top-container-deep>
```

GET a container

URL

```
/api/running/top-container-deep/inner-container
```

JSON response

```
{
   "inner-list": [
     {
       "k": "another key thing"
     }
   ],
   "a": "another string"
 }
```

XML response

```
<top-container-deep xmlns="http://example.com/ns/example"
         xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
     <inner-container>
```

```
        <a>another string</a>
        <inner-list>
            <k>another key thing</k>
        </inner-list>
    </inner-container>
</top-container-deep>
```

GET a whole list

URL

```
/api/running/top-list-shallow
```

JSON response

```
{
   "example:top-list-shallow": [
     {
       "k": "asdf"
     },
     {
       "k": "fdsa"
     }
   ]
 }
```

JSON collection response

```
{
   "collection": {
     "example:top-list-shallow": [
       {
         "k": "asdf"
       },
       {
         "k": "fdsa"
       }
     ]
   }
 }
```

XML response

```
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
        xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <top-list-shallow xmlns="http://example.com/ns/example">
        <k>asdf</k>
    </top-list-shallow>
    <top-list-shallow xmlns="http://example.com/ns/example">
        <k>fdsa</k>
    </top-list-shallow>
</data>
```

XML collection response

```
<collection>
     <top-list-shallow xmlns="http://example.com/ns/example"
         xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <k>asdf</k>
     </top-list-shallow>
     <top-list-shallow xmlns="http://example.com/ns/example"
         xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <k>fdsa</k>
     </top-list-shallow>
 </collection>
```

GET a specific list entry

URL

```
/api/running/top-list-shallow/asdf
```

JSON response

```
{
   "example:top-list-shallow": [
     {
       "k": "asdf"
     }
   ]
 }
```

JSON collection response

```
{
   "collection": {
     "example:top-list-shallow": [
       {
         "k": "asdf"
       }
     ]
   }
 }
```

XML response

```
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
         xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
     <top-list-shallow xmlns="http://example.com/ns/example">
         <k>asdf</k>
     </top-list-shallow>
 </data>
```

XML collection response

```
<collection>
     <k xmlns="http://example.com/ns/example"
```

```
     xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">asdf</k>
 </collection>
```

POST/PUT examples

POST/PUT a leaf

URL

```
/api/running/top-container-shallow
```

JSON body

```
{
     "a" : "some leaf 0"
 }
```

XML body

```
<example:top-container-shallow xmlns:example="http://example.com/ns/example">
   <a xmlns="http://example.com/ns/example">some leaf 0</a>
 </example:top-container-shallow>
```

POST/PUT a container

URL

```
/api/config/top-container-deep/inner-container
```

JSON body

```
{
     "a":"another string",
     "inner-list":[
         {
             "k":"another key thing"
         }
     ]
 }
```

XML body

```
<example:inner-container xmlns:example="http://example.com/ns/example">
   <a xmlns="http://example.com/ns/example">another string</a>
   <inner-list >
     <k xmlns="http://example.com/ns/example">another key thing</k>
   </inner-list>
 </example:inner-container>
```

POST/PUT a list entry

URL

```
/api/running/top-list-shallow/asdf
```

JSON body

```
{
   "top-list-shallow"  :
   [
     {
       "k" : "asdf"
     }
   ]
 }
```

XML body

```
<top-list-shallow xmlns="http://example.com/ns/example">
    <k xmlns="http://example.com/ns/example">asdf</k>
 </top-list-shallow>
```

POST an RPC

URL

```
/api/operations/in-no-out
```

JSON body

```
{
   "input" : {
     "in" : "asdf"
   }
 }
```

XML body

```
<input>
   <in="http://example.com/ns/example">asdf</in>
 </input>
```

DELETE examples

The DELETE operation has the same URL structure as the GET operation.

Note: You cannot perform a DELETE operation on containers.

Additional JSON POST Examples

URL

```
/api/config/top-container-shallow
```

JSON response

```
{
   "a" : "some leaf 0"
 }
```

URL

```
/api/config/top-list-shallow/some%20key
```

JSON response

```
{
   "top-list-shallow"  :
   [
     {
       "k" : "some key"
     }
   ]
 }
```

URL

```
/api/config/top-container-deep
```

JSON response

```
{
     "a":"some kind of string",
     "inner-list-shallow":[
         {
             "k":"some key thing"
         },
         {
             "k":"some other key thing"
         }
     ],
     "inner-container":{
         "a":"another string",
         "inner-list":[
             {
                 "k":"another key thing"
             }
         ]
     },
     "inner-list-deep":[
         {
```

```
            "k":"inner key",
            "inner-container-shallow":{
                "a":"an inner string"
            },
            "inner-container-deep":{
                "bottom-list-shallow":[
                    {
                        "k":"bottom key"
                    }
                ]
            }
        }
    ]
}
```

URL

```
/api/config/top-list-deep
```

JSON response

```
{
   "top-list-deep" :
   [
     {
       "k" : "some key",
       "inner-list" :
       [
         {
        "k" : "some other key",
        "a" : "some string",
        "inner-container" : {
          "a" : "some other string"
        }
      }
       ],
       "inner-container-shallow" :
       {
         "a" : "yet a third string"
       },
       "inner-container-deep" :
       {
         "bottom-list-shallow" :
      [
        {
        "k" : "yet a third key"
        }
      ]
       }
     }
   ]
}
```

URL

```
/api/config/multi-key/foo,bar
```

JSON response

```
{
   "multi-key" :
   [
     {
       "foo" : "key part 1",
       "bar" : "key part 2",
       "treasure" : 32
     }
   ]
 }
```

URL

```
/api/config/top-list-deep/key1/inner-list/some%20key%20thing
```

JSON response

```
{
     "inner-list-shallow":[
         {
             "k":"some key thing"
         }
     ]
 }
```

URL

```
/api/config/top-container-deep/inner-list-shallow/some%20key
```

JSON response

```
{
     "inner-list-shallow":[
         {
             "k":"some key"
         }
     ]
 }
```

URL

```
/api/config/top-list-deep/key1/inner-list/key2
```

JSON response

```
{
     "inner-list":[
         {
             "k":"key2"
         }
     ]
 }
```

URL

```
/api/config/top-container-deep/inner-container
```

JSON response

```
{
     "a":"another string",
     "inner-list":[
         {
             "k":"another key thing"
         }
     ]
 }
```

URL

```
/api/config/top-list-deep/some%20key/inner-container-shallow
```

JSON response

```
{
     "a" : "yet a third string"
 }
```

URL

```
/api/config/misc
```

JSON response

```
{
     "bool-leaf":true,
     "empty-leaf":[
         null
     ],
     "enum-leaf":"a",
     "int-leaf":42,
     "list-a":[
         {
             "id":0,
             "foo":"asdf"
         }
     ],
```

```
    "list-b":[
        {
            "id":0
        }
    ],
    "numbers":[
        {
            "int8-leaf":0,
            "int16-leaf":0,
            "int32-leaf":0,
            "int64-leaf":0,
            "uint8-leaf":0,
            "uint16-leaf":0,
            "uint32-leaf":0,
            "uint64-leaf":0,
            "decimal-leaf":0
        },
        {
            "int8-leaf":"1",
            "int16-leaf":"0",
            "int32-leaf":"0",
            "int64-leaf":"0",
            "uint8-leaf":"0",
            "uint16-leaf":"0",
            "uint32-leaf":"0",
            "uint64-leaf":"0",
            "decimal-leaf":"0"
        }
    ]
}
```

Event Streams and Notifications

This guide is for developers who want to implement and consume NETCONF notifications.

NETCONF notifications are asynchronous, one-way message notifications that originate from the NETCONF server. Clients can subscribe to event-streams defined in the server using a subscription mechanism.

By default, the Management Agent (**RW.MgmtAgent**) implements a `uagent_notification` notification stream.

An event notification can be generated by a tasklet using the Distributed Transaction System (RW.DTS). RW.MgmtAgent listens for all event notifications generated by RW.DTS and publishes them to Confd. Confd then publishes the event through `uagent_notification` NETCONF event stream. The published notification can be consumed by a web application through RW.REST using either a Websockets or HTTP event source.

Note: The REST notification is implemented based on the RESTCONF Protocol draft-ietf-netconf-restconf-09. See also the Internet Engineering Task Force (IETF) YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF) RFC6020.

The sections that follow describe an HTTP-based programmatic interface for accessing data defined in YANG, using the data stores defined in NETCONF.

Terminology:

- **Element** — XML Element.
- **Subscription** — An agreement and method to receive event notifications over a NETCONF session. A concept related to the delivery of notifications (if there are any to send) involving destination and selection of notifications. A subscription is bound to the lifetime of a session.
- **Operation** — Refers to NETCONF protocol operations defined in support of NETCONF notifications.
- **Event** — Something that occurs that might be of interest, such as a configuration change, a fault, a change in status, crossing a threshold, or an external input to the system. An event often results in an asynchronous message, sometimes referred to as a notification or event notification. The message is sent to interested parties to notify them that this event has occurred.
- **Replay** — The ability to send/re-send previously-logged notifications, upon request. These notifications are sent asynchronously. This feature is implemented by the NETCONF server and invoked by the NETCONF client.
- **Stream** — A set of event notifications that matches some forwarding criteria and available to NETCONF clients for subscription.

- **Filter** — A parameter that indicates which subset of all possible events are of interest. A filter is defined as one or more filter elements [NETCONF], each of which identifies a portion of the overall filter.

State

State is operational data provided by RW.RESTCONF. The schema `ietf-restconf-monitoring` for this operational data is defined in the RESTCONF Protocol draft-ietf-netconf-restconf-09.

GET operations that use `/api/operational/restconf-state` can be used to retrieve the streams and their properties.

GET streams

URL

```
/api/operational/restconf-state/streams
```

Each stream can be accessed using the following methods.

Stream	Method
WebSocket access, streaming using XML messages	[wss://host:port/ws_streams/STREAM-NAME]
WebSocket access, streaming using JSON messages	[wss://host:port/ws_streams/STREAM-NAME-JSON]
HTTP event source access, streaming using XML data	[https://host:port/streams/STREAM-NAME]
HTTP event source access, streaming using JSON data	[https://host:port/streams/STREAM-NAME-JSON]

JSON body example

```
{
   "ietf-restconf-monitoring:streams" :{
     "stream" : [
       {
         "access" : [
           {
             "encoding" : "ws_xml",
             "location" : "wss://10.0.1.7:8888/ws_streams/NETCONF"
           },
           {
             "encoding" : "ws_json",
```

```
            "location" : "wss://10.0.1.7:8888/ws_streams/NETCONF-JSON"
          },
          {
            "encoding" : "xml",
            "location" : "https://10.0.1.7:8888/streams/NETCONF"
          },
          {
            "encoding" : "json",
            "location" : "https://10.0.1.7:8888/streams/NETCONF-JSON"
          }
        ],
        "description" : "default NETCONF event stream",
        "replay-support" : "false",
        "name" : "NETCONF"
      },
      {
        "access" : [
          {
            "encoding" : "ws_xml",
            "location" :
"wss://10.0.1.7:8888/ws_streams/uagent_notification"
          },
          {
            "encoding" : "ws_json",
            "location" :
"wss://10.0.1.7:8888/ws_streams/uagent_notification-JSON"
          },
          {
            "encoding" : "xml",
            "location" : "https://10.0.1.7:8888/streams/uagent_notification"
          },
          {
            "encoding" : "json",
            "location" : "https://10.0.1.7:8888/streams/uagent_notification-
JSON"
          }
        ],
        "description" : "RW Uagent notifications",
        "replay-support" : "true",
        "replay-log-creation-time" : "2016-03-02T10:02:42+00:00",
        "name" : "uagent_notification"
      }
    ]
  }
 }
```

XML body example

```
<data>
     <restconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-
monitoring">
       <streams>
         <stream>
           <name>NETCONF</name>
           <description>default NETCONF event stream</description>
           <replay-support>false</replay-support>
```

```
            <access>
              <encoding>ws_xml</encoding>
              <location>wss://10.0.1.7:8888/ws_streams/NETCONF</location>
            </access>
            <access>
              <encoding>ws_json</encoding>
              <location>wss://10.0.1.7:8888/ws_streams/NETCONF-JSON</location>
            </access>
            <access>
              <encoding>xml</encoding>
              <location>https://10.0.1.7:8888/streams/NETCONF</location>
            </access>
            <access>
              <encoding>json</encoding>
              <location>https://10.0.1.7:8888/streams/NETCONF-JSON</location>
            </access>
          </stream>
          <stream>
            <name>uagent_notification</name>
            <description>RW Uagent notifications</description>
            <replay-support>true</replay-support>
            <replay-log-creation-time>2016-03-02T10:02:42+00:00</replay-log-
creation-time>
            <access>
              <encoding>ws_xml</encoding>

<location>wss://10.0.1.7:8888/ws_streams/uagent_notification</location>
            </access>
            <access>
              <encoding>ws_json</encoding>
              <location>wss://10.0.1.7:8888/ws_streams/uagent_notification-
JSON</location>
            </access>
            <access>
              <encoding>xml</encoding>

<location>https://10.0.1.7:8888/streams/uagent_notification</location>
            </access>
            <access>
              <encoding>json</encoding>
              <location>https://10.0.1.7:8888/streams/uagent_notification-
JSON</location>
            </access>
          </stream>
        </streams>
      </restconf-state>
 </data>
```

GET stream location

URL

```
/api/operational/restconf-state/streams/stream/{stream-
name}/access/{encoding}/location
```

Encoding can be one of [xml, json, ws_xml, ws_json]

URL example

```
/api/operational/restconf-
state/streams/stream/NETCONF/access/ws_json/location
```

JSON body example

```
{"ietf-restconf-monitoring:location" :
"wss://10.0.1.7:8888/ws_streams/NETCONF-JSON"}
```

XML body example

```
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <restconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-
monitoring">
     <streams>
       <stream>
         <name>NETCONF</name>
         <access>
           <encoding>ws_json</encoding>
           <location>wss://10.0.1.7:8888/ws_streams/NETCONF-JSON</location>
         </access>
       </stream>
     </streams>
   </restconf-state>
 </data>
```

Event Source

The REST interface serves as an event source for NETCONF notifications.

Query parameters

The following query parameters are optional in the GET URL for both WebSocket and HTTP calls.

Query parameter	Description
`filter`	Event content filter (based on XPath 1.0)
`start-time`	Replay event start time
`stop-time`	Replay event stop time

WebSocket event source

In a WebSocket event source, streaming occurs through WebSocket transport.

URL format

Opening the WebSocket with the following URL format enables subscription to events for the given stream.

```
wss://HOST:PORT/ws_streams/STREAM-NAME[?QUERY-PARAMS]
```

If the *STREAM-NAME* value contains a `JSON` suffix (*STREAM-NAME-JSON*), notifications are reported in JSON format. Otherwise, notifications are reported in XML format.

JSON event format

```
{
   "notification": {
   "eventTime" : "YYYY-MM-DDThh:mm:ss.microsec[TZ-Offset]",
   "...." : {...}
 }
```

XML event format

```
<notification>
   <eventTime>YYYY-MM-DDThh:mm:ss.microsec[TZ-Offset]</eventTime>
```

```
   ... ...
 </notification>
```

JavaScript example

```
var client = new WebSocket('wss://localhost:8888/ws_streams/NETCONF-JSON');

 client.onerror = function() {
     alert('Connection Error');
 };

 client.onopen = function() {
     alert('WebSocket Client Connected');
 };

 client.onclose = function() {
     alert('Websocket Client Closed');
 };

 client.onmessage = function(e) {
     if (typeof e.data === 'string') {
         alert("Received: '" + e.data + "'");
     }
 };
```

JSON stream example

The following notification is reported when a configuration is changed.

```
{
   "notification" : {
     "eventTime" : "2016-03-02T05:49:51.986098-05:00",
     "ietf-netconf-notifications:netconf-config-change" : {
       "datastore" : "running",
       "changed-by":{
         "username" : "admin",
         "source-host" : "127.0.0.1",
         "session-id" : 11
       },
       "edit" : [
         {
           "operation" : "replace",
           "target" : "/rwlog-mgmt:logging/rwlog-mgmt:console/rwlog-
mgmt:filter/rwlog-mgmt:category[rwlog-mgmt:name='rw-mgmtagt-log']/rwlog-
mgmt:severity"
         }
       ]
     }
   }
 }
```

XML stream example

The following notification is reported when a configuration is changed.

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2016-03-02T05:49:51.986098-05:00</eventTime>
```

```
   <netconf-config-change xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-
notifications">
     <changed-by>
       <username>admin</username>
       <session-id>11</session-id>
       <source-host>127.0.0.1</source-host>
     </changed-by>
     <datastore>running</datastore>
     <edit>
       <target xmlns:rwlog-mgmt="http://example.com/ns/riftware-1.0/rwlog-
mgmt">/rwlog-mgmt:logging/rwlog-mgmt:console/rwlog-mgmt:filter/rwlog-
mgmt:category[rwlog-mgmt:name='rw-mgmtagt-log']/rwlog-mgmt:severity</target>
       <operation>replace</operation>
     </edit>
   </netconf-config-change>
 </notification>
```

HTTP event source

The HTTP event source is based on based on W3C.CR-eventsource.

Run HTTP GET using the following URL format to subscribe to events for the given stream.

URL format

```
https://HOST:PORT/streams/STREAM-NAME[?QUERY-PARAMS]
```

If the *STREAM-NAME* value contains a `JSON` suffix (*STREAM-NAME-JSON*), notifications are reported in JSON format. Otherwise, notifications are reported in XML format.

GET request headers

The following headers specify that the accepted content is an event stream and the connection must be persisted.

```
Accept: text/event-stream
 Cache-Control: no-cache
 Connection: keep-alive
```

In both JSON and XML event formats, the `data:` tag appears for each line in the response.

JSON event format

```
data: {
 data: "notification": {
 data:    "eventTime" : "YYYY-MM-DDThh:mm:ss.microsec[TZ-Offset]",
 data:    "...." : {...}
 data: }
```

XML event format

```
data: <notification>
 data:    <eventTime>YYYY-MM-DDThh:mm:ss.microsec[TZ-Offset]</eventTime>
 data:    ... ...
 data: </notification>
```

YANG Node to JSON Conversion

This section describes how to use the RW.REST API to convert a YANG node, described by the URL path, into its JSON representation.

Note: Conversion works only for CONFIG nodes. All other node types are ignored by rwyanglib.

URL target

Use the following target to convert a YANG node to JSON.

```
/api/schema/....
```

The following table describes the JSON representation of a particular node.

URL path	Description
`/api/schema/X/Y/Z`	Returns the JSON representation of YANG node Z
`/api/schema`	Returns the JSON representation of all the top-level nodes of the schema

General examples

leaf

```
/api/schema/nd:nsd-catalog/nsd:nsd/nsd:id

 {
 "id": {
   "name": "id",
   "type": "leaf",
   "description": "Identifier for the NSD.",
   "cardinality": "0..1",
   "data-type": "string"
   }
 }
```

list

```
/api/schema/nsd:nsd-catalog/nsd:nsd/connection-point

 {
 "connection-point": {
   "name": "connection-point",
```

```
   "type": "list",
   "description": "...." // Not printing for doc brevity
   "cardinality": "0..N",
   "properties": [
     {
       "name": "name",
       "type": "leaf",
       "description": "Name of the NS connection point.",
       "cardinality": "0..1",
       "data-type": "string"
     },
     {
       "name": "type",
       "type": "leaf",
       "description": "Type of the connection point.",
       "cardinality": "0..1",
       "data-type": {
         "enumeration": {
           "enum": {
             "VPORT": {
               "value": 0
             }
           }
         }
       }
     }
   ]
 }
 }
```

leafref

```
/api/schema/nsd:nsd-catalog/nsd:nsd/constituent-vnfd/vnfd-id-ref

 {
 "vnfd-id-ref": {
   "name": "vnfd-id-ref",
   "type": "leaf",
   "description": "Identifier for the VNFD.",
   "cardinality": "0..1",
   "data-type": {
     "leafref": {
       "path": "/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id"
     }
   }
 }
 }
```

enumeration

```
/api/schema/nd:nsd-catalog/nsd:nsd/monitoring-param/json-query-method

 {
 "json-query-method": {
   "name": "json-query-method",
   "type": "leaf",
```

```
   "description": "",
   "cardinality": "0..1",
   "data-type": {
     "enumeration": {
       "enum": {
         "NAMEKEY": {
           "value": 0
         },
         "JSONPATH": {
           "value": 1
         },
         "OBJECTPATH": {
           "value": 2
         }
       }
     }
   }
 }
 }
```

Detailed mapping examples

YANG module used

```
module test-yang-json
 {
   namespace "http://example.com/ns/riftware-1.0/test-yang-json.yang";
   prefix "tyj";

   identity company {
     description "A company";
   }

   identity example {
     base company;
     description "";
   }

   container top {
     description "Top Container";

     leaf-list leaflist {
       type string;
     }

     leaf idrefex {
       type identityref { base "tyj:company"; }
     }

     leaf bin {
       type binary;
     }

     leaf leaf-1 {
       type string;
     }
```

```
    leaf leaf-2 {
      description "Leaf number 2";
      type uint32;
    }

    leaf linstid { type instance-identifier; }

    leaf n1 {config true; type int8 {range "-12..14";} }

    leaf leaf-4 {
      type enumeration {
        enum I_E_A { value 991; }
        enum I_E_B { value 992; }
        enum I_E_C { value 993; }
      }
    }

  list list-1 {
      description "List no. 1";
      key "id";

      leaf id {
        type uint16;
      }

      container list-cont {
        description
          "List 1 Container for the lack of good names";
        leaf yours-truly {
          type string;
        }
      }

      list int-list-1 {
        description "Internal list to list 1";
        key "item";
        leaf item {
          type string;
        }
        leaf name {
          type string;
        }
      }
    }

    leaf leaf-5 {
      description "Leafref example";
      type leafref {
        path "../tyj:leaf-2";
      }
    }

    container leaf_bin {
      description "for test RwYangDom.EmptyLeaf";
      leaf empty1 { type empty; }
      leaf empty2 { type empty; }
```

```
      leaf bool1 { type boolean; }
      leaf bool2 { type boolean; }
      leaf bool3 { type boolean; }
    }

    leaf bit-leaf {
      description "Bit leaf ";
      type bits {
        bit aggregate {
          position 0;
        }
        bit timeout {
          position 1;
        }
        bit sync {
          position 2;
        }
      }
    }

    leaf bindata {
      type binary;
    }

  }

  augment /tyjab:base-cont/tyjab:person {
    list company-list {
      key "iref1";
      leaf iref1 { type identityref { base "company"; } }
      leaf iref2 { type identityref { base "product"; } }
    }
  }
}
```

JSON attributes

Attribute	Description
"name"	Name of the YANG node.
"type"	YANG data type, such as leaf, list, leaf-list, and leafref.
"description"	Text description provided in the YANG module for that node.
"cardinality"	Number of elements contained by the node. Cardinality is set to `0..n` for list and leaf list and `0..1` for all other nodes.

Attribute	Description
"mandatory"	True if the YANG node is marked mandatory in the YANG specification.
"keys"	Key name for the list node (present in list node only). Keys have cardinality of 1.
"properties"	List describing the YANG node with regard to the composite data type.
"data-type"	Enumerates a YANG data type.

Mapping a simple leaf

YANG node

```
leaf leaf-2 {
   description "Leaf number 2";
   type uint32;
 }
```

JSON node

```
{
   "name": "leaf-2",
   "type": "leaf",
   "description": "Leaf number 2",
   "cardinality": "0..1",
   "data-type": "uint32"
   "properties": []
 }
```

Mapping an empty leaf

YANG node

```
leaf empty1 { type empty; }
```

JSON node

```
{
   "name": "empty1",
   "type": "leaf",
   "description": "",
   "cardinality": "0..1",
   "data-type": "empty"
```

```
   "properties": []
 }
```

Mapping a leaf range

YANG node

```
  leaf n1 {type int8 {range "-12..14";} }
```

JSON node

```
{
   "name": "n1",
   "type": "leaf",
   "description": "",
   "cardinality": "0..1",
   "data-type": "int8"
   "properties": []
 }
```

Note: Currently the range is not shown in the corresponding JSON representation.

Mapping a binary leaf

YANG node

```
  leaf bindata {
     type binary;
   }
```

JSON node

```
{
   "name": "bindata",
   "type": "leaf",
   "description": "",
   "cardinality": "0..1",
   "data-type": "binary"
   "properties": []
 }
```

Mapping leaf bits

YANG node

```
leaf bit-leaf {
   description "Bit leaf ";
   type bits {
     bit aggregate {
       position 0;
     }
     bit timeout {
```

```
       position 1;
     }
     bit sync {
       position 2;
     }
   }
 }
```

JSON node

```
{
   "name": "bit-leaf",
   "type": "leaf",
   "description": "Bit leaf ",
   "cardinality": "0..1",
   "data-type": {
     "bits": {
       "bit": {
         "aggregate": {
           "position": 0
         },
         "timeout": {
           "position": 1
         },
         "sync": {
           "position": 2
         }
       }
     }
   }
 "properties": []
 }
```

Mapping a leaf list

YANG node

```
  leaf-list leaflist {
     type string;
   }
```

JSON node

```
{
   "name": "leaflist",
   "type": "leaf_list",
   "description": "",
   "cardinality": "0..N",
   "data-type": "string"
   "properties": []
 }
```

Mapping a leafref

YANG node

```
leaf leaf-2 {
   description "Leaf number 2";
   type uint32;
 }

 leaf leaf-5 {
   description "Leafref example";
   type leafref {
     path "../tyj:leaf-2";
   }
 }
```

JSON node

```
{
   "name": "leaf-5",
   "type": "leaf",
   "description": "Leafref example",
   "cardinality": "0..1",
   "data-type": {
   "leafref": {
     "path": "../tyj:leaf-2"
   }
 }
   "properties": []
 }
```

Mapping a list

YANG node

```
list list-1 {
       description "List no. 1";
       key "id name";

       leaf id {
         type uint16;
         mandatory "false";
       }

       leaf name {
         type string;
         mandatory "false";
       }

       container list-cont {
         description
           "List 1 Container for the lack of good names";
         leaf yours-truly {
           type string;
         }
       }

       list int-list-1 {
```

```
        description "Internal list to list 1";
        key "item";
        leaf item {
          type string;
        }
        leaf name {
          type string;
        }
      }
 }
```

JSON node

```
{
      "name": "list-1",
      "type": "list",
      "description": "List no. 1",
      "cardinality": "0..N",
      "keys": [
        {
          "value": "id"
        },
        {
          "value": "name"
        }
      ],
      "properties": [
        {
          "name": "id",
          "type": "leaf",
          "description": "",
          "cardinality": "0..1",
          "mandatory": "true",
          "data-type": "uint16"
          "properties": []
        },
        {
          "name": "name",
          "type": "leaf",
          "description": "",
          "cardinality": "0..1",
          "mandatory": "true",
          "data-type": "string"
          "properties": []
        },
        {
          "name": "list-cont",
          "type": "container",
          "description": "List 1 Container for the lack of good names",
          "cardinality": "0..1",
          "properties": [
            {
              "name": "yours-truly",
              "type": "leaf",
              "description": "",
              "cardinality": "0..1",
```

```
                "data-type": "string"
                "properties": []
              }
            ]
          },
          {
            "name": "int-list-1",
            "type": "list",
            "description": "Internal list to list 1",
            "cardinality": "0..N",
            "keys": [
              {
                "value": "item"
              }
            ],
            "properties": [
              {
                "name": "item",
                "type": "leaf",
                "description": "",
                "cardinality": "0..1",
                "mandatory": "true",
                "data-type": "string"
                "properties": []
              },
              {
                "name": "name",
                "type": "leaf",
                "description": "",
                "cardinality": "0..1",
                "data-type": "string"
                "properties": []
              }
          ]
        }
      ]
 }
```

Mapping a container

YANG node

```
container leaf_bin {
   description "for test RwYangDom.EmptyLeaf";
   leaf empty1 { type empty; }
   leaf empty2 { type empty; }
   leaf bool1 { type boolean; }
   leaf bool2 { type boolean; }
   leaf bool3 { type boolean; }
 }
```

JSON node

The "properties" attribute is a list of JSON nodes that represents the types that the list will hold.

```
{
   "name": "leaf_bin",
   "type": "container",
   "description": "for test RwYangDom.EmptyLeaf",
   "cardinality": "0..1",
   "properties": [
     {
       "name": "empty1",
       "type": "leaf",
       "description": "",
       "cardinality": "0..1",
       "data-type": "empty"
       "properties": []
     },
     {
       "name": "empty2",
       "type": "leaf",
       "description": "",
       "cardinality": "0..1",
       "data-type": "empty"
       "properties": []
     },
     {
       "name": "bool1",
       "type": "leaf",
       "description": "",
       "cardinality": "0..1",
       "data-type": "boolean"
       "properties": []
     },
     {
       "name": "bool2",
       "type": "leaf",
       "description": "",
       "cardinality": "0..1",
       "data-type": "boolean"
       "properties": []
     },
     {
       "name": "bool3",
       "type": "leaf",
       "description": "",
       "cardinality": "0..1",
       "data-type": "boolean"
       "properties": []
     }
   ]
 }
```

Mapping an identity ref

YANG node

```
identity company {
   description "A company";
 }

 leaf idrefex {
   type identityref { base "tyj:company"; }
 }
```

JSON node

```
{
   "name": "idrefex",
   "type": "leaf",
   "description": "",
   "cardinality": "0..1",
   "data-type": {
     "idref": {
        "base": "tyj:example"
      }
   }
   "properties": []
 }
```

Mapping an enumeration

YANG node

```
leaf leaf-4 {
   type enumeration {
     enum I_E_A { value 991; }
     enum I_E_B { value 992; }
     enum I_E_C { value 993; }
   }
 }
```

JSON node

```
{
   "name": "leaf-4",
   "type": "leaf",
   "description": "",
   "cardinality": "0..1",
   "data-type": {
     "enumeration": {
       "enum": {
         "I_E_A": {
           "value": 991
         },
         "I_E_B": {
           "value": 992
         },
         "I_E_C": {
           "value": 993
```

```
          }
        }
      }
    }
    "properties": []
 }
```

Mapping an instance identifier

YANG node

```
leaf linstid { type instance-identifier; }
```

JSON node

```
{
   "name": "linstid",
   "type": "leaf",
   "description": "",
   "cardinality": "0..1",
   "data-type": "instance_id"
   "properties": []
 }
```

Mapping augmented nodes

YANG node

```
container base-cont {
   list person {
   key "name";

   leaf name {
     description
       "Name of the person";
     type string;
   }

   leaf phone-no {
     description
       "Phone number of the person";
     type string;
   }
  }
 }

 // Augmented node
 augment /tyjab:base-cont/tyjab:person {
   list company-list {
     key "iref1";
     leaf iref1 { type identityref { base "company"; } }
     leaf iref2 { type identityref { base "product"; } }
   }
 }
```

JSON node


```
{
 "person": {
   "name": "person",
   "type": "list",
   "description": "",
   "cardinality": "0..N",
   "keys": [
     {
       "value": "name"
     }
   ],
   "properties": [
     {
       "name": "name",
       "type": "leaf",
       "description": "Name of the person",
       "cardinality": "0..1",
       "mandatory": "true",
       "data-type": "string"
       "properties": []
     },
     {
       "name": "phone-no",
       "type": "leaf",
       "description": "Phone number of the person",
       "cardinality": "0..1",
       "data-type": "string"
       "properties": []
     },
     {
       "name": "test-yang-json:company-list",
       "type": "list",
       "description": "",
       "cardinality": "0..N",
       "keys": [
         {
           "value": "iref1"
         }
       ],
       "properties": [
         {
           "name": "iref1",
           "type": "leaf",
           "description": "",
           "cardinality": "0..1",
           "mandatory": "true",
           "data-type": {
             "idref": {
               "base": "tyj:example"
             }
           }
           "properties": []
         },
         {
```

```
          "name": "iref2",
          "type": "leaf",
          "description": "",
          "cardinality": "0..1",
          "data-type": {
            "idref": {
              "base": "tyj:cloud-platform"
            }
          }
          "properties": []
        }
      ]
    }
  ]
}
}
```

Notes and limitations

- Union and choice-case are not currently supported.
- In the JSON output, node names are not prefixed.

Orchestration API

The Orchestration API is your interface to network service lifecycle management functions and service instantiation functions. The **resource orchestrator** interfaces with the Network Service Orchestrator and resource orchestration engines through secured REST interfaces.

Use the open northbound APIs for service management and orchestration:

- Onboard, instantiate, and terminate network services (**NFVO** MANO `Os-Ma-nfvo` reference point)
- Onboard, instantiate, update, and delete virtual network functions (**VNFM** MANO `Or-Vnfm` reference point)

NS and VNF Package Management (rw-pkg-mgmt:rw-pkg-mgmt)

REST wrapper for the **NS** and **VNF** package service. Provides methods for onboarding, updating and downloading service and virtual network function packages.

Note: These APIs are different from the other REST APIs in this guide. In particular, onboarding and updating NS and VNF packages follow pull semantics, rather than push semantics. You must host a tgz file on any HTTP file server and then call the API to instruct the package management service to pull the package from this server.

Methods and relative URLs

rw-pkg-mgmt:rw-pkg-mgmt

Operation	Method	Relative URL	Description
Package onboard	POST	/api/operations/package-create	Onboard a network service descriptor package to the catalog. When you send a POST request, the back end downloads the tgz file, expands it, and validates the checksum.txt file. Then it pushes the descriptor in the package to the nsd:nsd endpoint described in "NS Descriptor Management (nsd:nsd)" on page 68. This operation validates the presence of: • VNF packages for the list of VNFs in the network service • Mandatory elements • External connection points required by the NS in the VNF descriptor The body of the POST request must contain the following items: • 'external-url': HTTP endpoint where the package (tgz) file is hosted • 'package-type': NSD/VNFD • 'package-id': unique identifier for this package See "rw-pkg-mgmt:package-create" on page 62.

Operation	Method	Relative URL	Description
Package update	POST	/api/operations/package-update	Similar to the package-onboard API in the previous row. The body of the POST request must contain the following items: • 'external-url': HTTP endpoint where the package (tgz) file is hosted • 'package-type': NSD/VNFD • 'package-id': unique identifier for this package See "rw-pkg-mgmt:package-update" on page 63.
Package onboard / update status	GET	/api/operational/download-jobs	Retrieve information about onboarding (create or update) jobs. See "rw-pkg-mgmt:download-jobs" on page 64.

Operation	Method	Relative URL	Description
Package export	POST	/api/operations/package-export	Create an export request for a package. The body of the POST request must contain the following items: • "package-type": NSD/VNFD • "package-id": ID of the package to be downloaded • "export-format": YAML • "export-grammar": OSM • "export-schema": MANO/RIFT After this POST operation is executed, the back end starts the process of generating a tgz for the package (NS or VNF) that is being requested. Check the status of this operation by issuing GET requests on https://<orchestrator_ip>:4567/api/export/707fef01-7bf0-40cc-a33b-c65947173e10/state. If successful, the response will contain a property “status” with a value of success and a property with filename. Then perform a GET request at https://<orchestrator_ip>:4567/api/export/<filename> to grab the package. **Note:** The status and actual package export is provided by a different server that runs on a different port from the remainder of the REST API service. See "rw-pkg-mgmt:package-export" on page 66.

MANO reference points

Os-Ma-nfvo

Schema

```
module rw-pkg-mgmt
 {
   namespace "http://example.com/ns/riftware-1.0/rw-pkg-mgmt";
   prefix "rw-pkg-mgmt";

   import ietf-yang-types {
```

```
    prefix "yang";
  }

  import rw-pb-ext {
    prefix "rwpb";
  }

  import rw-cli-ext {
    prefix "rwcli";
  }

  import rw-cloud {
    prefix "rwcloud";
  }

  import rwcal {
    prefix "rwcal";
  }

  import mano-types {
    prefix "manotypes";
  }

  revision 2016-06-01 {
    description
      "Initial revision.";
  }

  typedef task-status {
    type enumeration {
      enum QUEUED;
      enum IN_PROGRESS;
      enum DOWNLOADING;
      enum CANCELLED;
      enum COMPLETED;
      enum FAILED;
    }
  }

  typedef export-schema {
    type enumeration {
      enum RIFT;
      enum MANO;
    }
  }

  typedef export-grammar {
    type enumeration {
      enum OSM;
    }
  }

  typedef export-format {
    type enumeration {
      enum YAML;
      enum JSON;
```

```
    }
  }

  grouping external-url-data {
    leaf external-url {
      description "Url to download";
      type string;
    }

    leaf username {
      description "username if the url uses authentication";
      type string;
    }

    leaf password {
      description "password if the url uses authentication";
      type string;
    }
  }

  grouping package-identifer {
    leaf package-type {
      description "Type of the package";
      type manotypes:package-type;
    }

    leaf package-id {
      description "Id of the package";
      type string;
    }
  }

  grouping package-file-identifer {
    uses package-identifer;

    leaf package-path {
      description "Relative path in the package";
      type string;
    }
  }

  grouping download-task-status {
    leaf status {
      description "The status of the download task";
      type task-status;
      default QUEUED;
    }

    leaf detail {
      description "Detailed download status message";
      type string;
    }

    leaf progress-percent {
      description "The download progress percentage (0-100)";
      type uint8;
```

```
      default 0;
    }

    leaf bytes_downloaded {
      description "The number of bytes downloaded";
      type uint64;
      default 0;
    }

    leaf bytes_total {
      description "The total number of bytes to write";
      type uint64;
      default 0;
    }

    leaf bytes_per_second {
      description "The total number of bytes written per second";
      type uint32;
      default 0;
    }

    leaf start-time {
      description "start time (unix epoch)";
      type uint32;
    }

    leaf stop-time {
      description "stop time (unix epoch)";
      type uint32;
    }
  }

  container download-jobs {
    rwpb:msg-new DownloadJobs;
    description "Download jobs";
    config false;

    list job {
      rwpb:msg-new DownloadJob;
      key "download-id";

      leaf download-id {
        description "Unique UUID";
        type string;
      }

      leaf url {
        description "URL of the download";
        type string;
      }

      uses package-file-identifer;
      uses download-task-status;
    }
  }
```

```
  rpc get-package-endpoint {
    description "Retrieves the endpoint for the descriptor";

    input {
      uses package-identifer;
    }

    output {
     leaf endpoint {
        description "Endpoint that contains all the package-related data";
        type string;
      }
    }
  }

  rpc get-package-schema {
    description "Retrieves the schema for the package type";

    input {
      leaf package-type {
        description "Type of the package";
        type manotypes:package-type;
      }
    }

    output {
      leaf-list schema {
        description "List of all top level directories for the package.";
        type string;
      }
    }
  }

  rpc package-create {
    description "Creates a new package";

    input {
      uses package-identifer;
      uses external-url-data;
    }

    output {
     leaf transaction-id {
        description "Valid ID to track the status of the task";
        type string;
      }
    }
  }

  rpc package-update {
    description "Creates a new package";

    input {
      uses package-identifer;
      uses external-url-data;
    }
```

```
    output {
     leaf transaction-id {
        description "Valid ID to track the status of the task";
        type string;
      }
    }
  }

  rpc package-export {
    description "Export a package";

    input {
      uses package-identifer;

      leaf export-schema {
        description "Schema to export";
        type export-schema;
        default RIFT;
      }

      leaf export-grammar {
        description "Schema to export";
        type export-grammar;
        default OSM;
      }

      leaf export-format {
        description "Format to export";
        type export-format;
        default YAML;
      }

    }

    output {
     leaf transaction-id {
        description "Valid ID to track the status of the task";
        type string;
      }

     leaf filename {
        description "Valid ID to track the status of the task";
        type string;
      }
    }
  }

  rpc package-file-add {
    description "Retrieves the file from the URL and store it in the
package";

    input {
      uses package-file-identifer;
      uses external-url-data;
    }
```

```
     output {
      leaf task-id {
         description "Valid ID to track the status of the task";
         type string;
       }
     }
   }

   rpc package-file-delete {
     description "Retrieves the file from the URL and store it in the
package";

     input {
       uses package-file-identifer;
     }

     output {
      leaf status {
         description "Status of the delete operation";
         type string;
       }

       leaf error-trace {
         description "Trace in case of a failure";
         type string;
       }

     }
   }

 }
```

rw-pkg-mgmt:package-create

Onboard a network service descriptor package to the catalog.

When you send a POST request, the back end downloads the tgz file, expands it, and validates the checksum.txt file. Then it pushes the descriptor in the package to the nsd:nsd endpoint described in "NS Descriptor Management (nsd:nsd)" on page 68.

This operation validates the presence of:

- VNF packages for the list of VNFs in the network service
- Mandatory elements
- External connection points required by the NS in the VNF descriptor

The body of the POST request must contain the following items:

- 'external-url': HTTP endpoint where the package (tgz) file is hosted
- 'package-type': NSD/VNFD
- 'package-id': unique identifier for this package

REST URI path

/input

Fields

ID	Type	Cardinality	Description
package-type	enum	1	Type of the package being onboarded: • NSD • VNFD
package-id	string	1	ID of the package.
external-url	string	1	URL to download.
username	string	1	Username if the URL uses authentication.
password	string	1	Password if the URL uses authentication.

rw-pkg-mgmt:package-update

The body of the POST request must contain the following items:

- 'external-url': HTTP endpoint where the package (tgz) file is hosted
- 'package-type': NSD/VNFD
- 'package-id': unique identifier for this package

This API is similar to "rw-pkg-mgmt:package-create" on page 62.

REST URI path

/input

Fields

ID	Type	Cardinality	Description
package-type	enum	1	Type of the package being onboarded: • NSD • VNFD
package-id	string	1	ID of the package.
external-url	string	1	URL to download.
username	string	1	Username if the URL uses authentication.
password	string	1	Password if the URL uses authentication.

rw-pkg-mgmt:download-jobs

Retrieve information about onboarding (create or update) jobs.

REST URI path

/api/operational/download-jobs

Fields

ID	Type	Cardinality	Description
job	list	0..n	A list of jobs. See "rw-pkg-mgmt:job" on page 64.

rw-pkg-mgmt:job

/rw-pkg-mgmt:download-jobs/job/STRING

ID	Type	Cardinality	Description
download-id	string	1	Unique UUID.
url	string	1	URL of the download.
package-type	enum	1	Type of the package being onboarded: • NSD • VNFD
package-id	string	1	ID of the package.
package-path	string	1	Relative path in the package.

ID	Type	Cardinality	Description
status	enum	1	The status of the download task. • QUEUED (default) • IN_PROGRESS • DOWNLOADING • CANCELLED • COMPLETED • FAILED
detail	string	1	Detailed download status message.
progress-percent	uint8	1	[Default 0] The download progress percentage (0-100).
bytes_downloaded	uint64	1	[Default 0] The number of bytes downloaded.
bytes_total	uint64	1	[Default 0] The total number of bytes to write.
bytes_per_second	uint32	1	[Default 0] The total number of bytes written per second.
start-time	uint32	1	Start time (unix epoch).
stop-time	uint32	1	Stop time (unix epoch).

rw-pkg-mgmt:package-export

Create an export request for a package.

The body of the POST request must contain the following items:

- "package-type": NSD/VNFD
- "package-id": ID of the package to be downloaded
- "export-format": YAML
- "export-grammar": OSM
- "export-schema": MANO/RIFT

After this POST operation is executed, the back end starts the process of generating a tgz for the package (NS or VNF) that is being requested. Check the status of this operation by issuing GET requests on https://<orchestrator_ip>:4567/api/export/707fef01-7bf0-40cc-a33b-c65947173e10/state. If successful, the response will contain a property “status” with a value of success and a property with filename.

Then perform a GET request at https://<orchestrator_ip>:4567/api/export/<filename> to grab the package.

Note: The status and actual package export is provided by a different server that runs on a different port from the remainder of the REST API service.

REST URI path

/api/operations/package-export

Fields

ID	Type	Cardinality	Description
package-type	enum	1	Type of the package being onboarded: • NSD • VNFD
package-id	string	1	ID of the package.

ID	Type	Cardinality	Description
export-schema	enum	1	Schema to export: • RIFT • MANO
export-grammar	enum	1	Schema to export: • OSM
export-format	enum	1	Format to export: • YAML • JSON

NS Descriptor Management (nsd:nsd)

REST wrapper for the NSD service (nsd:nsd-catalog). Provides methods for onboarding, updating, querying, and deleting a network service descriptor.

Methods and relative URLs

nsd:nsd

Method	Relative URL	Description
POST	/api/config/nsd-catalog	Onboard a network service descriptor package to the catalog. When you send a POST request, the back end validates the presence of: • VNF packages for the list of VNFs in the network service • Mandatory elements • External connection points required by the NS in the VNF descriptor
GET	/api/running/nsd-catalog/nsd/{UUID}	Query a network service descriptor.
GET	/api/running/nsd-catalog/nsd	Retrieve a list of all network descriptors in the catalog.
PUT	/api/running/nsd-catalog/nsd/{UUID}	Update a network service descriptor. The back end verifies that NSD or its constituent VNFDs are not in use by a running instances of either the VNF or network service.

Method	Relative URL	Description
DELETE	/api/running/nsd-catalog/nsd/{UUID}	Delete a network service descriptor. When you send a DELETE request, the back end verifies that the specified NSD is not in use by a running instance of the network service.

nsd:vld

Method	Relative URL	Description
GET		Retrieve a list of a virtual links from the catalog

Fields

ID	Type	Cardinality	Description
id	string	1	Unique identifier for the Network Service Descriptor (NSD).
name	string	1	NSD name.
short-name	string	1	NSD short name to use as a label in the UI.
vendor	string	1	Vendor of the NSD.

ID	Type	Cardinality	Description
logo	string	1	File path of the vendor-specific logo. For example, icons/mylogo.png The logo should be part of the network service package. SVG format is preferred, but PNG is supported. Although there is no hard limit on size and dimension, a square image under 200px by 200px is preferred.
description	string	1	Description of the NSD.
version	string	1	Version of the NSD.
connection-point	list	0..n	A list of references to network service connection points. See "nsd:connection-point" on page 82.
vld	list	0..n	List of Virtual Link Descriptors (VLDs). See "nsd:vld" on page 83.
constituent-vnfd	list	0..n	List of Virtual Network Function Descriptors (VNFDs) that are part of this network service. See "nsd:constituent-vnfd" on page 87.
placement-groups	list	0..n	List of placement groups at the NS level. See "nsd:placement-groups" on page 88.
ip-profiles-list	list	0..n	List of IP profiles. See "nsd:ip-profiles" on page 90.

ID	Type	Cardinality	Description
vnf-dependency	list	0..n	List of VNF dependencies. See "nsd:vnf-dependency" on page 92.
monitoring-param	list	0..n	List of monitoring parameters at the network service level. See "nsd:monitoring-param" on page 93.
input-parameter-xpath	list	0..n	List of XPaths to parameters inside the NSD that can be customized during instantiation. See "nsd:input-parameter-xpath" on page 97.
parameter-pool	list	0..n	Pool of parameter values that must be pulled from during configuration. See "nsd:parameter-pool" on page 98.
service-primitive	list	0..n	Network service level configuration primitives. See "nsd:service-primitive" on page 99.
initial-config-primitive	list	0..n	Set of configuration primitives to be executed when the network service comes up. See "nsd:initial-config-primitive" on page 104.
terminate-config-primitive	list	0..n	Set of configuration primitives to be executed before during termination of the network service. See "nsd:terminate-config-primitive" on page 105.
key-pair	list	0..n	Used to configure the list of public keys to be injected as part of network service instantiation. See "nsd:key-pair" on page 106.

ID	Type	Cardinality	Description
user	list	0..n	List of users to be added through cloud-config. See "nsd:user" on page 107.

MANO reference points

Os-Ma-nfvo

Schema

```
module nsd
{
  namespace "urn:ietf:params:xml:ns:yang:nfvo:nsd";
  prefix "nsd";

  import rw-pb-ext {
    prefix "rwpb";
  }

  import vld {
    prefix "vld";
  }

  import vnfd {
    prefix "vnfd";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-yang-types {
    prefix "yang";
  }

  import mano-types {
    prefix "manotypes";
  }

  revision 2014-10-27 {
    description
      "Initial revision. This YANG file defines
       the Network Service Descriptor (NSD)";
    reference
      "Derived from earlier versions of base YANG files";
  }

  grouping primitive-parameter {
    leaf name {
```

```
      description
          "Name of the parameter.";
      type string;
    }

    leaf data-type {
      description
          "Data type associated with the name.";
      type manotypes:parameter-data-type;
    }

    leaf mandatory {
      description "Is this field mandatory";
      type boolean;
      default false;
    }

    leaf default-value {
      description "The default value for this field";
      type string;
    }

    leaf parameter-pool {
      description "NSD Parameter pool name to use for this parameter";
      type string;
    }
  }

  grouping nsd-descriptor {
    leaf id {
      description "Identifier for the NSD.";
      type string;
    }

    leaf name {
      description "NSD name.";
      mandatory true;
      type string;
    }

    leaf short-name {
      description "NSD short name.";
      type string;
    }

    leaf vendor {
      description "Vendor of the NSD.";
      type string;
    }

    leaf logo {
      description
        "File path for  the vendor specific logo. For example
icons/mylogo.png.
         The logo  should be part of the network service";
      type string;
```

```
    }

    leaf description {
      description "Description of the NSD.";
      type string;
    }

    leaf version {
      description "Version of the NSD";
      type string;
    }

    list connection-point {
      description
          "List for external connection points.
          Each NS has one or more external connection
          points. As the name implies that external
          connection points are used for connecting
          the NS to other NS or to external networks.
          Each NS exposes these connection points to
          the orchestrator. The orchestrator can
          construct network service chains by
          connecting the connection points between
          different NS.";

      key "name";
      leaf name {
        description
            "Name of the NS connection point.";
        type string;
      }

      leaf type {
        description
            "Type of the connection point.";
        type manotypes:connection-point-type;
      }
    }

    /* Model limitation,
       see the comments under vnfd-connection-point-ref
     */
    list vld {
      description
          "List of Virtual Link Descriptors.";

      key "id";

      leaf id {
        description
            "Identifier for the VLD.";
        type string;
      }

      leaf name {
        description
```

```
            "Virtual Link Descriptor (VLD) name.";
        type string;
      }

      leaf short-name {
        description
            "Short name for VLD for UI";
        type string;
      }

      leaf vendor {
        description "Provider of the VLD.";
        type string;
      }

      leaf description {
        description "Description of the VLD.";
        type string;
      }

      leaf version {
        description "Version of the VLD";
        type string;
      }

      leaf type {
        type manotypes:virtual-link-type;
      }

      leaf root-bandwidth {
        description
            "For ELAN this is the aggregate bandwidth.";
        type uint64;
      }

      leaf leaf-bandwidth {
        description
            "For ELAN this is the bandwidth of branches.";
        type uint64;
      }

      list vnfd-connection-point-ref {
        description
            "A list of references to connection points.";
        key "member-vnf-index-ref vnfd-connection-point-ref";

        leaf member-vnf-index-ref {
          description "Reference to member-vnf within constituent-vnfds";
          type leafref {
            path "../../../constituent-vnfd/member-vnf-index";
          }
        }

        leaf vnfd-id-ref {
          description
              "A reference to a vnfd. This is a
```

```
                leafref to path:
                    ../../nsd:constituent-vnfd
                    + [nsd:id = current()/../nsd:id-ref]
                    + /nsd:vnfd-id-ref
                NOTE: An issue with confd is preventing the
                use of xpath. Seems to be an issue with leafref
                to leafref, whose target is in a different module.
                Once that is resolved this will switched to use
                leafref";
            type leafref {
              path "../../../constituent-vnfd" +
                   "[member-vnf-index = current()/../member-vnf-index-ref]" +
                   "/vnfd-id-ref";
            }
          }

          leaf vnfd-connection-point-ref {
            description
              "A reference to a connection point name
                  in a vnfd. This is a leafref to path:
                      /vnfd:vnfd-catalog/vnfd:vnfd
                      + [vnfd:id = current()/../nsd:vnfd-id-ref]
                      + /vnfd:connection-point/vnfd:name
                  NOTE: An issue with confd is preventing the
                  use of xpath. Seems to be an issue with leafref
                  to leafref, whose target is in a different module.
                  Once that is resolved this will switched to use
                  leafref";
            type string;
          }
        }

        // replicate for pnfd container here
        uses manotypes:provider-network;

        leaf mgmt-network {
           description "Flag indicating whether this network is a VIM
management network";
           type boolean;
           default false;
        }

        choice init-params {
          description "Extra parameters for VLD instantiation";

          case vim-network-ref {
            leaf vim-network-name {
              description
                  "Name of network in VIM account. This is used to indicate
                     pre-provisioned network name in cloud account.";
              type string;
            }
          }

          case vim-network-profile {
            leaf ip-profile-ref {
```

```
            description "Named reference to IP-profile object";
            type string;
          }
        }

      }
    }

    list constituent-vnfd {
      description
          "List of VNFDs that are part of this
          network service.";

      key "member-vnf-index";

      leaf member-vnf-index {
        description
          "Identifier/index for the VNFD. This separate id
           is required to ensure that multiple VNFs can be
           part of single NS";
        type uint64;
      }

      leaf vnfd-id-ref {
        description
          "Identifier for the VNFD.";
        type leafref {
          path "/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id";
        }
      }

      leaf start-by-default {
        description
          "VNFD is started as part of the NS instantiation";
        type boolean;
        default true;
      }
    }


    list placement-groups {
      description "List of placement groups at NS level";

      key "name";
      uses manotypes:placement-group-info;

      list member-vnfd {
        description
            "List of VNFDs that are part of this placement group";

        key "member-vnf-index-ref";

        leaf member-vnf-index-ref {
          description "member VNF index of this member VNF";
          type leafref {
            path "../../../constituent-vnfd/member-vnf-index";
```

```
          }
        }

        leaf vnfd-id-ref {
          description
              "Identifier for the VNFD.";
          type leafref {
            path "../../../constituent-vnfd" +
                 "[member-vnf-index = current()/../member-vnf-index-ref]" +
                 "/vnfd-id-ref";
          }
        }
      }
    }

    uses manotypes:ip-profile-list;

    list vnf-dependency {
      description
          "List of VNF dependencies.";
      key vnf-source-ref;
      leaf vnf-source-ref {
        type leafref {
          path "../../constituent-vnfd/vnfd-id-ref";
        }
      }
      leaf vnf-depends-on-ref {
        description
            "Reference to VNF that source VNF depends.";
        type leafref {
          path "../../constituent-vnfd/vnfd-id-ref";
        }
      }
    }


    list monitoring-param {
      description
        "List of monitoring parameters from VNFs that should be
        propagated up into NSR";
      key "id";

      leaf id {
        type string;
      }

      leaf name {
        type string;
      }

      uses manotypes:monitoring-param-value;
      uses manotypes:monitoring-param-ui-data;
      uses manotypes:monitoring-param-aggregation;

      list vnfd-monitoring-param {
        description "A list of VNFD monitoring params";
```

```
        key "member-vnf-index-ref vnfd-monitoring-param-ref";

        leaf vnfd-id-ref {
          description
             "A reference to a vnfd. This is a
              leafref to path:
                  ../../../../nsd:constituent-vnfd
                  + [nsd:id = current()/../nsd:id-ref]
                  + /nsd:vnfd-id-ref
              NOTE: An issue with confd is preventing the
              use of xpath. Seems to be an issue with leafref
              to leafref, whose target is in a different module.
              Once that is resolved this will switched to use
              leafref";
          type leafref {
            path "../../../constituent-vnfd" +
                 "[member-vnf-index = current()/../member-vnf-index-ref]" +
                 "/vnfd-id-ref";
          }
        }

        leaf vnfd-monitoring-param-ref {
          description "A reference to the VNFD monitoring param";
          type leafref {
            path "/vnfd:vnfd-catalog/vnfd:vnfd"
              + "[vnfd:id = current()/../vnfd-id-ref]"
              + "/vnfd:monitoring-param/vnfd:id";
          }
        }

        leaf member-vnf-index-ref {
          description
            "Mandatory reference to member-vnf within constituent-vnfds";
          type leafref {
            path "../../../constituent-vnfd/member-vnf-index";
          }
        }
      }
    }

    uses manotypes:input-parameter-xpath;

    list parameter-pool {
      description
        "Pool of parameter values which must be
         pulled from during configuration";
      key "name";

      leaf name {
        description
            "Name of the configuration value pool";
        type string;
      }

      container range {
        description
```

```
            "Create a range of values to populate the pool with";

        leaf start-value {
          description
              "Generated pool values start at this value";
          type uint32;
          mandatory true;
        }

        leaf end-value {
          description
              "Generated pool values stop at this value";
          type uint32;
          mandatory true;
        }
      }
    }

    uses manotypes:ns-service-primitive;

    list initial-config-primitive {
      rwpb:msg-new NsdInitialConfigPrimitive;
      description
        "Initial set of configuration primitives for NSD.";
      key "seq";

      uses manotypes:initial-config;
    }

    list terminate-config-primitive {
      rwpb:msg-new NsdTerminateConfigPrimitive;
      description
        "Set of configuration primitives during
         termination for NSD.";
      key "seq";

      uses manotypes:event-config;
    }

    uses manotypes:cloud-config;

  }



  container nsd-catalog {

    list nsd {
      key "id";

      uses nsd-descriptor;
    }
  }

}
```

Examples

See "API Examples" on page 346

nsd:connection-point

A list of references to network service connection points.

Each network service (NS):

- Has one or more external connection points used to link the NS to other NS or to external networks.
- Exposes these connection points to the orchestrator.

The orchestrator can construct network service chains by joining the connection points between different network services.

REST URI path

/nsd:nsd-catalog/nsd/STRING/connection-point/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the NS external connection point.
type	enum	1	Type of connection point. **Supported types:** VPORT: Virtual Port

nsd:vld

List of Virtual Link Descriptors (VLDs)

Network connections are defined by connection points and virtual links. There are three types of connection points:

- Connect a network service to the outside world, such as the network service endpoint, described in the NSD
- Connect between **VNFs** within a network service, such as the external interface of the VNF, described in the VNFD
- Connect between VMs, described in the **VNFC**

There are also two types of virtual links:

- External virtual links, which can be connected to network service endpoints and external VNF interfaces
- Internal virtual links, which can be connected to external VNF interfaces and VNFCs

Virtual links also follow the Metro Ethernet Forum **E-LINE**, **E-TREE**, and **E-LAN** services. Virtual link descriptors (VLDs) contain the bandwidth and **QoS** requirements of the interconnection.

VLDs are required for a functioning NSD.

REST URI path

/nsd:nsd-catalog/nsd/STRING/vld/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Unique identifier for the VLD.
name	string	1	VLD name.
short-name	string	1	**NSD** short name to use as label in the UI.
vendor	string	1	Provider of the VLD.
description	string	1	Description of the VLD.

ID	Type	Cardinality	Description
version	string	1	Version of the VLD.
type	enum	1	Type of the virtual link. **Supported values:** **ELAN**: A multipoint service connecting a set of VNFs.
root-bandwidth	uint64	1	For ELAN this is the aggregate bandwidth.
leaf-bandwidth	uint64	1	For ELAN this is the bandwidth of branches.
vnfd-connection-point-ref	list	0..n	A list of references to connection points. See "nsd:vnfd-connection-point-ref" on page 85.
provider-network	container	1	Container for the provider network. See "nsd:provider-network " on page 86.
mgmt-network	boolean	1	[Default *true*] Denotes whether this network is a **VIM** management network.
vim-network-name	string		Name of network in VIM account. This field indicates pre-provisioned network name in cloud account.
ip-profile-ref	string	1	Named reference to IP-profile object.

nsd:vnfd-connection-point-ref

URI path: /nsd:nsd-catalog/nsd/STRING/vld/STRING/vnfd-connection-point-ref/0,STRING

ID	Type	Cardinality	Description
member-vnf-index-ref	reference	1	Reference to member-vnf within constituent units. This is a leafref to path: ../../../nsd:constituent-vndf/nsd:member-vnf-index
vnf-id-ref	string	1	Reference to a VNFD. This is a leafref to path: ../../../nsd:constituent-vnfd + [nsd:id = current()/../nsd:id-ref] + /nsd:vnfd-id-ref
vnfd-connection-point-ref	string	1	Reference to a connection point name in a VNFD. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd + [vnfd:id = current()/../nsd:vnfd-id-ref] + /vnfd:connection-point/vnfd:name

nsd:provider-network

URI path /nsd:nsd-catalog/nsd/STRING/vld/STRING/provider-network

ID	Type	Cardinality	Description
physical-network	string	1	Name of the physical network on which the provider network is built.
overlay-type	enum	0..1	Identifies the type of the overlay network, which is a virtual network that is built on top of an existing network and is supported by its infrastructure. **Supported values:** • LOCAL — A network that can be realized on a single host only. • FLAT — The simplest networking environment in which each instance receives a fixed IP from the pool. All instances are attached to the same bridges. • VLAN — A network of computers in which the computers behaves as if they are connected to the same wire. However, the computers might be physically located on different segments of a LAN. • VXLAN — A proposed encapsulation protocol for running an overlay network on existing **Layer 3** infrastructure • GRE — GRE tunnels encapsulate isolated **Layer2** network traffic in IP packets. Packets are routed between compute and networking nodes using the hosts' network connectivity and routing tables.
segmentation-id	uint32	1	Segmentation ID.

nsd:constituent-vnfd

A list of Virtual Network Function Descriptors (**VNFDs**) that are part of this network service.

REST URI path

/nsd:nsd-catalog/nsd/STRING/constituent-vnfd/0

Fields

ID	Type	Cardinality	Description
member-vnfd-index	uint64	1	[Required] Identifier/index for the VNFD. **Note:** This separate ID is required so that multiple **VNF**s can be part of a single network service.
vnfd-id-ref	leafref	1	Identifier for the VNFD. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id
start-by-default	boolean	1	[Default *true*] VNFD is started as part of network service instantiation.

nsd:placement-groups

A list of placement groups at the network service level.

REST URI path

/nsd:nsd-catalog/nsd/STRING/placement-groups/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Place group construct to define the compute resource placement strategy in cloud environment.
requirement	string	1	Describes the intent/rationale behind this placement group. **Note:** This free-text field is for human consumption only
strategy	enum	1	Strategy associated with this placement group. **Supported values:** • COLOCATION: [Default] Share the physical infrastructure (hypervisor/network) among all members of this group. • ISOLATION: Do not share the physical infrastructure (hypervisor/network) among the members of this group
member-vnfd	list	0..n	List of VNFDs that are part of this placement group. See "nsd:member-vnfd" on page 89.

nsd:member-vnfd

/nsd:nsd-catalog/nsd/STRING/placement-groups/STRING/member-vnfd/0

ID	Type	Cardinality	Description
member-vnf-index-ref	leafref	1	Member VNF index of this member VNF. This is a leafref to path: ../../../constituent-vnfd/member-vnf-index
vnfd-id-ref	leafref	1	Identifier for the VNFD. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id

nsd:ip-profiles

List of IP profiles. IP profiles describe the IP characteristics for the virtual link.

REST URI path

/nsd:nsd-catalog/nsd/STRING/ip-profiles/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the IP profile.
description	string	1	Description of the IP profile.
ip-profile-params	container	1	Information about the IP profile. See "nsd:ip-profile-params" on page 90.

nsd:ip-profile-params

/nsd:nsd-catalog/nsd/STRING/ip-profiles/STRING/ip-profile-params

ID	Type	Cardinality	Description
ip-version	enum	1	[Default IPv4] Version of the Internet Protocol used.
subnet-address	union	1	Subnet IP prefix associated with this IP profile.
gateway-address	union	1	IP address of the default gateway associated with this IP profile.
security-group	string	1	Name of the security group.

ID	Type	Cardinality	Description
dns-server	list	0..n	List of DNS servers associated with this IP profile. See "nsd:dns-server" on page 91.
dhcp-params	container	1	Container for DHCP parameters. See "nsd:ip-profile-params" on page 90.
subnet-prefix-pool	string	1	VIM-specific reference to pre-created subnet prefix.

nsd:dns-server

/nsd:nsd-catalog/nsd/STRING/ip-profiles/STRING/ip-profile-params/dns-server/UNION_VALUE

ID	Type	Cardinality	Description
address	union	1	List of DNS servers associated with this IP profile.

nsd:dhcp-params

/nsd:nsd-catalog/nsd/STRING/ip-profiles/STRING/ip-profile-params/dhcp-params

ID	Type	Cardinality	Description
enabled	boolean	1	[Default *true*] Indicates if DHCP is enabled.
start-address	union	1	Start IP address of the IP address range associated with DHCP domain.
count	uint32	1	Size of the DHCP pool associated with DHCP domain.

nsd:vnf-dependency

List of **VNF** dependencies in the Network Service Descriptor.

REST URI path

/nsd:nsd-catalog/nsd/STRING/vnf-dependency/STRING

Fields

ID	Type	Cardinality	Description
vnf-dependency	leafref	1	List of VNF dependencies. This is a leafref to path: ../../../vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id
vnf-depends-on-ref	leafref	1	Reference to VNFD that on which the source VNF depends. This is a leafref to path: ../../../vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id

nsd:monitoring-param

List of monitoring parameters from **VNFs** to propagate to the Network Service Record (NSR).

REST URI path

/nsd:nsd-catalog/nsd/STRING/monitoring-param/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Identifier for the parameter.
name	string	1	Name of the monitoring parameter
value-type	enum	1	The type of the parameter value. **Supported values:** • INT (default) • DECIMAL • STRING
value-type	enum	1	The type of the parameter value. **Supported values:** • INT (default) • DECIMAL • STRING
numeric-constraints	container	1	Constraints for the numbers. See "nsd:numeric-constraints" on page 95.
text-constraints	container	1	Constraints for the string. See "nsd:text-constraints" on page 95.

ID	Type	Cardinality	Description
value-integer	int64	1	Current value for integer parameter.
value-decimal	decimal164	1	Current value for decimal parameter.
value-string	string	1	Current value for the string parameter.
description	string	1	
group-tag	string	1	A simple tag to group monitoring parameters
widget-type	enum	1	Type of the widget, typically used by the UI. **Supported values:** • HISTOGRAM • BAR • GAUGE • SLIDER • COUNTER • TEXTBOX
aggregation-type	enum	1	Aggregation type for the monitoring parameter. **Supported values:** • AVERAGE • MINIMUM • MAXIMUM • COUNT • SUM

ID	Type	Cardinality	Description
vnfd-monitoring-param	list	0..n	A list of VNFD monitoring parameters. See "nsd:vnfd-monitoring-param" on page 96.

nsd:numeric-constraints

/nsd:nsd-catalog/nsd/STRING/monitoring-param/STRING/numeric-constraints

ID	Type	Cardinality	Description
min-value	uint64	1	Minimum value for the parameter.
max-value	uint64	1	Maximum value for the parameter.

nsd:text-constraints

/nsd:nsd-catalog/nsd/STRING/monitoring-param/STRING/text-constraints

ID	Type	Cardinality	Description
min-length	uint8	1	Minimum string length for the parameter.
max-length	uint8	1	Maximum string length for the parameter.

nsd:vnfd-monitoring-param

/nsd:nsd-catalog/nsd/STRING/monitoring-param/STRING/vnfd-monitoring-param/0,STRING

ID	Type	Cardinality	Description
vnfd-id-ref	leafref	1	A reference to a VNFD. This is a leafref to path: ../../../../nsd:constituent-vnfd + [nsd:id = current()/../nsd:id-ref] + /nsd:vnfd-id-ref
vnfd-monitoring-param-ref	leafref	1	A reference to the VNFD monitoring parameter. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd + [vnfd:id = current()/../vnfd-id-ref] + /vnfd:monitoring-param/vnfd:id
member-vnf-index-ref	leafref	1	Optional reference to member-vnf within constituent-vnfds.

nsd:input-parameter-xpath

List of **XPaths** to parameters inside the Network Service Descriptor that can be customized during instantiation.

REST URI path

/nsd:nsd-catalog/nsd/STRING/input-parameter-xpath/STRING

Fields

ID	Type	Cardinality	Description
xpath	string	1	An xpath that specifies the element in a descriptor.
label	string	1	A descriptive string.
default-value	string	1	A default value for this input parameter.

nsd:parameter-pool

Pool of parameter values from which to pull during configuration.

REST URI path

/nsd:nsd-catalog/nsd/STRING/parameter-pool/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the configuration value pool.
range	container	1	Create a range of values from which to populate the pool. See "nsd:range" on page 98.

nsd:range

/nsd:nsd-catalog/nsd/STRING/parameter-pool/STRING/range

ID	Type	Cardinality	Description
start-value	uint32	1	[Required] Generated pool values start at this value.
end-value	uint32	1	[Required] Generated pool values end at this value.

nsd:service-primitive

Network service-level service primitives for the Network Service Descriptor.

REST URI path

/nsd:nsd-catalog/nsd/STRING/service-primitive/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the service primitive.
parameter	list	0..n	List of parameters for the service primitive. See "nsd:parameter" on page 100.
parameter-group	list	0..n	Grouping of parameters that are logically grouped in UI. See "nsd:parameter-group" on page 101.
vnf-primitive-group	list	0..n	List of service primitives grouped by the VNF. See "nsd:vnf-primitive-group" on page 102.
used-defined-script	string	1	A user-defined script.

nsd:parameter

/nsd:nsd-catalog/nsd/STRING/service-primitive/STRING/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of parameter.
data-type	enum	1	Data type associated with the name. **Supported values:** • STRING • INTEGER • BOOLEAN
mandatory	boolean	1	[Default is *false*] Specifies whether this field is mandatory.
default-value	string	1	The default value for the field.
parameter-pool	string	1	NSD parameter pool name to use for this parameter.
read-only	boolean	1	The value should be dimmed by the UI. Applies only to parameters with default values.
hidden	boolean	1	The value should be hidden by the UI. Applies only to parameters with default values
out	boolean	1	[Default is *false*] Specifies if this is an output of the primitive execution

nsd:parameter-group

/nsd:nsd-catalog/nsd/STRING/service-primitive/STRING/parameter-group/STRING

ID	Type	Cardinality	Description
name	string	1	Name of parameter group.
parameter	list	0..n	List of parameters to the service primitive. See "nsd:parameter-group:parameter" on page 101.
mandatory	boolean	1	[Default *true*] Specifies whether this group mandatory.

nsd:parameter-group:parameter

/nsd:nsd-catalog/nsd/STRING/service-primitive/STRING/parameter-group/STRING/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of parameter.
data-type	enum	1	Data type associated with the name. **Supported values:** • STRING • INTEGER • BOOLEAN
mandatory	boolean	1	[Default is *false*] Specifies whether this field is mandatory.
default-value	string	1	The default value for the field.

ID	Type	Cardinality	Description
parameter-pool	string	1	NSD parameter pool name to use for this parameter.
read-only	boolean	1	The value should be dimmed by the UI. Applies only to parameters with default values.
hidden	boolean	1	The value should be hidden by the UI. Applies only to parameters with default values
out	boolean	1	[Default is *false*] Specifies if this is an output of the primitive execution

nsd:vnf-primitive-group

/nsd:nsd-catalog/nsd/STRING/service-primitive/STRING/vnf-primitive-group/0

ID	Type	Cardinality	Description
member-vnf-index-ref	uint64	1	Reference to member-vnf within constituent-vnfds.
vnfd-id-ref	string	1	A reference to a VNFD. This is a leafref to path: ../../../../nsd:constituent-vnfd + [nsd:id = current()/../nsd:id-ref] + /nsd:vnfd-id-ref
vnfd-name	string	1	Name of the VNFD.

ID	Type	Cardinality	Description
primitive	list	1	A list of VNF primitives. See "nsd:vnf-primitive-group:primitive" on page 103

nsd:vnf-primitive-group:primitive

/nsd:nsd-catalog/nsd/STRING/service-primitive/STRING/vnf-primitive-group/0/primitive/0

ID	Type	Cardinality	Description
index	uint32	1	Index of this primitive.
name	string	1	Name of the primitive in the VNF primitive.

nsd:initial-config-primitive

Initial set of configuration primitives for the NSD, that are to be executed when the network service comes up.

REST URI path

/nsd:nsd-catalog/nsd/STRING/initial-config-primitive/0

Fields

ID	Type	Cardinality	Description
seq	uint64	1	Sequence number for the configuration primitive.
name	string	1	[Required] Name of the configuration primitive.
user-defined-script	string	1	A user-defined script.
parameter	list	0..n	List of parameters to the initial-config-primitive. See "nsd:parameter" on page 104.

nsd:parameter

/nsd:nsd-catalog/nsd/STRING/initial-config-primitive/0/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the parameter.
value	string	1	Value of the parameter.

nsd:terminate-config-primitive

Set of configuration primitives to be executed before tearing down the network service.

REST URI path

/nsd:nsd-catalog/nsd/STRING/terminate-config-primitive/0

Fields

ID	Type	Cardinality	Description
seq	uint64	1	Sequence number for the configuration primitive.
name	string	1	[Required] Name of the configuration primitive.
user-defined-script	string	1	A user-defined script.
parameter	list	0..n	List of parameters to the primitive. See "nsd:parameter" on page 105.

nsd:parameter

/nsd:nsd-catalog/nsd/STRING/terminate-config-primitive/0/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the parameter.
value	string	1	Value of the parameter.

nsd:key-pair

Used to configure the list of public keys to be injected as part of network service instantiation.

REST URI path

/nsd:nsd-catalog/nsd/STRING/key-pair/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of this key pair.
key	string	1	Key associated with this key pair.

nsd:user

List of users to be added through cloud-config.

REST URI path

/nsd:nsd-catalog/nsd/STRING/user/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the user.
user-info	string	1	The user name's real name.
key-pair	string	1	Used to configure the list of public keys to be injected as part of network service instantiation. See "nsd:key-pair" on page 107.

nsd:key-pair

/nsd:nsd-catalog/nsd/STRING/user/STRING/key-pair/STRING

ID	Type	Cardinality	Description
name	string	1	Name of this key pair.
key	string	1	Key associated with this key pair.

NS Lifecycle Management (nsr:nsd)

REST wrapper for network service lifecycle management (nsr:ns-instance-config). Provides methods for instantiating, updating, finding, and terminating a network service (NS). Also provides methods for creating, updating, listing, and deleting or VNF forwarding graph (VNFFG).

Methods and relative URLs

Method	Relative URL	Description
POST	/api/config/ns-instance-config	Instantiate a network service.
GET	/api/operational/ns-instance-opdata/nsr/{nsr-id}?deep	Monitor the instantiated network service.
PUT	/api/running/ns-instance-config/{nsr-id}	Update a network service.
DELETE	/api/running/ns-instance-config/nsr{UUID}	Terminate a network service record.

Fields

ID	Type	Cardinality	Description
id	string	1	Identifier of the network service record (NSR).
name	string	1	NSR name.
short-name	string	1	NSR short name (for display on UI).
description	string	1	Description of the NSR.

ID	Type	Cardinality	Description
admin-status	enum	1	Administrative status of the NS instance. Value can be: • ENABLED (default) • DISABLED
nsd	container	1	Network service descriptor used to instantiate this network service. See "nsr:nsd" on page 116.
input-parameter	list	0..n	List of XPath input parameters. See "nsr:input-parameter" on page 145.
nsd-placement-group-maps	list	0..n	Mapping from mano-placement groups construct from NSD to cloud platform placement group construct. See "nsr:nsd-placement-group-maps" on page 146.
vnfd-placement-group-maps	list	0..n	Mapping from mano-placement groups construct from VNFD to cloud platform placement group construct. See "nsr:vnfd-placement-group-maps" on page 149.
ssh-authorized-key	list	0..n	List of authorized ssh keys as part of cloud-config. See "nsr:ssh-authorized-key" on page 152.
user	list	0..n	List of users to be added through cloud-config. See "nsr:user" on page 153.

MANO reference points

Os-Ma-nfvo

Schema

```
module nsr
 {
   namespace "urn:ietf:params:xml:ns:yang:nfvo:nsr";
   prefix "nsr";

   import rw-pb-ext {
     prefix "rwpb";
   }

   import vlr {
     prefix "vlr";
   }

   import vld {
     prefix "vld";
   }

   import nsd {
     prefix "nsd";
   }

   import vnfd {
     prefix "vnfd";
   }

   import vnfr {
     prefix "vnfr";
   }

   import ietf-inet-types {
     prefix "inet";
   }

   import ietf-yang-types {
     prefix "yang";
   }

   import mano-types {
     prefix "manotypes";
   }

   import rw-sdn {
     prefix "rwsdn";
   }

   revision 2015-09-10 {
     description
       "Initial revision. This YANG file defines
        the Network Service Record (NSR)";
     reference
       "Derived from earlier versions of base YANG files";
   }

   typedef config-states {
```

```
     type enumeration {
       enum init;
       enum configuring;
       enum config_not_needed;
       enum configured;
       enum failed;
       enum terminate;
     }
   }

   typedef trigger-type {
     type enumeration {
       enum ns-primitive;
       enum vnf-primitive;
     }
   }

   grouping cloud-config {
     description "List of cloud config parameters";

     list ssh-authorized-key {
       key "key-pair-ref";

       description "List of authorized ssh keys as part of cloud-config";

       leaf key-pair-ref {
         description "A reference to the key pair entry in the global key
pair table";
         type leafref {
           path "/nsr:key-pair/nsr:name";
         }
       }
     }
     list user {
       key "name";

       description "List of users to be added through cloud-config";
       leaf name {
         description "Name of the user ";
         type string;
       }
       leaf user-info {
         description "The user name's real name";
         type string;
       }
       list ssh-authorized-key {
         key "key-pair-ref";

         description "Used to configure the list of public keys to be
injected as part
                          of ns instantiation";

         leaf key-pair-ref {
           description "A reference to the key pair entry in the global key
pair table";
           type leafref {
```

```
            path "/nsr:key-pair/nsr:name";
          }
        }
      }
    }
  }

  list key-pair {
    key "name";
    description "Used to configure the list of public keys to be injected as
part
                 of ns instantiation";
    leaf name {
      description "Name of this key pair";
      type string;
    }

    leaf key {
      description "Key associated with this key pair";
      type string;
    }
  }

  grouping event-config-primitive {
    leaf seq {
      description
          "Sequence number for the configuration primitive.";
      type uint64;
    }

    leaf name {
      description
          "Name of the configuration primitive.";
      type string;
      mandatory "true";
    }

    leaf user-defined-script {
      description
          "A user defined script.";
      type string;
    }

    list parameter {
      key "name";
      leaf name {
        type string;
      }

      leaf value {
        type string;
      }
    }
  }

  rpc start-network-service {
```

```
    description "Start the network service";
    input {
      leaf name {
        mandatory true;
        description "Name of the Network Service";
        type string;
      }
      leaf nsd-ref {
        description "Reference to NSR ID ref";
        mandatory true;
        type leafref {
          path "/nsd:nsd-catalog/nsd:nsd/nsd:id";
        }
      }
      uses ns-instance-config-params;
    }

    output {
      leaf nsr-id {
        description "Automatically generated parameter";
        type yang:uuid;
      }
    }
  }



  container ns-instance-config {

    list nsr {
      key "id";
      unique "name";

      leaf id {
        description "Identifier for the NSR.";
        type yang:uuid;
      }

      leaf name {
        description "NSR name.";
        type string;
      }

      leaf short-name {
        description "NSR short name.";
        type string;
      }

      leaf description {
        description "NSR description.";
        type string;
      }

      leaf admin-status {
        description
          "This is the administrative status of the NS instance";
```

```
        type enumeration {
          enum ENABLED;
          enum DISABLED;
        }
      }

      container nsd {
        description "NS descriptor used to instantiate this NS";
        uses nsd:nsd-descriptor;
      }

      uses ns-instance-config-params;
    }
  }

  grouping ns-instance-config-params {
    uses manotypes:input-parameter;


    list nsd-placement-group-maps {
      description
          "Mapping from mano-placement groups construct from NSD to cloud
           platform placement group construct";

      key "placement-group-ref";

      leaf placement-group-ref {
        description
          "Reference for NSD placement group";
        // type leafref {
        //   path "../../nsd/placement-groups/name";
        // }
        type string;
      }
      uses manotypes:placement-group-input;
    }

    list vnfd-placement-group-maps {
      description
          "Mapping from mano-placement groups construct from VNFD to cloud
          platform placement group construct";

    key "placement-group-ref vnfd-id-ref";

    leaf vnfd-id-ref {
      description
          "A reference to a vnfd. This is a
          leafref to path:
          ../../../../nsd:constituent-vnfd
          + [nsr:id = current()/../nsd:id-ref]
          + /nsd:vnfd-id-ref
          NOTE: An issue with confd is preventing the
          use of xpath. Seems to be an issue with leafref
          to leafref, whose target is in a different module.
          Once that is resovled this will switched to use
```

```
          leafref";
      type yang:uuid;
    }

      leaf placement-group-ref {
        description
            "A reference to VNFD placement group";
        type leafref {
          path "/vnfd:vnfd-catalog/vnfd:vnfd[vnfd:id = current()/" +
              "../nsr:vnfd-id-ref]/vnfd:placement-groups/vnfd:name";
        }
      }

      uses manotypes:placement-group-input;
    }

    uses cloud-config;
  }



   }
```

Examples

See "API Examples" on page 346

nsr:nsd

Network service descriptor (NSD) used to instantiate this network service (NS).

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd

Fields

ID	Type	Cardinality	Description
id	string	1	Unique identifier for the NSD.
name	string	1	NSD name.
short-name	string	1	NSD short name to use as a label in the UI.
vendor	string	1	Vendor of the NSD.
logo	string	1	File path of the vendor-specific logo. For example, icons/mylogo.png The logo should be part of the network service package. SVG format is preferred, but PNG is supported. Although there is no hard limit on size and dimension, a square image under 200px by 200px is preferred.
description	string	1	Description of the NSD.
version	string	1	Version of the NSD.

ID	Type	Cardinality	Description
connection-point	list	0..n	A list of references to network service connection points. See "nsr:connection-point" on page 119.
vld	list	0..n	List of Virtual Link Descriptors (VLDs). See "nsr:vld" on page 120.
constituent-vnfd	list	0..n	List of Virtual Network Function Descriptors (VNFDs) that are part of this network service. See "nsr:constituent-vnfd" on page 124.
placement-groups	list	0..n	List of placement groups at the NS level. See "nsr:placement-groups" on page 125.
ip-profiles-list	list	0..n	List of IP profiles. See "nsr:ip-profiles" on page 126.
vnf-dependency	list	0..n	List of VNF dependencies. See "nsr:vnf-dependency" on page 129.
monitoring-param	list	0..n	List of monitoring parameters at the network service level. See "nsr:monitoring-param" on page 129.
input-parameter-xpath	list	0..n	List of XPaths to parameters inside the NSD that can be customized during instantiation. See "nsr:input-parameter-xpath" on page 134

ID	Type	Cardinality	Description
parameter-pool	list	0..n	Pool of parameter values that must be pulled from during configuration. See "nsr:parameter-pool" on page 135.
service-primitive	list	0..n	Network service level configuration primitives. See "nsr:service-primitive" on page 136.
initial-config-primitive	list	0..n	Set of configuration primitives to be executed when the network service comes up. See "nsr:initial-config-primitive" on page 141
terminate-config-primitive	list	0..n	Set of configuration primitives to be executed before during termination of the network service. See "nsr:terminate-config-primitive" on page 142.
key-pair	list	0..n	Used to configure the list of public keys to be injected as part of network service instantiation. See "nsr:key-pair" on page 143.
user	list	0..n	List of users to be added through cloud-config. See "nsr:user" on page 144.

nsr:connection-point

A list of references to network service connection points.

Each network service (NS):

- Has one or more external connection points used to link the NS to other NS or to external networks.
- Exposes these connection points to the orchestrator.

The orchestrator can construct network service chains by joining the connection points between different network services.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/connection-point/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the NS external connection point.
type	enum	1	Type of connection point. **Supported types:** VPORT: Virtual Port

nsr:vld

List of Virtual Link Descriptors (VLDs)

Network connections are defined by connection points and virtual links. There are three types of connection points:

- Connect a network service to the outside world, such as the network service endpoint, described in the NSD
- Connect between **VNFs** within a network service, such as the external interface of the VNF, described in the VNFD
- Connect between VMs, described in the **VNFC**

There are also two types of virtual links:

- External virtual links, which can be connected to network service endpoints and external VNF interfaces
- Internal virtual links, which can be connected to external VNF interfaces and VNFCs

Virtual links also follow the Metro Ethernet Forum **E-LINE**, **E-TREE**, and **E-LAN** services. Virtual link descriptors (VLDs) contain the bandwidth and **QoS** requirements of the interconnection.

VLDs are required for a functioning NSD.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/vld/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Unique identifier for the VLD.
name	string	1	VLD name.
short-name	string	1	**NSD** short name to use as label in the UI.
vendor	string	1	Provider of the VLD.

ID	Type	Cardinality	Description
description	string	1	Description of the VLD.
version	string	1	Version of the VLD.
type	enum	1	Type of the virtual link. **Supported values:** **ELAN**: A multipoint service connecting a set of VNFs.
root-bandwidth	uint64	1	For ELAN this is the aggregate bandwidth.
leaf-bandwidth	uint64	1	For ELAN this is the bandwidth of branches.
vnfd-connection-point-ref	list	0..n	A list of references to connection points. See "nsr:vnfd-connection-point-ref" on page 122.
provider-network	container	1	Container for the provider network. See "nsr:provider-network " on page 123.
mgmt-network	boolean	1	[Default *true*] Denotes whether this network is a **VIM** management network.
vim-network-name	string		Name of network in VIM account. This field indicates pre-provisioned network name in cloud account.
ip-profile-ref	string	1	Named reference to IP-profile object.

nsr:vnfd-connection-point-ref

/nsr:ns-instance-config/nsr/STRING/nsd/vld/STRING/vnfd-connection-point-ref/0,STRING

ID	Type	Cardinality	Description
member-vnf-index-ref	reference	1	Reference to member-vnf within constituent units. This is a leafref to path: ../../../nsd:constituent-vndf/nsd:member-vnf-index
vnf-id-ref	string	1	Reference to a VNFD. This is a leafref to path: ../../../nsd:constituent-vnfd + [nsd:id = current()/../nsd:id-ref] + /nsd:vnfd-id-ref
vnfd-connection-point-ref	string	1	Reference to a connection point name in a VNFD. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd + [vnfd:id = current()/../nsd:vnfd-id-ref] + /vnfd:connection-point/vnfd:name

nsr:provider-network

/nsr:ns-instance-config/nsr/STRING/nsd/vld/STRING/provider-network

ID	Type	Cardinality	Description
physical-network	string	1	Name of the physical network on which the provider network is built.
overlay-type	enum	0..1	Identifies the type of the overlay network, which is a virtual network that is built on top of an existing network and is supported by its infrastructure. **Supported values:** • LOCAL — A network that can be realized on a single host only. • FLAT — The simplest networking environment in which each instance receives a fixed IP from the pool. All instances are attached to the same bridges. • VLAN — A network of computers in which the computers behaves as if they are connected to the same wire. However, the computers might be physically located on different segments of a LAN. • VXLAN — A proposed encapsulation protocol for running an overlay network on existing **Layer 3** infrastructure • GRE — GRE tunnels encapsulate isolated **Layer 2** network traffic in IP packets. Packets are routed between compute and networking nodes using the hosts' network connectivity and routing tables.
segmentation-id	uint32	1	Segmentation ID.

nsr:constituent-vnfd

A list of Virtual Network Function Descriptors (VNFDs) that are part of this network service.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/constituent-vnfd/0

Fields

ID	Type	Cardinality	Description
member-vnfd-index	uint64	1	[Required] Identifier/index for the VNFD. **Note:** This separate ID is required so that multiple VNFs can be part of a single network service.
vnfd-id-ref	leafref	1	Identifier for the VNFD. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id
start-by-default	boolean	1	[Default *true*] VNFD is started as part of network service instantiation.

nsr:placement-groups

A list of placement groups at the network service level.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/placement-groups/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Place group construct to define the compute resource placement strategy in cloud environment.
requirement	string	1	Describes the intent/rationale behind this placement group. **Note:** This free-text field is for human consumption only
strategy	enum	1	Strategy associated with this placement group. **Supported values:** • COLOCATION: [Default] Share the physical infrastructure (hypervisor/network) among all members of this group. • ISOLATION: Do not share the physical infrastructure (hypervisor/network) among the members of this group
member-vnfd	list	0..n	List of VNFDs that are part of this placement group. See "nsr:member-vnfd" on page 126.

nsr:member-vnfd

/nsr:ns-instance-config/nsr/STRING/nsd/placement-groups/STRING/member-vnfd/0

ID	Type	Cardinality	Description
member-vnf-index-ref	leafref	1	Member VNF index of this member VNF. This is a leafref to path: ../../../constituent-vnfd/member-vnf-index
vnfd-id-ref	leafref	1	Identifier for the VNFD. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id

nsr:ip-profiles

List of IP profiles. IP profiles describe the IP characteristics for the virtual link.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/ip-profiles/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the IP profile.
description	string	1	Description of the IP profile.
ip-profile-params	container	1	Information about the IP profile. See "nsr:ip-profile-params" on page 127.

nsr:ip-profile-params

/nsr:ns-instance-config/nsr/STRING/nsd/ip-profiles/STRING/ip-profile-params

ID	Type	Cardinality	Description
ip-version	enum	1	[Default IPv4] Version of the Internet Protocol used.
subnet-address	union	1	Subnet IP prefix associated with this IP profile.
gateway-address	union	1	IP address of the default gateway associated with this IP profile.
security-group	string	1	Name of the security group.
dns-server	list	0..n	List of **DNS** servers associated with this IP profile. See "nsr:dns-server" on page 128.
dhcp-params	container	1	Container for **DHCP** parameters. See "nsr:ip-profile-params" on page 127.
subnet-prefix-pool	string	1	**VIM**-specific reference to pre-created subnet prefix.

nsr:dns-server

/nsr:ns-instance-config/nsr/STRING/nsd/ip-profiles/STRING/ip-profile-params/dns-server/UNION_VALUE

ID	Type	Cardinality	Description
address	union	1	List of DNS servers associated with this IP profile.

nsr:dhcp-params

/nsr:ns-instance-config/nsr/STRING/nsd/ip-profiles/STRING/ip-profile-params/dhcp-params

ID	Type	Cardinality	Description
enabled	boolean	1	[Default *true*] Indicates if DHCP is enabled.
start-address	union	1	Start IP address of the IP address range associated with DHCP domain.
count	uint32	1	Size of the DHCP pool associated with DHCP domain.

nsr:vnf-dependency

List of **VNF** dependencies in the Network Service Descriptor.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/vnf-dependency/STRING

Fields

ID	Type	Cardinality	Description
vnf-dependency	leafref	1	List of VNF dependencies. This is a leafref to path: ../../../vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id
vnf-depends-on-ref	leafref	1	Reference to VNFD that on which the source VNF depends. This is a leafref to path: ../../../vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id

nsr:monitoring-param

List of monitoring parameters from **VNFs** to propagate to the Network Service Record (NSR).

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/monitoring-param/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Identifier for the parameter.

ID	Type	Cardinality	Description
name	string	1	Name of the monitoring parameter
value-type	enum	1	The type of the parameter value. **Supported values:** • INT (default) • DECIMAL • STRING
value-type	enum	1	The type of the parameter value. **Supported values:** • INT (default) • DECIMAL • STRING
numeric-constraints	container	1	Constraints for the numbers. See "nsr:numeric-constraints" on page 131.
text-constraints	container	1	Constraints for the string. See "nsr:text-constraints" on page 132.
value-integer	int64	1	Current value for integer parameter.
value-decimal	decimal164	1	Current value for decimal parameter.
value-string	string	1	Current value for the string parameter.
description	string	1	
group-tag	string	1	A simple tag to group monitoring parameters

ID	Type	Cardinality	Description
widget-type	enum	1	Type of the widget, typically used by the UI. **Supported values:** • HISTOGRAM • BAR • GAUGE • SLIDER • COUNTER • TEXTBOX
aggregation-type	enum	1	Aggregation type for the monitoring parameter. **Supported values:** • AVERAGE • MINIMUM • MAXIMUM • COUNT • SUM
vnfd-monitoring-param	list	0..n	A list of VNFD monitoring parameters. See "nsr:vnfd-monitoring-param" on page 132.

nsr:numeric-constraints

/nsr:ns-instance-config/nsr/STRING/nsd/monitoring-param/STRING/numeric-constraints

ID	Type	Cardinality	Description
min-value	uint64	1	Minimum value for the parameter.

ID	Type	Cardinality	Description
max-value	uint64	1	Maximum value for the parameter.

nsr:text-constraints

/nsr:ns-instance-config/nsr/STRING/nsd/monitoring-param/STRING/text-constraints

ID	Type	Cardinality	Description
min-length	uint8	1	Minimum string length for the parameter.
max-length	uint8	1	Maximum string length for the parameter.

nsr:vnfd-monitoring-param

/nsr:ns-instance-config/nsr/STRING/nsd/monitoring-param/STRING/vnfd-monitoring-param/0,STRING

ID	Type	Cardinality	Description
vnfd-id-ref	leafref	1	A reference to a VNFD. This is a leafref to path: ../../../../nsd:constituent-vnfd + [nsd:id = current()/../nsd:id-ref] + /nsd:vnfd-id-ref

ID	Type	Cardinality	Description
vnfd-monitoring-param-ref	leafref	1	A reference to the VNFD monitoring parameter. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd + [vnfd:id = current()/../vnfd-id-ref] + /vnfd:monitoring-param/vnfd:id
member-vnf-index-ref	leafref	1	Optional reference to member-vnf within constituent-vnfds.

nsr:input-parameter-xpath

List of **XPaths** to parameters inside the Network Service Descriptor that can be customized during instantiation.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/input-parameter-xpath/STRING

Fields

ID	Type	Cardinality	Description
xpath	string	1	An xpath that specifies the element in a descriptor.
label	string	1	A descriptive string.
default-value	string	1	A default value for this input parameter.

nsr:parameter-pool

Pool of parameter values from which to pull during configuration.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/parameter-pool/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the configuration value pool.
range	container	1	Create a range of values from which to populate the pool. See "nsr:range" on page 135.

nsr:range

/nsr:ns-instance-config/nsr/STRING/nsd/parameter-pool/STRING/range

ID	Type	Cardinality	Description
start-value	uint32	1	[Required] Generated pool values start at this value.
end-value	uint32	1	[Required] Generated pool values end at this value.

nsr:service-primitive

Network service-level service primitives for the Network Service Descriptor.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/service-primitive/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the service primitive.
parameter	list	0..n	List of parameters for the service primitive. See "nsr:parameter" on page 136.
parameter-group	list	0..n	Grouping of parameters that are logically grouped in UI. See "nsr:parameter-group" on page 138.
vnf-primitive-group	list	0..n	List of service primitives grouped by the VNF. See "nsr:vnf-primitive-group" on page 139.
used-defined-script	string	1	A user-defined script.

nsr:parameter

/nsr:ns-instance-config/nsr/STRING/nsd/service-primitive/STRING/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of parameter.

ID	Type	Cardinality	Description
data-type	enum	1	Data type associated with the name. **Supported values:** • STRING • INTEGER • BOOLEAN
mandatory	boolean	1	[Default is *false*] Specifies whether this field is mandatory.
default-value	string	1	The default value for the field.
parameter-pool	string	1	NSD parameter pool name to use for this parameter.
read-only	boolean	1	The value should be dimmed by the UI. Applies only to parameters with default values.
hidden	boolean	1	The value should be hidden by the UI. Applies only to parameters with default values
out	boolean	1	[Default is *false*] Specifies if this is an output of the primitive execution

nsr:parameter-group

/nsr:ns-instance-config/nsr/STRING/nsd/service-primitive/STRING/parameter-group/STRING

ID	Type	Cardinality	Description
name	string	1	Name of parameter group.
parameter	list	0..n	List of parameters to the service primitive. See "nsr:parameter-group:parameter" on page 138.
mandatory	boolean	1	[Default *true*] Specifies whether this group mandatory.

nsr:parameter-group:parameter

/nsr:ns-instance-config/nsr/STRING/nsd/service-primitive/STRING/parameter-group/STRING/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of parameter.
data-type	enum	1	Data type associated with the name. **Supported values:** • STRING • INTEGER • BOOLEAN
mandatory	boolean	1	[Default is *false*] Specifies whether this field is mandatory.
default-value	string	1	The default value for the field.

ID	Type	Cardinality	Description
parameter-pool	string	1	NSD parameter pool name to use for this parameter.
read-only	boolean	1	The value should be dimmed by the UI. Applies only to parameters with default values.
hidden	boolean	1	The value should be hidden by the UI. Applies only to parameters with default values
out	boolean	1	[Default is *false*] Specifies if this is an output of the primitive execution

nsr:vnf-primitive-group

/nsr:ns-instance-config/nsr/STRING/nsd/service-primitive/STRING/vnf-primitive-group/0

ID	Type	Cardinality	Description
member-vnf-index-ref	uint64	1	Reference to member-vnf within constituent-vnfds.
vnfd-id-ref	string	1	A reference to a VNFD. This is a leafref to path: ../../../../nsd:constituent-vnfd + [nsd:id = current()/../nsd:id-ref] + /nsd:vnfd-id-ref
vnfd-name	string	1	Name of the VNFD.

ID	Type	Cardinality	Description
primitive	list	1	A list of VNF primitives. See "nsr:vnf-primitive-group:primitive" on page 140

nsr:vnf-primitive-group:primitive

/nsr:ns-instance-config/nsr/STRING/nsd/service-primitive/STRING/vnf-primitive-group/0/primitive/0

ID	Type	Cardinality	Description
index	uint32	1	Index of this primitive.
name	string	1	Name of the primitive in the VNF primitive.

nsr:initial-config-primitive

Initial set of configuration primitives for the NSD, that are to be executed when the network service comes up.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/initial-config-primitive/0

Fields

ID	Type	Cardinality	Description
seq	uint64	1	Sequence number for the configuration primitive.
name	string	1	[Required] Name of the configuration primitive.
user-defined-script	string	1	A user-defined script.
parameter	list	0..n	List of parameters to the initial-config-primitive. See "nsr:parameter" on page 141.

nsr:parameter

/nsr:ns-instance-config/nsr/STRING/nsd/initial-config-primitive/0/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the parameter.
value	string	1	Value of the parameter.

nsr:terminate-config-primitive

Set of configuration primitives to be executed before tearing down the network service.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/terminate-config-primitive/0

Fields

ID	Type	Cardinality	Description
seq	uint64	1	Sequence number for the configuration primitive.
name	string	1	[Required] Name of the configuration primitive.
user-defined-script	string	1	A user-defined script.
parameter	list	0..n	List of parameters to the primitive. See "nsr:parameter" on page 142.

nsr:parameter

/nsr:ns-instance-config/nsr/STRING/nsd/terminate-config-primitive/0/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the parameter.
value	string	1	Value of the parameter.

nsr:key-pair

Used to configure the list of public keys to be injected as part of network service instantiation.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/key-pair/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of this key pair.
key	string	1	Key associated with this key pair.

nsr:user

List of users to be added through cloud-config.

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd/user/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the user.
user-info	string	1	The user name's real name.
key-pair	list	0..n	Used to configure the list of public keys to be injected as part of network service instantiation. See "nsr:key-pair" on page 144.

nsr:key-pair

/nsr:ns-instance-config/nsr/STRING/user/STRING/ssh-authorized-key/STRING

/nsr:ns-instance-config/nsr/STRING/nsd/user/STRING/key-pair/STRING

ID	Type	Cardinality	Description
key-pair-ref	leafref	1	A reference to the key-pair entry in the global key pair table.

nsr:input-parameter

List of **XPaths** input parameters.

REST URI path

/nsr:ns-instance-config/nsr/STRING/input-parameter/STRING

Fields

ID	Type	Cardinality	Description
xpath	string	1	An XPath that specifies which element in a descriptor is to be modified.
value	string	1	The value that the element specified by the XPath should take when a record is created.

nsr:nsd-placement-group-maps

Mapping from mano-placement groups construct from NSD to cloud platform placement group construct .

REST URI path

/nsr:ns-instance-config/nsr/STRING/nsd-placement-group-maps/STRING

Fields

ID	Type	Cardinality	Description
placement-group-ref	string	1	Reference for the NSD placement group.
cloud-type	enum	1	Cloud account type: • aws • cloudsim • cloudsim_proxy • mock • openmano • openstack • vsphere • openvim • prop_cloud1
availability-zone	container	1	Name of the availability zone. See "nsr:availability-zone" on page 147.
server-group	container	1	Name of the affinity/anti-affinity server group. See "nsr:server-group" on page 147.

ID	Type	Cardinality	Description
host-aggregate	list	0..n	Name of the host aggregate. See "nsr:host-aggregate" on page 148.
aws-construct	empty		
openmano-construct	empty		
vsphere-construct	empty		
mock-construct	empty		
cloudsim-construct	empty		

nsr:availability-zone

/nsr:ns-instance-config/nsr/STRING/nsd-placement-group-maps/STRING/availability-zone

ID	Type	Cardinality	Description
name	string	1	Name of the availability zone.

nsr:server-group

/nsr:ns-instance-config/nsr/STRING/nsd-placement-group-maps/STRING/server-group

ID	Type	Cardinality	Description
name	string	1	Name of the affinity/anti-affinity server group.

nsr:host-aggregate

/nsr:ns-instance-config/nsr/STRING/nsd-placement-group-maps/STRING/host-aggregate/STRING

ID	Type	Cardinality	Description
metadata-key	string	1	
metadata-value	string	1	

nsr:vnfd-placement-group-maps

Mapping from mano-placement groups construct from VNFD to cloud platform placement group construct.

REST URI path

/nsr:ns-instance-config/nsr/STRING/vnfd-placement-group-maps/STRING,STRING

Fields

ID	Type	Cardinality	Description
vnfd-id-ref	string	1	A reference to a vnfd. This is a leafref to path: ../../../../nsd:constituent-vnfd + [nsr:id = current()/../nsd:id-ref] + /nsd:vnfd-id-ref
placement-group-ref	leafref	1	A reference to VNFD placement group.
cloud-type	enum	1	Cloud account type: • aws • cloudsim • cloudsim_proxy • mock • openmano • openstack • vsphere • openvim • prop_cloud1
availability-zone	container	1	Name of the availability zone. See "nsr:availability-zone" on page 150.

ID	Type	Cardinality	Description
server-group	container	1	Name of the affinity/anti-affinity server group. See "nsr:server-group" on page 151.
host-aggregate	list	0..n	Name of the host aggregate. See "nsr:host-aggregate" on page 151.
aws-construct	empty		
openmano-construct	empty		
vsphere-construct	empty		
mock-construct	empty		
cloudsim-construct	empty		

nsr:availability-zone

/nsr:ns-instance-config/nsr/STRING/vnfd-placement-group-maps/STRING,STRING/availability-zone

ID	Type	Cardinality	Description
name	string	1	Name of the availability zone.

nsr:server-group

/nsr:ns-instance-config/nsr/STRING/vnfd-placement-group-maps/STRING,STRING/server-group

ID	Type	Cardinality	Description
name	string	1	Name of the affinity/anti-affinity server group.

nsr:host-aggregate

/nsr:ns-instance-config/nsr/STRING/vnfd-placement-group-maps/STRING,STRING/host-aggregate/STRING

ID	Type	Cardinality	Description
metadata-key	string	1	
metadata-value	string	1	

nsr:ssh-authorized-key

List of authorized SSH keys as part of cloud-config.

REST URI path

/nsr:ns-instance-config/nsr/STRING/ssh-authorized-key/STRING

Fields

ID	Type	Cardinality	Description
key-pair-ref	leafref	1	A reference to the key-pair entry in the global key pair table.

nsr:user

List of users to be added through cloud-config.

REST URI path

/nsr:ns-instance-config/nsr/STRING/user/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the user.
user-info	string	1	The user name's real name.
ssh-authorized-key	list	0..n	Used to configure the list of public keys to be injected as part of network service instantiation. See "nsr:ssh-authorized-key" on page 153

nsr:ssh-authorized-key

/nsr:ns-instance-config/nsr/STRING/user/STRING/ssh-authorized-key/STRING

ID	Type	Cardinality	Description
key-pair-ref	leafref	1	A reference to the key-pair entry in the global key pair table.

VNF Descriptor Management (vnfd:vnfd)

REST wrapper for VNFD descriptor management provides methods for onboarding, updating, querying, and deleting a VNF descriptor through the vnfd:vnfd-catalog.

The VNF includes one or more VDUs (the VM that hosts the network function), virtual links, and connection points. Each of these components (called nodes) has specific requirements, attributes, and capabilities, such as computational properties, that are defined in the VNF descriptor.

Methods and relative URLs

Method	Relative URL	Description
POST	/api/config/vnfd-catalog	Onboard a VNF package to the catalog.
PUT	/api/running/vnfd-catalog/vnfd/{UUID}	Update a VNFD. The orchestrator verifies that the VNFD is not in use by the NSD or by a running instances of a VNF.
GET	/api/running/vnfd-catalog/vnfd/{UUID}	Query a VNF descriptor.
DELETE	/api/running/vnfd-catalog/vnfd/{UUID}	Delete a VNF package. **Note:** The VNFD you want to delete cannot be in use by either the NSD or by a running instance of the VNF.

Fields

Descriptor details for the Virtual Network Function (**VNF**).

ID	Type	Cardinality	Description
id	string	1	Identifier for the **VNFD**.

ID	Type	Cardinality	Description
name	string	1	VNFD name.
short-name	string	1	VNFD short name to use as a label in the UI.
vendor	string	1	Provider of the VNFD.
logo	string	1	File path of the vendor-specific logo. For example, icons/mylogo.png The logo should be part of the VNF package. SVG format is preferred, but PNG is supported. Although there is no hard limit on size and dimension, a square image under 200px by 200px is preferred.
description	string	1	Description of the VNFD.
version	string	1	Version of the VNFD.
vnf-configuration	container		Information about the **VNF** configuration for the management interface. See "vnfd:vnf-configuration" on page 167.
mgmt-interface	container	1	Interface over which the VNF is managed. See "vnfd:mgmt-interface" on page 174.
internal-vld	list	0..n	List of internal Virtual Link Descriptors (**VLD**). See "vnfd:internal-vld" on page 176.
ip-profiles	list	0..n	List of IP profiles. An IP profile describes the IP characteristics for the virtual-link. See "vnfd:ip-profiles" on page 179.

ID	Type	Cardinality	Description
connection-point	list	0..n	The list for external connection points. See "vnfd:connection-point" on page 182.
vdu	list	0..n	List of virtual deployment units, which are VMs that host the network function. See "vnfd:vdu" on page 183
vdu-dependency	list	0..n	List of VDU dependencies. See "vnfd:vdu-dependency" on page 214.
service-function-chain	enum	1	Type of node in service function chaining architecture. **Supported types:** • UNAWARE (default) • CLASSIFIER • SF • SFF
service-function-type	string	1	Type of service function. **Note:** This value needs to map to a service function type in the OpenDaylight platform to support VNFFG.
http-endpoint	list	0..n	List of http endpoints to be used by monitoring-param. See "vnfd:http-endpoint" on page 215.
monitoring-param	list	0..n	List of monitoring parameters at the network service level. See vnfd:monitoring-param.

ID	Type	Cardinality	Description
placement-groups	list	0..n	List of placement groups at VNF level. See "vnfd:placement-groups" on page 217.

MANO reference points

Os-Ma-nfvo and Or-Vnfm

Schema

```
module vnfd
 {
   namespace "urn:ietf:params:xml:ns:yang:nfvo:vnfd";
   prefix "vnfd";

   import mano-types {
     prefix "manotypes";
   }

   import rw-pb-ext {
     prefix "rwpb";
   }

   import ietf-yang-types {
     prefix "yang";
   }

   import ietf-inet-types {
     prefix "inet";
   }

   revision 2015-09-10 {
     description
       "Initial revision. This YANG file defines
        the Virtual Network Function (VNF)";
     reference
       "Derived from earlier versions of base YANG files";
   }

   grouping common-connection-point {
     leaf name {
       description "Name of the connection point";
       type string;
     }

     leaf id {
       description "Identifier for the internal connection points";
       type string;
     }
```

```
    leaf short-name {
      description "Short name of the connection point";
      type string;
    }

    leaf type {
      description "Type of the connection point.";
      type manotypes:connection-point-type;
    }
  }

  grouping virtual-interface {
    container virtual-interface {
      description
          "Container for the virtual interface properties";

      leaf type {
        description
            "Specifies the type of virtual interface
             between VM and host.
             VIRTIO          : Use the traditional VIRTIO interface.
             PCI-PASSTHROUGH : Use PCI-PASSTHROUGH interface.
             SR-IOV          : Use SR-IOV interface.
             E1000           : Emulate E1000 interface.
             RTL8139         : Emulate RTL8139 interface.
             PCNET           : Emulate PCNET interface.
             OM-MGMT         : Used to specify openmano mgmt external-
connection type";

        type enumeration {
          enum OM-MGMT;
          enum PCI-PASSTHROUGH;
          enum SR-IOV;
          enum VIRTIO;
          enum E1000;
          enum RTL8139;
          enum PCNET;
        }
        default "VIRTIO";
      }

      leaf vpci {
        description
            "Specifies the virtual PCI address. Expressed in
             the following format dddd:dd:dd.d. For example
             0000:00:12.0. This information can be used to
             pass as metadata during the VM creation.";
        type string;
      }

      leaf bandwidth {
        description
            "Aggregate bandwidth of the NIC.";
        type uint64;
      }
```

```
    }
  }

  container vnfd-catalog {

    description
        "Virtual Network Function Descriptor (VNFD).";

    list vnfd {
      key "id";

      leaf id {
        description "Identifier for the VNFD.";
        type string;
      }

      leaf name {
        description "VNFD name.";
        mandatory true;
        type string;
      }

      leaf short-name {
        description "VNFD short name.";
        type string;
      }

      leaf vendor {
        description "Vendor of the VNFD.";
        type string;
      }

      leaf logo {
        description
            "Vendor logo for the Virtual Network Function";
        type string;
      }

      leaf description {
        description "Description of the VNFD.";
        type string;
      }

      leaf version {
        description "Version of the VNFD";
        type string;
      }

      uses manotypes:vnf-configuration;

      container mgmt-interface {
        description
            "Interface over which the VNF is managed.";

        choice endpoint-type {
          description
```

```
              "Indicates the type of management endpoint.";

            case ip {
              description
                  "Specifies the static IP address for managing the VNF.";
              leaf ip-address {
                type inet:ip-address;
              }
            }

            case vdu-id {
              description
                  "Use the default management interface on this VDU.";
              leaf vdu-id {
                type leafref {
                  path "/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:vdu/vnfd:id";
                }
              }
            }

            case cp {
              description
                  "Use the ip address associated with this connection point.";
              leaf cp {
                type leafref {
                  path "/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:connection-
point/vnfd:name";
                }
              }
            }
          }

          leaf port {
            description
                "Port for the management interface.";
            type inet:port-number;
          }

          container dashboard-params {
            description "Parameters for the VNF dashboard";

            leaf path {
              description "The HTTP path for the dashboard";
              type string;
            }

            leaf https {
              description "Pick HTTPS instead of HTTP , Default is false";
              type boolean;
            }

            leaf port {
              description "The HTTP port for the dashboard";
              type inet:port-number;
            }
          }
```

```
      }

      list internal-vld {
        key "id";
        description
            "List of Internal Virtual Link Descriptors (VLD).
            The internal VLD describes the basic topology of
            the connectivity (e.g. E-LAN, E-Line, E-Tree)
            between internal VNF components of the system.";

        leaf id {
          description "Identifier for the VLD";
          type string;
        }

        leaf name {
          description "Name of the internal VLD";
          type string;
        }

        leaf short-name {
          description "Short name of the internal VLD";
          type string;
        }

        leaf description {
          type string;
        }

        leaf type {
          type manotypes:virtual-link-type;
        }

        leaf root-bandwidth {
          description
              "For ELAN this is the aggregate bandwidth.";
          type uint64;
        }

        leaf leaf-bandwidth {
          description
              "For ELAN this is the bandwidth of branches.";
          type uint64;
        }

        list internal-connection-point {
          key "id-ref";
          description "List of internal connection points in this VLD";
          leaf id-ref {
            description "reference to the internal connection point id";
            type leafref {
              path "../../../vdu/internal-connection-point/id";
            }
          }
        }
        uses manotypes:provider-network;
```

```
      }

      list connection-point {
        key "name";
        description
            "List for external connection points. Each VNF has one
            or more external connection points. As the name
            implies that external connection points are used for
            connecting the VNF to other VNFs or to external networks.
            Each VNF exposes these connection points to the
            orchestrator. The orchestrator can construct network
            services by connecting the connection points between
            different VNFs. The NFVO will use VLDs and VNFFGs at
            the network service level to construct network services.";

        uses common-connection-point;
      }

      list vdu {
        description "List of Virtual Deployment Units";
        key "id";

        leaf id {
          description "Unique id for the VDU";
          type string;
        }

        leaf name {
          description "Unique name for the VDU";
          type string;
        }

        leaf description {
            description "Description of the VDU.";
            type string;
        }

        leaf count {
          description "Number of instances of VDU";
          type uint64;
        }

        leaf mgmt-vpci {
          description
              "Specifies the virtual PCI address. Expressed in
             the following format dddd:dd:dd.d. For example
             0000:00:12.0. This information can be used to
             pass as metadata during the VM creation.";
          type string;
        }

        uses manotypes:vm-flavor;
        uses manotypes:guest-epa;
        uses manotypes:vswitch-epa;
        uses manotypes:hypervisor-epa;
        uses manotypes:host-epa;
```

```
        list alarm {
          key "alarm-id";

          uses manotypes:alarm;
        }

        uses manotypes:image-properties;

        choice cloud-init-input {
          description
            "Indicates how the contents of cloud-init script are provided.
             There are 2 choices - inline or in a file";

          case inline {
            leaf cloud-init {
              description
                "Contents of cloud-init script, provided inline, in cloud-
config format";
              type string;
            }
          }

          case filename {
            leaf cloud-init-file {
              description
                "Name of file with contents of cloud-init script in cloud-
config format";
                type string;
            }
          }
        }

        list internal-connection-point {
          key "id";
          description
              "List for internal connection points. Each VNFC
              has zero or more internal connection points.
              Internal connection points are used for connecting
              the VNF components internal to the VNF. If a VNF
              has only one VNFC, it may not have any internal
              connection points.";

          uses common-connection-point;

          leaf internal-vld-ref {
            type leafref {
              path "../../../internal-vld/id";
            }
          }
        }

        list internal-interface {
          description
              "List of internal interfaces for the VNF";
          key name;
```

```
        leaf name {
          description
              "Name of internal interface. Note that this
              name has only local significance to the VDU.";
          type string;
        }

        leaf vdu-internal-connection-point-ref {
          type leafref {
            path "../../internal-connection-point/id";
          }
        }
        uses virtual-interface;
      }

      list external-interface {
        description
            "List of external interfaces for the VNF.
            The external interfaces enable sending
            traffic to and from VNF.";
        key name;

        leaf name {
          description
              "Name of the external interface. Note that
              this name has only local significance.";
          type string;
        }

        leaf vnfd-connection-point-ref {
          description
            "Name of the external connection point.";
          type leafref {
            path "../../../connection-point/name";
          }
        }
        uses virtual-interface;
      }

      list volumes {
        key "name";

        leaf name {
          description "Name of the disk-volumes, e.g. vda, vdb etc";
          type string;
        }

        uses manotypes:volume-info;
      }
    }

    list vdu-dependency {
      description
          "List of VDU dependencies.";
```

```
          key vdu-source-ref;
          leaf vdu-source-ref {
            type leafref {
              path "../../vdu/id";
            }
          }

          leaf vdu-depends-on-ref {
            description
                "Reference to the VDU that
                source VDU depends.";
            type leafref {
              path "../../vdu/id";
            }
          }
        }

        leaf service-function-chain {
          description "Type of node in Service Function Chaining
Architecture";

          type enumeration {
            enum UNAWARE;
            enum CLASSIFIER;
            enum SF;
            enum SFF;
          }
          default "UNAWARE";
        }

        leaf service-function-type {
          description
              "Type of Service Function.
               NOTE: This needs to map with Service Function Type in ODL to
               support VNFFG. Service Function Type is mandatory param in ODL
               SFC. This is temporarily set to string for ease of use";
              type string;
        }

        uses manotypes:monitoring-param;

        list placement-groups {
          description "List of placement groups at VNF level";

          key "name";
          uses manotypes:placement-group-info;

          list member-vdus {

            description
                "List of VDUs that are part of this placement group";
            key "member-vdu-ref";

            leaf member-vdu-ref {
              type leafref {
                path "../../../vdu/id";
```

```
                    }
                }
            }
        }
    }
}
```

Examples

See "API Examples" on page 346.

vnfd:vnf-configuration

Information about the VNF configuration for the management interface.

Note: If the network service contains multiple instances of the same VNF, each VNF instance could have a different configuration.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration

Fields

ID	Type	Cardinality	Description
netconf	container	1	Use NETCONF for configuring the VNF. See "vnfd:netconf" on page 168.
rest	container	1	Use REST for configuring the VNF. See "vnfd:rest" on page 169.
script	container	1	Use a custom script for configuring the VNF. This script will be executed in the Orchestrator. All required dependencies for the script should be available in the Orchestrator system. See "vnfd:script" on page 169.
juju	container	1	Use Juju for configuring the VNF. See "vnfd:juju" on page 170.
config-access	container	1	IP address to be used to configure this VNF. See "vnfd:config-access" on page 170.

ID	Type	Cardinality	Description
config-attributes	container	1	Miscellaneous config attributes to be considered while processing the NSD to apply configuration. See "vnfd:config-atrributes" on page 170.
config-primitive	list	0..n	List of service primitives supported by the configuration agent for this VNF. See "vnfd:config-primitive" on page 171.
initial-config-primitive	list	0..n	Initial set of configuration primitives. See "vnfd:initial-config-primitive" on page 173.
config-template	string	1	Configuration template for each VNF.

vnfd:netconf

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration/netconf

ID	Type	Cardinality	Description
target	enum	1	NETCONF configuration target. **Supported values:** • RUNNING • CANDIDATE

ID	Type	Cardinality	Description
protocol	enum	1	Protocol to use for NETCONF. **Supported values:** • NONE • SSH
port	inet:port-number	1	Port for the NETCONF server.

vnfd:rest

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration/rest

ID	Type	Cardinality	Description
port	inet_port-number	1	Port for the REST server

vnfd:script

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration/script

ID	Type	Cardinality	Description
script-type	enum	1	Script type to use. **Supported values:** • BASH • EXPECT

vnfd:juju

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration/juju

ID	Type	Cardinality	Description
charm	string	1	Juju charm to use to use with the VNF.

vnfd:config-access

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration/config-access

ID	Type	Cardinality	Description
mgmt-ip-address	union	1	IP address to be used to configure this VNF. **Note:** This parameter is optional if it is possible to dynamically resolve the IP.
username	string	1	User name for configuration.
password	string	1	Password for configuration access authentication.

vnfd:config-atrributes

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration/config-attributes

ID	Type	Cardinality	Description
config-priority	uint64	1	Order of configuration priority to be applied to each VNF in this network service. A low number takes precedence over a high number.

ID	Type	Cardinality	Description
config-delay	uint64	1	Wait time (in seconds) before applying the configuration to this VNF.

vnfd:config-primitive

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration/config-primitive/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the service primitive.
parameter	list	0..n	List of parameters to the service primitive. See "vnfd:config-primitive:parameter" on page 171.
user-defined-script	string	1	A user-defined script. If a script is defined, the script will be executed using bash.

vnfd:config-primitive:parameter

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration/config-primitive/STRING/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of parameter

ID	Type	Cardinality	Description
data-type	enum	1	Data type associated with the name. **Supported values:** • STRING • INTEGER • BOOLEAN
mandatory	boolean	1	[Default *false*] Defines whether this field is mandatory.
default-value	string	1	The default value for the field. **Note:** This value is required to set the read-only and hidden parameters.
parameter-pool	string	1	NSD parameter pool name to use for this parameter.
read-only	boolean	1	If set to *true*, the value is dimmed in the UI. **Note:** This parameter can be set to true only on parameters that specify a default-value field.
hidden	boolean	1	If set to true, the value is hidden from view in the UI. **Note:** This parameter can be set to true only on parameters that specify a default-value field.
out	boolean	1	Specifies if this is an output of the primitive execution.

vnfd:initial-config-primitive

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration/initial-config-primitive/0

ID	Type	Cardinality	Description
seq	uint64	1	Sequence number for the configuration primitive.
name	string	1	Name of the configuration primitive.
parameter	list	0..n	List of parameters to the configuration primitive. See "vnfd:initial-config-primitive:parameter" on page 173.
user-defined-script	string	1	A user-defined script
config-primitive-ref	leafref	1	Reference to a config primitive name. **Note:** The referenced config primitive should have all the input paramaters predefined either with default values or dependency references.

vnfd:initial-config-primitive:parameter

/vnfd:vnfd-catalog/vnfd/STRING/vnf-configuration/initial-config-primitive/0/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the parameter.
value	string	1	Value of the parameter.

vnfd:mgmt-interface

Interface over which the **VNF** is managed.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/mgmt-interface

Fields

ID	Type	Cardinality	Description
ip-address	union	1	Specifies the static IP address for managing the VNF.
vdu-id	leafref	1	Use the default management interface on this VDU. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd/vnfd:vdu/vnfd:id
cp	leafref	1	Use the IP address for the VNFD associated with this connection point endpoint. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd/vnfd:connection-point/vnfd:name
port	uint16	1	Port number for the management interface.
dashboard-params	container	1	Parameters for the VNF dashboard of the management interface. See "vnfd:dashboard-params" on page 175.

vnfd:dashboard-params

/vnfd:vnfd-catalog/vnfd/STRING/mgmt-interface/dashboard-params

ID	Type	Cardinality	Description
path	string	1	The HTTP path for the dashboard.
https	boolean	1	[Default *false*] Choose HTTPS instead of HTTP.
port	uint16	1	The HTTP port for the dashboard

vnfd:internal-vld

List of internal Virtual Link Descriptors (VLDs). Internal VLDs describe the basic topology of the connectivity—such as E-LAN— between internal VNF Components (**VNFC**) within the system.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/internal-vld/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Unique identifier for the internal VLD.
name	string	1	Name of the internal VLD.
short-name	string	1	Short name for the internal VLD to use as a label in the UI.
description	string	1	Description of the internal VLD.
type	enum	1	Type of virtual link. **Supported values:** ELAN: A multipoint service that connects a set of VDUs.
root-bandwidth	uint64	1	For ELAN this is the aggregate bandwidth.
leaf-bandwidth	uint64	1	For ELAN this is the bandwidth of branches.
internal-connection-point	list	0..n	List of internal connection points in this VLD. See "vnfd:internal-connection-point" on page 177.

ID	Type	Cardinality	Description
provider-network	container	1	Container for the provider network. See "vnfd:provider-network" on page 177.
vim-network-name	string	1	Name of pre-provisioned network in the VIM (cloud) account.
ip-profile-ref	string	1	Named reference to an ip-profile object.

vnfd:internal-connection-point

/vnfd:vnfd-catalog/vnfd/STRING/internal-vld/STRING/internal-connection-point/STRING

ID	Type	Cardinality	Description
id-ref	leafref	1	Reference to the internal connection point ID.

vnfd:provider-network

/vnfd:vnfd-catalog/vnfd/STRING/internal-vld/STRING/provider-network

ID	Type	Cardinality	Description
physical-network	string	1	Name of the physical network on which the provider network is built.

ID	Type	Cardinality	Description
overlay-type	enum	1	Identifies the type of the overlay network, which is a virtual network that is built on top of an existing network and is supported by its infrastructure. **Supported values:** • LOCAL — A network that can be realized on a single host only. • FLAT — The simplest networking environment in which each instance receives a fixed IP from the pool. All instances are attached to the same bridges. • VLAN — A network of computers in which the computers behaves as if they are connected to the same wire. However, the computers might be physically located on different segments of a LAN. • VXLAN — A proposed encapsulation protocol for running an overlay network on existing **Layer 3** infrastructure • GRE — GRE tunnels encapsulate isolated **Layer 2** network traffic in IP packets. Packets are routed between compute and networking nodes using the hosts' network connectivity and routing tables.
segmentation-id	uint32	1	Segmentation ID

vnfd:ip-profiles

List of IP profiles. IP profiles describe the IP characteristics for the virtual link.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/ip-profiles/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the IP profile.
description	string	1	Description of the IP profile.
ip-profile-params	container	1	Information about the IP profile. See "vnfd:ip-profile-params" on page 179

vnfd:ip-profile-params

/vnfd:vnfd-catalog/vnfd/STRING/ip-profiles/STRING/ip-profile-params

ID	Type	Cardinality	Description
ip-version	enum	1	[Default IPv4] Version of the Internet Protocol used.
subnet-address	union	1	Subnet IP prefix associated with this IP profile.
gateway-address	union	1	IP address of the default gateway associated with this IP profile.
security-group	string	1	Name of the security group.

ID	Type	Cardinality	Description
dns-server	list	0..n	List of DNS servers associated with this IP profile. See "vnfd:dns-server" on page 180.
dhcp-params	container	1	Container for DHCP parameters. See "vnfd:ip-profile-params" on page 179.
subnet-prefix-pool	string	1	VIM-specific reference to pre-created subnet prefix.

vnfd:dns-server

/vnfd:vnfd-catalog/vnfd/STRING/ip-profiles/STRING/ip-profile-params/dns-server/UNION_VALUE

ID	Type	Cardinality	Description
address	union	1	List of DNS servers associated with this IP profile.

vnfd:dhcp-params

/vnfd:vnfd-catalog/vnfd/STRING/ip-profiles/STRING/ip-profile-params/dhcp-params

ID	Type	Cardinality	Description
enabled	boolean	1	[Default *true*] Indicates if DHCP is enabled.
start-address	union	1	Start IP address of the IP address range associated with DHCP domain.

ID	Type	Cardinality	Description
count	uint32	1	Size of the DHCP pool associated with DHCP domain.

vnfd:connection-point

List of external connection points, in which each VNF:

- Has one or more points that are used to connect a VNF to other VNFs or to external networks
- Exposes these connection points to the orchestrator (**NFVO**)

The orchestrator constructs network services by connecting the connection points between different VNFs.

The orchestrator uses **VLDs** and **VNFFGs** at the network service level to construct network services.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/connection-point/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the connection point.
id	string	1	Unique identifier of the connection point
short-name	string	1	Short name of the connection point to use as a label in the UI.
type	enum	1	Type of connection point. **Supported values:** VPORT: Virtual Port
port-security-enabled	boolean	1	Enables the port security for the port
static-ip-address	union	1	Static IP address for the connection point

vnfd:vdu

A VDU is a basic part of VNF. VDUs are virtual machines that host the network function, such as:

- Virtual machine specification
- Computation properties (RAM size, disk size, memory page size, number of CPUs, number of cores per CPU, number of threads per core)
- Storage requirements
- Initiation and termination scripts
- High availability redundancy model
- Scale out/scale in limits

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Unique identifier for the VDU.
name	string	1	Unique name for the VDU.
description	string	1	Description of the VDU.
count	uint64	1	Number of instances of the VDU.
mgmt-vpci	string	1	Specifies the virtual PCI address, expressed in the following format dddd:dd:dd.d. For example 0000:00:12.0. This information can be used to pass as metadata during the VM creation.

ID	Type	Cardinality	Description
vm-flavor	container	1	Flavor of the virtual machine (VM) instance. See "vnfd:vm-flavor" on page 186.
guest-epa	container	1	EPA attributes for the guest operating system. See "vnfd:guest-epa" on page 187.
vswitch-epa	container	1	EPA attributes for Open vSwitch. See "vnfd:vswitch-epa" on page 193.
hypervisor-epa	container	1	EPA attributes for the hypervisor. See "vnfd:hypervisor-epa" on page 194.
host-epa	container	1	EPA attributes for the host operating system. See "vnfd:host-epa" on page 195.
alarm	list	0..n	A list of alarms. See "vnfd:alarm" on page 200.
image	string	1	Image name for the software image. If the image name is found within the VNF package it will be uploaded to all cloud accounts during the onboarding process. Otherwise, the image must be added to the cloud account with the same name as entered in this field.
image-checksum	string	1	Image md5sum for the software image. The md5sum, if provided, along with the image name, uniquely identifies an image uploaded to the CAL.
cloud-init	string	1	Contents of cloud-init script, provided inline, in cloud-config format

ID	Type	Cardinality	Description
cloud-init-file	string	1	Name of file with contents of cloud-init script in cloud-config format.
supplemental-boot-data	container	1	Container for custom VIM data. See "vnfd:supplemental-boot-data" on page 204.
internal-connection-point	list	0..n	List for internal connection points. See "vnfd:internal-connection-point" on page 206.
internal-interface	list	0..n	List of internal interfaces for the VNF. See "vnfd:internal-interface" on page 207.
external-interface	list	0..n	List of external interfaces for the VNF. See "vnfd:external-interface" on page 209.

vnfd:vm-flavor

Flavor is an alternative term for a VM instance type.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/vm-flavor

Fields

ID	Type	Cardinality	Description
vpcu-count	uint16	1	Number of VCPUs for the VM.
memory-mb	uint64	1	Amount of memory in MB to allocate to the VM.
storage-gb	uint64	1	Amount of disk space in GB to allocate to the VM.

vnfd:guest-epa

EPA attributes for the guest operating system.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/guest-epa

Fields

ID	Type	Cardinality	Description
trusted-execution	boolean	1	If set to *true*, indicates this VM should be allocated from trusted pool.
mempage-size	enum	1	Memory page allocation size. **Supported values:** • LARGE: Require hugepages (either 2MB or 1GB) • SMALL: Doesn't require hugepages • SIZE_2MB: Requires 2MB hugepages • SIZE_1GB: Requires 1GB hugepages • PREFER_LARGE: Application prefers hugepages **Note:** If a VM requires hugepages, choose LARGE or SIZE_2MB or SIZE_1GB. If the VM prefers hugepages, choose PREFER_LARGE.
cpu-pinning-policy	enum	1	Describes the association between virtual CPUs in the guest and the physical CPUs in the host. **Supported values:** • DEDICATED: Virtual CPUs are pinned to physical CPUs • SHARED: Multiple VMs may share the same physical CPUs. • ANY: (Default) Any policy is acceptable for the VM

ID	Type	Cardinality	Description
cpu-thread-pinning-policy	enum	1	Describes how to place the guest CPUs when the host supports hyper threads. **Default values:** • AVOID: Avoids placing a guest on a host with threads. • SEPARATE: Places vCPUs on separate cores, and avoids placing two vCPUs on two threads of same core. • ISOLATE: Places each vCPU on a different core, and places no vCPUs from a different guest on the same core. • PREFER: Attempts to place vCPUs on threads of the same core.
pcie-device	list	0..1	List of PCIE passthrough devices. See "vnfd:pcie-device" on page 189
numa-unaware	empty		Details about the numa-node-policy are null.
numa-node-policy	container	1	Defines numa topology of the guest, specifying if the guest should run on a host with one numa node or multiple numa nodes. **Example:** A guest might need 8 VCPUs and 4 GB of memory with the VCPUs and memory distributed across multiple NUMA nodes. In this scenario, NUMA node 1 could run with 6 VCPUs and 3GB, and NUMA node 2 could run with 2 vcpus and 1GB. See "vnfd:numa-node-policy" on page 190

vnfd:pcie-device

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/guest-epa/pcie-device/STRING

ID	Type	Cardinality	Description
device-id	string	1	Device identifier.
count	uint64	1	Number of devices to attach to the VM.

vnfd:numa-node-policy

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/guest-epa/numa-node-policy

ID	Type	Cardinality	Description
node-cnt	uint16	1	Number of NUMA nodes to expose to the VM.
mem-policy	enum	1	Specifies how to allocate memory in a multi-node scenario. **Supported values:** • STRICT: The memory must be allocated from the memory attached to the NUMA node. • PREFERRED: The memory should be allocated from the memory attached to the NUMA node
node	list	0..n	List of NUMA nodes. See "vnfd:numa-node-policy:node" on page 190.

vnfd:numa-node-policy:node

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/guest-epa/numa-node-policy/node/0

ID	Type	Cardinality	Description
id	uint64	1	NUMA node identification. Typically 0 or 1.
vpcu	list	0..n	List of VPCUs to allocate on this NUMA node. See "vnfd:numa-node-policy:node:vcpu" on page 191
memory-mb	uint64	1	Memory size in MB for this NUMA node.

ID	Type	Cardinality	Description
num-cores	uint8	1	Number of cores.
paired-threads	container	1	Container for paired threads. See "vnfd:numa-node-policy:node:paired-threads" on page 191.
num-threads	uint8	1	OpenMANO NUMA type selection.

vnfd:numa-node-policy:node:vcpu

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/guest-epa/numa-node-policy/node/0/vcpu/0

ID	Type	Cardinality	Description
id	uint64	1	List of VCPUs IDs to allocate on this NUMA node.

vnfd:numa-node-policy:node:paired-threads

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/guest-epa/numa-node-policy/node/0/paired-threads

ID	Type	Cardinality	Description
num-paired-threads	uint8	1	Number of paired-threads.

ID	Type	Cardinality	Description
paired-thread-ids	list	0..n	List of thread paired to use in case of paired thread NUMA. See "vnfd:numa-node-policy:node:paired-threads:paired-thread-ids" on page 192

vnfd:numa-node-policy:node:paired-threads:paired-thread-ids

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/guest-epa/numa-node-policy/node/0/paired-threads/paired-thread-ids/0

ID	Type	Cardinality	Description
thread-a	uint8	1	Thread ID
thread-b	uint8	1	Thread ID

vnfd:vswitch-epa

EPA attributes for Open vSwitch.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/vswitch-epa

Fields

ID	Type	Cardinality	Description
ovs-acceleration	enum	1	Specifies Open vSwitch acceleration mode. **Supported values:** • MANDATORY: OVS acceleration is required • PREFERRED: OVS acceleration is preferred • DISABLED: OVS acceleration is disabled.
ovs-offload	enum	1	Specifies Open vSwitch hardware offload mode. **Supported values:** • MANDATORY: OVS offload is required • PREFERRED: OVS offload is preferred • DISABLED: OVS offload is disabled

vnfd:hypervisor-epa

EPA attributes for the hypervisor.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/hypervisor-epa

Fields

ID	Type	Cardinality	Description
type	enum	1	Specifies the type of hypervisor. For example, KVM, XEN. Value can be: • KVM: KVM • XEN: XEN
version	string	1	Version of the hypervisor.

vnfd:host-epa

Specifies the host-level EPA attributes.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/host-epa

Fields

ID	Type	Cardinality	Description
cpu-model	enum	1	Host CPU model. **Supported values:** • PREFER_WESTMERE • REQUIRE_WESTMERE • PREFER_SANDYBRIDGE • REQUIRE_SANDYBRIDGE • PREFER_IVYBRIDGE • REQUIRE_IVYBRIDGE • PREFER_HASWELL • REQUIRE_HASWELL • PREFER_BROADWELL • REQUIRE_BROADWELL • PREFER_NEHALEM • REQUIRE_NEHALEM • PREFER_PENRYN • REQUIRE_PENRYN • PREFER_CONROE • REQUIRE_CONROE • PREFER_CORE2DUO • REQUIRE_CORE2DUO

ID	Type	Cardinality	Description
cpu-arch	enum	1	Host CPU architecture. **Supported values:** • PREFER_X86 • REQUIRE_X86 • PREFER_X86_64 • REQUIRE_X86_64 • PREFER_I686 • REQUIRE_I686 • PREFER_IA64 • REQUIRE_IA64 • PREFER_ARMV7 • REQUIRE_ARMV7 • PREFER_ARMV8 • REQUIRE_ARMV8
cpu-vendor	enum	1	Host CPU vendor. **Supported values:** • PREFER_INTEL • REQUIRE_INTEL • PREFER_AMD • REQUIRE_AMD
cpu-socket-count	uint64	1	Number of sockets on the host.
cpu-core-count	uint64	1	Number of cores on the host.
cpu-core-thread-count	uint64	1	Number of threads per cores on the host.

ID	Type	Cardinality	Description
cpu-feature	list	0..1	List of CPU features. See "vnfd:cpu-feature" on page 197.
om-cpu-model-string	string	1	OpenMano CPU model string.
om-cpu-feature	list	0..n	OpenMano CPU features. See "vnfd:om-cpu-feature" on page 199.

vnfd:cpu-feature

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/host-epa/cpu-feature/PREFER_AES

ID	Type	Cardinality	Description

ID	Type	Cardinality	Description
feature	enum	1	**Enumeration for CPU features:** • AES: CPU supports advanced instruction set for AES (Advanced Encryption Standard). • CAT: Cache Allocation Technology (CAT) allows an operating system, hypervisor, or similar system management agent to specify the amount of L3 cache (currently the last-level cache in most server and client platforms) space an application can fill. **Note:** As a hint to hardware functionality, certain features, such as power management, may override CAT settings. • CMT: Cache Monitoring Technology (CMT) allows an Operating System, Hypervisor, or similar system management agent to determine the usage of cache based on applications running on the platform. The implementation is directed at L3 cache monitoring (currently the last-level cache in most server and client platforms). • DDIO: Intel Data Direct I/O (DDIO) enables Ethernet server NICs and controllers talk directly to the processor cache without a detour via system memory. This enumeration specifies if the VM requires a DDIO capable host. **Supported values:** • PREFER_AES • REQUIRE_AES • PREFER_CAT • REQUIRE_CAT • PREFER_CMT • REQUIRE_CMT • PREFER_DDIO • REQUIRE_DDIO • REQUIRE_VME • PREFER_VME • REQUIRE_DE • PREFER_DE • REQUIRE_PSE • PREFER_PSE • REQUIRE_TSC • PREFER_TSC

vnfd:om-cpu-feature

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/host-epa/om-cpu-feature/STRING

ID	Type	Cardinality	Description
feature	string	1	CPU feature.

vnfd:alarm

Information about alarms.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/alarm/STRING

Fields

ID	Type	Cardinality	Description
alarm-id	string	1	Reserved field for the identifier assigned by the **VIM** provider.
name	string	1	A human-readable string to identify the alarm.
description	string	1	Description of the alarm.
vdur-id	string	1	Identifier of the VDU record (VDUR) associated with this alarm.
actions	container	1	Actions related to the alarm. See "vnfd:actions" on page 202.
repeat	boolean	1	[Default *true*] Indicates whether the alarm should emit repeatedly after the associated threshold has been crossed.
enabled	boolean	1	[Default *true*] Indicates whether the alarm has been enabled or disabled.

ID	Type	Cardinality	Description
severity	enum	1	A measure of the important or urgency of the alarm. **Supported types:** • LOW • MODERATE • CRITICAL
metric	enum	1	Metric types that can be tracked by this alarm. **Supported values:** • CPU_UTILIZATION • MEMORY_UTILIZATION • STORAGE_UTILIZATION
statistic	enum	1	Type of statistic to use to measure a metric, which determines threshold crossing for an alarm. **Supported values:** • AVERAGE • MINIMUM • MAXIMUM • COUNT • SUM

ID	Type	Cardinality	Description
operation	enum	1	The relational operator used to define whether an alarm should be triggered when the metric statistic goes above or below a specified threshold value. **Supported values:** • GE — Greater than or equal to • LE — Less than or equal to • GT — Greater than • LT — Less than • EQ — Equal
value	decimal164	1	Defines the threshold (up to 4 fraction digits) that, if crossed, will trigger the alarm.
period	uint32	1	Defines the length of time (seconds) for which metric data are collected to evaluate the chosen statistic.
evaluation	uint32	1	Number of samples of the metric statistic used to evaluate threshold crossing. Each sample or evaluation is equal to the metric statistic obtained for a given period. **Note:** This value can be used to mitigate spikes in the metric that may skew the statistic of interest.

vnfd:actions

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/alarm/STRING/actions

ID	Type	Cardinality	Description
ok	list	0..n	See "vnfd:actions" on page 202.

ID	Type	Cardinality	Description
insufficient-data	list	0..n	See "vnfd:actions:insufficient-data" on page 203.
alarm	list	0..n	See "vnfd:actions:alarm" on page 203.

vnfd:actions:ok

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/alarm/STRING/actions/ok/STRING

ID	Type	Cardinality	Description
url	string	1	

vnfd:actions:insufficient-data

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/alarm/STRING/actions/insufficient-data/STRING

ID	Type	Cardinality	Description
url	string	1	

vnfd:actions:alarm

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/alarm/STRING/actions/alarm/STRING

ID	Type	Cardinality	Description
url	string	1	

vnfd:supplemental-boot-data

Container for custom VIM data.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/supplemental-boot-data

Fields

ID	Type	Cardinality	Description
config-file	list	0..n	List of configuration files to be mounted onto an additional drive. See "vnfd:config-file" on page 204.
custom-meta-data	list	0..n	List of metadata to be associated with the instance. See "vnfd:custom-meta-data" on page 205.
boot-data-drive	boolean	1	[Default *false*] Specifies whether the VIM should implement additional drives to host config-files or metadata.

vnfd:config-file

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/supplemental-boot-data/config-file/STRING

ID	Type	Cardinality	Description
source	string	1	Name of the configuration file.
dest	string	1	Full path of the destination in the guest.

vnfd:custom-meta-data

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/supplemental-boot-data/custom-meta-data/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the metadata parameter.
data-type	enum	1	Data type of the metadata parameter.
value	string	1	Value of the metadata parameter.

vnfd:internal-connection-point

List for internal connection points. Each VNFC has zero or more internal connection points. Internal connection points are used for connecting the VNF components internal to the VNF. If a VNF has only one VNFC, it may not have any internal connection points.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/internal-connection-point/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the connection point.
id	string	1	Identifier for the internal connection points.
short-name	string	1	Short name to use as a label in the UI.
type	enum	1	Type of connection point. **Supported values:** VPORT: Virtual Port
port-security-enabled	boolean		Specifies whether to enable port security for the port.
static-ip-address	union	1	Static IP address for the connection point
internal-vld-ref	leafref	1	This is a leafref to path: ../../../internal-vld/id

vnfd:internal-interface

Internal interfaces enable traffic between virtual network functions.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/internal-interface/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the internal interface inside the VDU. **Note:** This name has only local significance to the VDU.
vdu-internal-connection-point-ref	leafref	1	Reference to an internal connection point. This is a leafref to path: ../../internal-connection-point/id
virtual-interface	container	1	Container for the virtual interface properties. See "vnfd:virtual-interface" on page 208.

vnfd:virtual-interface

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/internal-interface/STRING/virtual-interface

ID	Type	Cardinality	Description
type	enum	1	Specifies the type of virtual interface between VM and host. **Supported values:** • VIRTIO: [Default] Use the traditional VIRTIO interface • PCI-PASSTHROUGH: Use PCI-PASSTHROUGH interface • SR-IOV: Use SR-IOV interface • E1000 : Emulate E1000 interface • RTL8139 : Emulate RTL8139 interface • PCNET : Emulate PCNET interface • OM-MGMT: Used to specify OpenMANO management internal-connection type
vpci	string	1	Specifies the virtual PCI address in format dddd:dd:dd.d. For example: 0000:00:12.0 This information can be used to pass as metadata during the VM creation.
bandwidth	uint64	1	Specifies the aggregate bandwidth requirement for the NIC.

vnfd:external-interface

External interfaces enable traffic between VNFs.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/external-interface/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the external interface inside the VDU. **Note:** This name has only local significance to the VDU.
vnfd-connection-point-ref	reference	1	Reference to an external connection point. This is a leafref to path: ../../../connection-point/name
virtual-interface	container	1	Virtual interface properties. See "vnfd:virtual-interface" on page 210.

vnfd:virtual-interface

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/external-interface/STRING/virtual-interface

ID	Type	Cardinality	Description
type	enum	1	Specifies the type of virtual interface between VM and host. **Supported values:** • VIRTIO: [Default] Use the traditional VIRTIO interface • PCI-PASSTHROUGH: Use PCI-PASSTHROUGH interface • SR-IOV: Use SR-IOV interface • E1000 : Emulate E1000 interface • RTL8139 : Emulate RTL8139 interface • PCNET : Emulate PCNET interface • OM-MGMT: Used to specify OpenMANO management internal-connection type
vpci	string	1	Specifies the virtual PCI address in format dddd:dd:dd.d. For example: 0000:00:12.0 **Note:** This information can be used to pass as metadata during the VM creation.
bandwidth	uint64	1	Specifies the aggregate bandwidth requirement for the NIC.

vnfd:volumes

Defines disk volumes to be attached to the **VDU**, such as if a VNF requires multiple disks to boot the virtual machine.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/volumes/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the disk-volumes, such as vda, vdb.
description	string	1	Description for the volume.
size	uint64	1	Size of the disk, in GB.
ephemeral	empty		
image	string	1	Image name for the software image to be used. If the image name is found within the VNF package it will be uploaded to all cloud accounts during the onboarding process. Otherwise, the image must be added to the cloud account with the same name as entered in this field.
image-checksum	string	1	Image md5sum for the software image. The md5sum, if provided, along with the image name, uniquely identifies an image uploaded to the CAL.
volume-ref	string	1	Reference to the pre-existing volume in **VIM.**
boot-volume	boolean	1	If set to *true*, indicates that this is a boot volume

ID	Type	Cardinality	Description
boot-priority	int32		Boot priority associated with volume.
device_bus	enum	1	Type of disk-bus on which this disk is exposed to the guest operating system. **Supported values:** • IDE • USB • VIRTIO • SCSI
device_type	enum	1	Type of device as exposed to the guest operating system. **Supported values:** • DISK • CDROM • FLOPPY • LUN (logical unit number)
custom-meta-data	list	0..n	List of metadata to be associated with the instance. See "vnfd:custom-meta-data" on page 212

vnfd:custom-meta-data

/vnfd:vnfd-catalog/vnfd/STRING/vdu/STRING/volumes/STRING/custom-meta-data/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the metadata parameter.

ID	Type	Cardinality	Description
data-type	enum	1	Data type of the metadata parameter.
value	string	1	Value of the metadata parameter.

vnfd:vdu-dependency

List of virtual deployment unit (**VDU**) dependencies, from which the orchestrator determines the order of startup among the VDUs.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/vdu-dependency/STRING

Fields

ID	Type	Cardinality	Description
vdu-source-ref	leafref	1	Identifier of the VDU. This is a leafref to path: ../../vdu/id
vdu-depends-on-ref	leafref	1	Reference to the VDU on which the source VDU depends. This is a leafref to path: ../../vdu/id

vnfd:http-endpoint

List of http endpoints to be used by monitoring parameters.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/http-endpoint/STRING

Fields

ID	Type	Cardinality	Description
path	string	1	The HTTP path on the management server.
https	boolean	1	[Default *false*] Pick HTTPS instead of HTTP.
port	uint16	1	HTTP port to connect to.
username	string	1	HTTP basic auth user name.
password	string	1	HTTP basic auth password.
polling_interval_secs	uint8	1	[Default 2] HTTP polling interval in seconds.
method	enum	1	Method to be performed at the URI. **Supported values:** • GET (default) • POST • PUT • GET • DELETE • PATCH • OPTIONS

ID	Type	Cardinality	Description
data	string	1	The data to be sent with POST.
headers	list	0..n	List of custom HTTP headers to put on the HTTP request. See "vnfd:headers" on page 216.

vnfd:headers

/vnfd:vnfd-catalog/vnfd/STRING/http-endpoint/STRING/headers/STRING

ID	Type	Cardinality	Description
key	string	1	HTTP header key.
value	string	1	HTTP header value.

vnfd:placement-groups

List of placement groups at VNF level. The placement group construct defines the compute resource placement strategy in a cloud environment.

REST URI path

/vnfd:vnfd-catalog/vnfd/STRING/placement-groups/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Placement group name.
requirement	string	1	Describes the intent/rationale behind this placement group. **Note:** This free-text field is for human consumption only.
strategy	enum	1	Strategy associated with this placement group. **Supported values:** • COLOCATION: [Default] Share the physical infrastructure (hypervisor/network) among all members of this group. • ISOLATION: Do not share the physical infrastructure (hypervisor/network) among the members of this group
member-vdus	list	0..n	List of VDUs that are part of this placement group. See "vnfd:member-vdus" on page 218.

vnfd:member-vdus

/vnfd:vnfd-catalog/vnfd/STRING/placement-groups/STRING/member-vdus/STRING

ID	Type	Cardinality	Description
member-vdu-ref	leafref	1	Reference to the VDU in the VNF. This is a leafref to path: ../../../vdu/id

VNF Lifecycle Management (vnfr:vnfr)

REST wrapper for the VNF lifecycle management service. Provides methods for querying a VNF and retrieving VNF records from the vnfr:vnfr-catalog.

Methods and relative URLs

Method	Relative URL	Description
GET	/api/running/vnfr-catalog/vnfr{UUID}	Query a VNF.
GET	/api/running/vnfr-catalog/vnfr/	Retrieve a list of all VNF records from the catalog.

Fields

ID	Type	Cardinality	Description
id	string	1	Identifier for the VNFR.
nsr-id-ref	string	1	NS instance identifier. This is a leafref to path: /nsr:ns-instance-config/nsr:nsr/nsr:id
member-vnf-index-ref	leafref	1	Path to member VNF index in the network service. /nsd:nsd-catalog/nsd:nsd/nsd:constituent-vnfd/nsd:member-vnf-index
dashboard-url	string	1	Dashboard URL.
name	string	1	VNFR name.

ID	Type	Cardinality	Description
short-name	string	1	Short name for the VNFR (for display on the UI).
vendor	string	1	Vendor/provider of the VNFR.
description	string	1	Description of the VNFR.
version	string	1	Version of the VNFR.
create-time	uint32	1	Creation timestamp of this VNF. The timestamp is expressed as seconds since unix epoch - 1970-01-01T00:00:00Z
uptime	uint32	1	Active period (in seconds) of this VNF.
vnfd	container	1	VNF descriptor used to instantiate this VNF. See "vnfr:vnfd" on page 233.
vnfd-ref	leafref	1	Reference to a VNF descriptor. This is a leafref path to: /vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id
vnf-configuration	container	1	Container for the VNF configuration. **Note:** If the NS contains multiple instances of the same VNF, each instance of the VNF may have different configuration. See "vnfr:vnf-configuration" on page 290.
mgmt-interface	container	1	Container for the management interface. See "vnfr:mgmt-interface" on page 297.

ID	Type	Cardinality	Description
internal-vlr	list	0..n	List of references to virtual link records (VLR) in the VLR catalog. See "vnfr:internal-vlr" on page 298.
connection-point	list	0..n	List of external connection points for VNFs. See "vnfr:connection-point" on page 299.
vdur	list	0..n	List of virtual deployment units (VDU). See "vnfr:vdur" on page 301.
http-endpoint	list	0..n	List of http endpoints to be used by monitoring-param. See "vnfr:http-endpoint" on page 335.
monitoring-param	list	0..n	List of monitoring parameters at the network service level. See "vnfr:monitoring-param" on page 337.
operational-status	enum	1	The operational status of the VNFR instance init: • init : The VDU has just started • vm-init-phase : The VDUs in the VNF is being created in VIM • vm-alloc-pending : The VM alloc is pending in VIM • running : The VDU is active in VM • terminate : The VDU is being terminated • vm-terminate-phase: The VDU in the VNF is being terminated in VIM • terminated : The VDU is in the terminated state • failed : The VDU instantiation failed This element uses manotypes:monitoring-param

ID	Type	Cardinality	Description
config-status	enum	1	The configuration status of the NS instance: • configuring: At least one of the VNFs in this instance is in configuring state • configured: All the VNFs in this NS instance are configured or config-not-needed state
placement-groups-info	list	0..n	Placement groups to which this VDU belongs and its cloud construct. See "vnfr:placement-groups-info" on page 341.
cloud-config	container	1	Container for cloud configuration information about public keys and users. See "vnfr:cloud-config" on page 344.

MANO reference points

Or-Vnfm and Os-Ma-nfvo

Schema

```
module vnfr
 {
   namespace "urn:ietf:params:xml:ns:yang:nfvo:vnfr";
   prefix "vnfr";

   import mano-types {
     prefix "manotypes";
   }

   import rw-pb-ext {
     prefix "rwpb";
   }

   import vnfd {
     prefix "vnfd";
   }

   import nsd {
     prefix "nsd";
   }
```

```
  import vlr {
    prefix "vlr";
  }

  import ietf-yang-types {
    prefix "yang";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  grouping placement-group-info {
    list placement-groups-info {
      description
          "
          Placement groups to which this VDU belongs and its
          cloud construct
          ";
      key "name";
      uses manotypes:placement-group-info;
      uses manotypes:placement-group-input;
    }
  }


  grouping virtual-interface {
    container virtual-interface {
      description
          "Container for the virtual interface properties";

      leaf type {
        description
            "Specifies the type of virtual interface
             between VM and host.
             VIRTIO          : Use the traditional VIRTIO interface.
             PCI-PASSTHROUGH : Use PCI-PASSTHROUGH interface.
             SR-IOV          : Use SR-IOV interface.";
        type enumeration {
          enum VIRTIO;
          enum PCI-PASSTHROUGH;
          enum SR-IOV;
        }
      }

      leaf bandwidth {
        description
            "Aggregate bandwidth of the NIC.";
        type uint64;
      }

      leaf ovs-offload {
        description
            "Defines if the NIC supports OVS offload.
             MANDATORY : OVS offload support in the NIC is mandatory.
             PREFERRED : OVS offload support in the NIC is preferred.";
```

```
        type enumeration {
          enum MANDATORY;
          enum PREFERRED;
        }
      }

      leaf vendor-id {
        description
            "Specifies the vendor specific id for
             the device. This is used when a NIC from
             specific HW vendor is required.";
        type string;
      }

      leaf datapath-library {
        description
            "Specifies the name and version of the datapath
             library the NIC is expected to support.";
        type string;
      }

      leaf provider-network-name {
        description
            "Name of the provider network to which this
             NIC is attached.";
        type string;
      }
    }
  }

  container vnfr-catalog {
    config false;
    list vnfr {
      description
          "Virtual Network Function Record (VNFR).";
      key "id";
      unique "name";

      leaf id {
        description "Identifier for the VNFR.";
        type yang:uuid;
      }

      leaf nsr-id-ref {
        description
            "NS instance identifier.
             This is a leafref /nsr:ns-instance-config/nsr:nsr/nsr:id";
        type yang:uuid;
      }

      leaf member-vnf-index-ref {
        description "Reference to member VNF index in Network service.";
        type leafref {
          path "/nsd:nsd-catalog/nsd:nsd/nsd:constituent-vnfd/nsd:member-
vnf-index";
        }
```

```
      }

      leaf dashboard-url {
        description "Dashboard URL";
        type inet:uri;
      }

      leaf name {
        description "VNFR name.";
        type string;
      }

      leaf short-name {
        description "VNFR short name.";
        type string;
      }

      leaf vendor {
        description "Vendor of the VNFR.";
        type string;
      }

      leaf description {
        description "Description of the VNFR.";
        type string;
      }

      leaf version {
        description "Version of the VNFR";
        type string;
      }

      leaf create-time {
        description
          "Creation timestamp of this Virtual Network
          Function.  The timestamp is expressed as
          seconds since unix epoch - 1970-01-01T00:00:00Z";

        type uint32;
      }

      leaf uptime {
        description
          "Active period of this Virtual Network Function.
          Uptime is expressed in seconds";

        type uint32;
      }

      container vnfd {
        description "VNF descriptor used to instantiate this VNF";
        uses vnfd:vnfd-descriptor;
      }
      leaf vnfd-ref {
        description "Reference to VNFD";
        type leafref {
```

```
        path "/vnfd:vnfd-catalog/vnfd:vnfd/vnfd:id";
      }
    }

    // Use parameters provided here to configure this VNF
    uses manotypes:vnf-configuration;

    // Mainly used by Mon-params & dashboard url
    container mgmt-interface {
      leaf ip-address {
        type inet:ip-address;
      }
      leaf port {
        type inet:port-number;
      }

      container ssh-key {
        description "SSH key pair used for this VNF";
        leaf public-key {
          description "Public key configured on this VNF";
          type string;
        }
        leaf private-key-file {
          description "Path to the private key file";
          type string;
        }
      }
    }

    list internal-vlr {
      key "vlr-ref";

      leaf vlr-ref {
        description "Reference to a VLR record in the VLR catalog";
        type leafref {
          path "/vlr:vlr-catalog/vlr:vlr/vlr:id";
        }
      }

      leaf-list internal-connection-point-ref {
        type leafref {
          path "../../vdur/internal-connection-point/id";
        }
      }
    }

    list connection-point {
      key "name";
      description
          "List for external connection points. Each VNF has one
           or more external connection points. As the name
           implies that external connection points are used for
           connecting the VNF to other VNFs or to external networks.
           Each VNF exposes these connection points to the
           orchestrator. The orchestrator can construct network
           services by connecting the connection points between
```

```
            different VNFs. The NFVO will use VLDs and VNFFGs at
            the network service level to construct network services.";

        uses vnfd:common-connection-point;

        leaf vlr-ref {
          description
              "Reference to the VLR associated with this connection point";
          type  leafref {
            path "/vlr:vlr-catalog/vlr:vlr/vlr:id";
          }
        }

        leaf ip-address {
          description
              "IP address assigned to the external connection point";
          type inet:ip-address;
        }
        leaf mac-address {
          description
              "MAC address assigned to the external connection point";
          // type inet:mac-address;
          type string;
        }
        leaf connection-point-id {
          rwpb:field-inline "true";
          rwpb:field-string-max 64;
          type string;
        }
      }

      list vdur {
        description "List of Virtual Deployment Units";
        key "id";
        unique "name";

        leaf id {
          description "Unique id for the VDU";
          type yang:uuid;
        }

        leaf name {
          description "name of the instantiated VDUR";
          type string;
        }

        leaf vdu-id-ref {
          type leafref {
            path "../../vnfd/vdu/id";
          }
        }

        leaf vim-id {
          description "Allocated VM resource id";
          type string;
        }
```

```
        leaf flavor-id {
          description "VIM assigned flavor id";
          type string;
        }

        leaf image-id {
          description "VIM assigned image id";
          type string;
        }

        leaf management-ip {
          description "Management IP address";
          type inet:ip-address;
        }

        leaf vm-management-ip {
          description "VM Private Management IP address";
          type inet:ip-address;
        }

        leaf console-url {
          description "Console URL for this VDU, if available";
          type inet:uri;
        }

        uses manotypes:vm-flavor;
        uses manotypes:guest-epa;
        uses manotypes:vswitch-epa;
        uses manotypes:hypervisor-epa;
        uses manotypes:host-epa;

        uses manotypes:supplemental-boot-data;

        list volumes {
          key "name";

          leaf name {
            description "Name of the disk-volumes, e.g. vda, vdb etc";
            type string;
          }

          leaf volume-id {
            description "VIM assigned volume id";
            type string;
          }

          uses manotypes:volume-info;
        }

        list alarms {
          description
              "A list of the alarms that have been created for this VDU";

          key "alarm-id";
          uses manotypes:alarm;
```

```
        }

        list internal-connection-point {
          key "id";
          description
              "List for internal connection points. Each VNFC
               has zero or more internal connection points.
               Internal connection points are used for connecting
               the VNF components internal to the VNF. If a VNF
               has only one VNFC, it may not have any internal
               connection points.";

          uses vnfd:common-connection-point;

          leaf ip-address {
            description
                "IP address assigned to the internal connection point";
            type inet:ip-address;
          }
          leaf mac-address {
            description
                "MAC address assigned to the internal connection point";
            // type inet:mac-address;
            type string;
          }
        }

        list internal-interface {
          description
              "List of internal interfaces for the VNF";
          key name;

          leaf name {
            description
                "Name of internal interface. Note that this
                 name has only local significance to the VDU.";
            type string;
          }

          leaf vdur-internal-connection-point-ref {
            type leafref {
              path "../../internal-connection-point/id";
            }
          }
          uses virtual-interface;
        }

        list external-interface {
          description
              "List of external interfaces for the VNF.
               The external interfaces enable sending
               traffic to and from VNF.";
          key name;

          leaf name {
            description
```

```
                "Name of the external interface. Note that
                 this name has only local significance.";
            type string;
          }

          leaf vnfd-connection-point-ref {
            description
              "Name of the external connection point.";
            type leafref {
              path "../../../connection-point/name";
            }
          }
          uses virtual-interface;
        }
        leaf operational-status {
          description
            "The operational status of the VDU
              init                : The VDU has just started.
              vm-init-phase       : The VDUs in the VNF is being created in
VIM.
              vm-alloc-pending    : The  VM alloc is pending in VIM
              running             : The VDU is active in VM
              terminate           : The VDU is being terminated
              vm-terminate-phase  : The VDU in the VNF is being terminated
in VIM.
              terminated          : The VDU is in the terminated state.
              failed              : The VDU  instantiation failed.
            ";

          type enumeration {
            rwpb:enum-type "VduOperationalStatus";
            enum init;
            enum vm-init-phase;
            enum vm-alloc-pending;
            enum running;
            enum terminate;
            enum vl-terminate-phase;
            enum terminated;
            enum failed;
          }
        }
        uses placement-group-info;
      }

      uses manotypes:monitoring-param;

      leaf operational-status {
        description
          "The operational status of the VNFR instance
            init                : The VNF has just started.
            vl-init-phase       : The internal VLs in the VNF are being
instantiated.
            vm-init-phase       : The VMs for VDUs in the VNF are being
instantiated.
            running             : The VNF is in running state.
            terminate           : The VNF is being terminated.
```

```
              vm-terminate-phase  : The VMs in the VNF are being terminated.
              vl-terminate-phase  : The internal VLs in the VNF are being
terminated.
              terminated          : The VNF is in the terminated state.
              failed              : The VNF instantiation failed
            ";

          type enumeration {
            rwpb:enum-type "VnfrOperationalStatus";
            enum init;
            enum vl-init-phase;
            enum vm-init-phase;
            enum running;
            enum terminate;
            enum vm-terminate-phase;
            enum vl-terminate-phase;
            enum terminated;
            enum failed;
          }
        }
        leaf config-status {
          description
            "The configuration status of the NS instance
              configuring: At least one of the VNFs in this instance is in
configuring state
              configured:  All the VNFs in this NS instance are configured or
config-not-needed state
            ";

          type enumeration {
            enum configuring {
              value 1;
            }
            enum configured {
              value 2;
            }
            enum failed {
              value 3;
            }
            enum config-not-needed {
              value 4;
            }
          }
        }
        uses placement-group-info;
        container cloud-config {
          rwpb:msg-new VnfrCloudConfig;
          uses manotypes:cloud-config;
        }
      }
    }

    rpc create-alarm {
      description "Create an alert for a running VDU";
      input {
        leaf cloud-account {
```

```
        mandatory true;
        type string;
      }

      leaf vdur-id {
        mandatory true;
        type string;
      }

      container alarm {
        uses manotypes:alarm;
      }
    }

    output {
      leaf alarm-id {
        type string;
      }
    }
  }

  rpc destroy-alarm {
    description "Destroy an alert that is associated with a running VDU";
    input {
      leaf cloud-account {
        mandatory true;
        type string;
      }

      leaf alarm-id {
        mandatory true;
        type string;
      }
    }
  }
}
```

Examples

See "API Examples" on page 346.

vnfr:vnfd

VNF descriptor (**VNFD**) used to instantiate this VNF.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd

Fields

ID	Type	Cardinality	Description
id	string	1	Identifier for the VNFD.
name	string	1	VNFD name.
short-name	string	1	VNFD short name to use as a label in the UI.
vendor	string	1	Provider of the VNFD.
logo	string	1	File path of the vendor-specific logo. For example, icons/mylogo.png The logo should be part of the VNF package. SVG format is preferred, but PNG is supported. Although there is no hard limit on size and dimension, a square image under 200px by 200px is preferred.
description	string	1	Description of the VNFD.
version	string	1	Version of the VNFD.
vnf-configuration	container		Information about the **VNF** configuration for the management interface. See "vnfr:vnf-configuration" on page 236.

ID	Type	Cardinality	Description
config-parameter	container	1	List of VNF configuration parameter requests and sources. See vnfr:config-parameter
mgmt-interface	container	1	Interface over which the VNF is managed. See "vnfr:mgmt-interface" on page 243.
internal-vld	list	0..n	List of Internal Virtual Link Descriptors (VLD). See "vnfr:internal-vld" on page 245.
ip-profiles	list	0..n	List of IP profiles. An IP profile describes the IP characteristics for the virtual-link. See "vnfr:ip-profiles" on page 248.
connection-point	list	0..n	The list for external connection points. See "vnfr:connection-point" on page 250.
vdu	list	0..n	List of virtual deployment units, which are VMs that host the network function. See "vnfr:vdu" on page 251.
vdu-dependency	list	0..n	List of VDU dependencies. See "vnfr:vdu-dependency" on page 281.

ID	Type	Cardinality	Description
service-function-chain	enum	1	Type of node in service function chaining architecture. **Supported types:** • UNAWARE (default) • CLASSIFIER • SF • SFF
service-function-type	string	1	Type of service function. **Note:** This value needs to map to a service function type in the OpenDaylight platform to support VNFFG
http-endpoint	list	0..n	List of http endpoints to be used by monitoring-param. See "vnfr:http-endpoint" on page 282.
monitoring-param	list	0..n	List of monitoring parameters at the network service level. See vnfd:monitoring-param.
placement-groups	list	0..n	List of placement groups at VNF level. See "vnfr:placement-groups" on page 288.

vnfr:vnf-configuration

Information about the VNF configuration for the management interface.

Note: If the network service contains multiple instances of the same VNF, each VNF instance could have a different configuration.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration

Fields

ID	Type	Cardinality	Description
netconf	container	1	Use NETCONF for configuring the VNF. See "vnfr:vnf-configuration" on page .
rest	container	1	Use REST for configuring the VNF. See "vnfr:rest" on page 238.
script	container	1	Use a custom script for configuring the VNF. This script will be executed in the Orchestrator. All required dependencies for the script should be available in the Orchestrator system. See "vnfr:script" on page 238.
juju	container	1	Use Juju for configuring the VNF. See "vnfr:juju" on page 238.
config-access	container	1	IP address to be used to configure this VNF. See "vnfr:config-access" on page 239.

ID	Type	Cardinality	Description
config-attributes	container	1	Miscellaneous config attributes to be considered while processing the NSD to apply configuration. See "vnfr:config-atrributes" on page 239.
config-primitive	list	0..n	List of service primitives supported by the configuration agent for this VNF. See "vnfr:config-primitive" on page 240.
config-template	string	1	Configuration template for each VNF.

vnfr:netconf

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration/netconf

ID	Type	Cardinality	Description
target	enum	1	NETCONF configuration target. **Supported values:** • RUNNING • CANDIDATE
protocol	enum	1	Protocol to use for NETCONF. **Supported values:** • NONE • SSH
port	inet:port-number	1	Port for the NETCONF server.

vnfr:rest

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration/rest

ID	Type	Cardinality	Description
port	inet_port-number	1	Port for the REST server

vnfr:script

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration/script

ID	Type	Cardinality	Description
script-type	enum	1	Script type to use. **Supported values:** • BASH • EXPECT

vnfr:juju

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration/juju

ID	Type	Cardinality	Description
charm	string	1	Juju charm to use to use with the VNF.

vnfr:config-access

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration/config-access

ID	Type	Cardinality	Description
mgmt-ip-address	union	1	IP address to be used to configure this VNF. **Note:** This parameter is optional if it is possible to dynamically resolve the IP.
username	string	1	User name for configuration.
password	string	1	Password for configuration access authentication.

vnfr:config-atrributes

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration/config-attributes

ID	Type	Cardinality	Description
config-priority	uint64	1	Order of configuration priority to be applied to each VNF in this network service. A low number takes precedence over a high number.
config-delay	uint64	1	Wait time (in seconds) before applying the configuration to this VNF.

vnfr:config-primitive

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration/config-primitive/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the service primitive.
parameter	list	0..n	List of parameters to the service primitive. See "vnfr:config-primitive:parameter" on page 240.
user-defined-script	string	1	A user-defined script. If a script is defined, the script will be executed using bash.

vnfr:config-primitive:parameter

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration/config-primitive/STRING/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of parameter
data-type	enum	1	Data type associated with the name. **Supported values:** • STRING • INTEGER • BOOLEAN
mandatory	boolean	1	[Default *false*] Defines whether this field is mandatory.

ID	Type	Cardinality	Description
default-value	string	1	The default value for the field. **Note:** This value is required to set the read-only and hidden parameters.
parameter-pool	string	1	NSD parameter pool name to use for this parameter.
read-only	boolean	1	If set to *true*, the value is dimmed in the UI. **Note:** This parameter can be set to true only on parameters that specify a default-value field.
hidden	boolean	1	If set to true, the value is hidden from view in the UI. **Note:** This parameter can be set to true only on parameters that specify a default-value field.
out	boolean	1	Specifies if this is an output of the primitive execution.

vnfr:initial-config-primitive

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration/initial-config-primitive/0

ID	Type	Cardinality	Description
seq	uint64	1	Sequence number for the configuration primitive.
name	string	1	Name of the configuration primitive.

ID	Type	Cardinality	Description
parameter	list	0..n	List of parameters to the configuration primitive. See "vnfr:initial-config-primitive:parameter" on page 242.
user-defined-script	string	1	A user-defined script
config-primitive-ref	leafref	1	Reference to a config primitive name. **Note:** The referenced config primitive should have all the input paramaters predefined either with default values or dependency references.

vnfr:initial-config-primitive:parameter

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vnf-configuration/initial-config-primitive/0/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the parameter.
value	string	1	Value of the parameter.

vnfr:mgmt-interface

Interface over which the **VNF** is managed.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/mgmt-interface

Fields

ID	Type	Cardinality	Description
ip-address	union	1	Specifies the static IP address for managing the VNF.
vdu-id	leafref	1	Use the default management interface on this VDU. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd/vnfd:vdu/vnfd:id
cp	leafref	1	Use the IP address for the VNFD associated with this connection point endpoint. This is a leafref to path: /vnfd:vnfd-catalog/vnfd:vnfd/vnfd:connection-point/vnfd:name
port	uint16	1	Port number for the management interface.
dashboard-params	container	1	Parameters for the VNF dashboard of the management interface. See "vnfr:dashboard-params" on page 244.

vnfr:dashboard-params

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/mgmt-interface/dashboard-params

ID	Type	Cardinality	Description
path	string	1	The HTTP path for the dashboard.
https	boolean	1	[Default *false*] Choose HTTPS instead of HTTP.
port	uint16	1	The HTTP port for the dashboard

vnfr:internal-vld

List of internal Virtual Link Descriptors (VLDs). Internal VLDs describe the basic topology of the connectivity—such as E-LAN— between internal VNF Components (**VNFC**) within the system.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/internal-vld/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Unique identifier for the internal VLD.
name	string	1	Name of the internal VLD.
short-name	string	1	Short name for the internal VLD to use as a label in the UI.
description	string	1	Description of the internal VLD.
type	enum	1	Type of virtual link. **Supported values:** ELAN: A multipoint service that connects a set of VDUs.
root-bandwidth	uint64	1	For ELAN this is the aggregate bandwidth.
leaf-bandwidth	uint64	1	For ELAN this is the bandwidth of branches.
internal-connection-point	list	0..n	List of internal connection points in this VLD. See "vnfr:internal-connection-point" on page 246.

ID	Type	Cardinality	Description
provider-network	container	1	Container for the provider network. See "vnfr:provider-network" on page 246.
vim-network-name	string	1	Name of pre-provisioned network in the VIM (cloud) account.
ip-profile-ref	string	1	Named reference to an ip-profile object.

vnfr:internal-connection-point

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/internal-vld/STRING/internal-connection-point/STRING

ID	Type	Cardinality	Description
id-ref	leafref	1	Reference to the internal connection point ID.

vnfr:provider-network

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/internal-vld/STRING/provider-network

ID	Type	Cardinality	Description
physical-network	string	1	Name of the physical network on which the provider network is built.

ID	Type	Cardinality	Description
overlay-type	enum	1	Identifies the type of the overlay network, which is a virtual network that is built on top of an existing network and is supported by its infrastructure. **Supported values:** • LOCAL — A network that can be realized on a single host only. • FLAT — The simplest networking environment in which each instance receives a fixed IP from the pool. All instances are attached to the same bridges. • VLAN — A network of computers in which the computers behaves as if they are connected to the same wire. However, the computers might be physically located on different segments of a LAN. • VXLAN — A proposed encapsulation protocol for running an overlay network on existing **Layer 3** infrastructure • GRE — GRE tunnels encapsulate isolated **Layer 2** network traffic in IP packets. Packets are routed between compute and networking nodes using the hosts' network connectivity and routing tables.
segmentation-id	uint32	1	Segmentation ID

vnfr:ip-profiles

List of IP profiles. IP profiles describe the IP characteristics for the virtual link.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/ip-profiles/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the IP profile.
description	string	1	Description of the IP profile.
ip-profile-params	container	1	Information about the IP profile. See "vnfr:ip-profile-params" on page 248

vnfr:ip-profile-params

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/ip-profiles/STRING/ip-profile-params

ID	Type	Cardinality	Description
ip-version	enum	1	[Default IPv4] Version of the Internet Protocol used.
subnet-address	union	1	Subnet IP prefix associated with this IP profile.
gateway-address	union	1	IP address of the default gateway associated with this IP profile.
security-group	string	1	Name of the security group.

ID	Type	Cardinality	Description
dns-server	list	0..n	List of DNS servers associated with this IP profile. See "vnfr:dns-server" on page 249.
dhcp-params	container	1	Container for DHCP parameters. See "vnfr:ip-profile-params" on page 248.
subnet-prefix-pool	string	1	VIM-specific reference to pre-created subnet prefix.

vnfr:dns-server

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/ip-profiles/STRING/ip-profile-params

ID	Type	Cardinality	Description
address	union	1	List of DNS servers associated with this IP profile.

vnfr:dhcp-params

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/ip-profiles/STRING/ip-profile-params/dhcp-params

ID	Type	Cardinality	Description
enabled	boolean	1	[Default *true*] Indicates if DHCP is enabled.
start-address	union	1	Start IP address of the IP address range associated with DHCP domain.
count	uint32	1	Size of the DHCP pool associated with DHCP domain.

vnfr:connection-point

List of external connection points, in which each VNF:

- Has one or more points that are used to connect a VNF to other VNFs or to external networks
- Exposes these connection points to the orchestrator (**NFVO**)

The orchestrator constructs network services by connecting the connection points between different VNFs.

The orchestrator uses **VLDs** and **VNFFGs** at the network service level to construct network services.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/connection-point/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the connection point.
id	string	1	Unique identifier of the connection point
short-name	string	1	Short name of the connection point to use as a label in the UI.
type	enum	1	Type of connection point. **Supported values:** VPORT: Virtual Port
port-security-enabled	boolean	1	Enables the port security for the port

ID	Type	Cardinality	Description
static-ip-address	union	1	Static IP address for the connection point

vnfr:vdu

A VDU is a basic part of VNF. VDUs are virtual machines that host the network function, such as:

- Virtual machine specification
- Computation properties (RAM size, disk size, memory page size, number of CPUs, number of cores per CPU, number of threads per core)
- Storage requirements
- Initiation and termination scripts
- High availability redundancy model
- Scale out/scale in limits

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Unique identifier for the VDU.
name	string	1	Unique name for the VDU.
description	string	1	Description of the VDU.
count	uint64	1	Number of instances of the VDU.

ID	Type	Cardinality	Description
mgmt-vpci	string	1	Specifies the virtual PCI address, expressed in the following format dddd:dd:dd.d. For example 0000:00:12.0. This information can be used to pass as metadata during the VM creation.
vm-flavor	container	1	Flavor of the virtual machine (VM) instance. See "vnfr:vm-flavor" on page 254.
guest-epa	container	1	EPA attributes for the guest operating system. See "vnfr:guest-epa" on page 255.
vswitch-epa	container	1	EPA attributes for Open vSwitch. See "vnfr:vswitch-epa" on page 260
hypervisor-epa	container	1	EPA attributes for the hypervisor. See "vnfr:hypervisor-epa" on page 261.
host-epa	container	1	EPA attributes for the host operating system. See "vnfr:host-epa" on page 262.
alarm	list	0..n	A list of alarms. See "vnfr:alarm" on page 267
image	string	1	Image name for the software image. If the image name is found within the VNF package it will be uploaded to all cloud accounts during the onboarding process. Otherwise, the image must be added to the cloud account with the same name as entered in this field.

ID	Type	Cardinality	Description
image-checksum	string	1	Image md5sum for the software image. The md5sum, if provided, along with the image name, uniquely identifies an image uploaded to the CAL.
cloud-init	string	1	Contents of cloud-init script, provided inline, in cloud-config format
cloud-init-file	string	1	Name of file with contents of cloud-init script in cloud-config format.
supplemental-boot-data	container	1	Container for custom VIM data. See "vnfr:supplemental-boot-data" on page 271.
internal-connection-point	list	0..n	List for internal connection points. See "vnfr:internal-connection-point" on page 273.
internal-interface	list	0..n	List of internal interfaces for the VNF. See "vnfr:internal-interface" on page 274.
external-interface	list	0..n	List of external interfaces for the VNF. See "vnfr:external-interface" on page 276.
volumes	list	0..n	List of disk-volumes to be attached to VDU. See "vnfr:volumes" on page 278.

vnfr:vm-flavor

Flavor is an alternative term for a VM instance type.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/vm-flavor

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/vm-flavor

Fields

ID	Type	Cardinality	Description
vpcu-count	uint16	1	Number of VCPUs for the VM.
memory-mb	uint64	1	Amount of memory in MB to allocate to the VM.
storage-gb	uint64	1	Amount of disk space in GB to allocate to the VM.

vnfr:guest-epa

EPA attributes for the guest operating system.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/guest-epa

Fields

ID	Type	Cardinality	Description
trusted-execution	boolean	1	If set to *true*, indicates this VM should be allocated from trusted pool.
mempage-size	enum	1	Memory page allocation size. **Supported values:** • LARGE: Require hugepages (either 2MB or 1GB) • SMALL: Doesn't require hugepages • SIZE_2MB: Requires 2MB hugepages • SIZE_1GB: Requires 1GB hugepages • PREFER_LARGE: Application prefers hugepages **Note:** If a VM requires hugepages, choose LARGE or SIZE_2MB or SIZE_1GB. If the VM prefers hugepages, choose PREFER_LARGE.
cpu-pinning-policy	enum	1	Describes the association between virtual CPUs in the guest and the physical CPUs in the host. **Supported values:** • DEDICATED: Virtual CPUs are pinned to physical CPUs • SHARED: Multiple VMs may share the same physical CPUs. • ANY: (Default) Any policy is acceptable for the VM

ID	Type	Cardinality	Description
cpu-thread-pinning-policy	enum	1	Describes how to place the guest CPUs when the host supports hyper threads. **Default values:** • AVOID: Avoids placing a guest on a host with threads. • SEPARATE: Places vCPUs on separate cores, and avoids placing two vCPUs on two threads of same core. • ISOLATE: Places each vCPU on a different core, and places no vCPUs from a different guest on the same core. • PREFER: Attempts to place vCPUs on threads of the same core.
pcie-device	list	0..1	List of PCIE passthrough devices. See "vnfr:pcie-device" on page 257
numa-unaware	empty		Details about the numa-node-policy are null.
numa-node-policy	container	1	Defines numa topology of the guest, specifying if the guest should run on a host with one numa node or multiple numa nodes. **Example:** A guest might need 8 VCPUs and 4 GB of memory with the VCPUs and memory distributed across multiple NUMA nodes. In this scenario, NUMA node 1 could run with 6 VCPUs and 3GB, and NUMA node 2 could run with 2 vcpus and 1GB. See "vnfr:numa-node-policy" on page 257

vnfr:pcie-device

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/guest-epa/pcie-device/STRING

ID	Type	Cardinality	Description
device-id	string	1	Device identifier.
count	uint64	1	Number of devices to attach to the VM.

vnfr:numa-node-policy

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/guest-epa/numa-node-policy

ID	Type	Cardinality	Description
node-cnt	uint16	1	Number of NUMA nodes to expose to the VM.
mem-policy	enum	1	Specifies how to allocate memory in a multi-node scenario. **Supported values:** • STRICT: The memory must be allocated from the memory attached to the NUMA node. • PREFERRED: The memory should be allocated from the memory attached to the NUMA node
node	list	0..n	List of NUMA nodes. See "vnfr:numa-node-policy:node" on page 258.

vnfr:numa-node-policy:node

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/guest-epa/numa-node-policy/node/0

ID	Type	Cardinality	Description
id	uint64	1	NUMA node identification. Typically 0 or 1.
vpcu	list	0..n	List of VPCUs to allocate on this NUMA node. See "vnfr:numa-node-policy:node:vcpu" on page 258
memory-mb	uint64	1	Memory size in MB for this NUMA node.
num-cores	uint8	1	Number of cores.
paired-threads	container	1	Container for paired threads. See "vnfr:numa-node-policy:node:paired-threads" on page 259.
num-threads	uint8	1	OpenMANO NUMA type selection.

vnfr:numa-node-policy:node:vcpu

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/guest-epa/numa-node-policy/node/0/vcpu/0

ID	Type	Cardinality	Description
id	uint64	1	List of VCPUs IDs to allocate on this NUMA node.

vnfr:numa-node-policy:node:paired-threads

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/guest-epa/numa-node-policy/node/0/paired-threads

ID	Type	Cardinality	Description
num-paired-threads	uint8	1	Number of paired-threads.
paired-thread-ids	list	0..n	List of thread paired to use in case of paired thread NUMA. See "vnfr:numa-node-policy:node:paired-threads:paired-thread-ids" on page 259

vnfr:numa-node-policy:node:paired-threads:paired-thread-ids

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/guest-epa/numa-node-policy/node/0/paired-threads/paired-thread-ids/0

ID	Type	Cardinality	Description
thread-a	uint8	1	Thread ID
thread-b	uint8	1	Thread ID

vnfr:vswitch-epa

EPA attributes for Open vSwitch.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/vswitch-epa

Fields

ID	Type	Cardinality	Description
ovs-acceleration	enum	1	Specifies Open vSwitch acceleration mode. **Supported values:** • MANDATORY: OVS acceleration is required • PREFERRED: OVS acceleration is preferred • DISABLED: OVS acceleration is disabled.
ovs-offload	enum	1	Specifies Open vSwitch hardware offload mode. **Supported values:** • MANDATORY: OVS offload is required • PREFERRED: OVS offload is preferred • DISABLED: OVS offload is disabled

vnfr:hypervisor-epa

EPA attributes for the hypervisor.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/hypervisor-epa

Fields

ID	Type	Cardinality	Description
type	enum	1	Specifies the type of hypervisor. For example, KVM, XEN. Value can be: • KVM: KVM • XEN: XEN
version	string	1	Version of the hypervisor.

vnfr:host-epa

Specifies the host-level EPA attributes.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/host-epa

Fields

ID	Type	Cardinality	Description
cpu-model	enum	1	Host CPU model. **Supported values:** • PREFER_WESTMERE • REQUIRE_WESTMERE • PREFER_SANDYBRIDGE • REQUIRE_SANDYBRIDGE • PREFER_IVYBRIDGE • REQUIRE_IVYBRIDGE • PREFER_HASWELL • REQUIRE_HASWELL • PREFER_BROADWELL • REQUIRE_BROADWELL • PREFER_NEHALEM • REQUIRE_NEHALEM • PREFER_PENRYN • REQUIRE_PENRYN • PREFER_CONROE • REQUIRE_CONROE • PREFER_CORE2DUO • REQUIRE_CORE2DUO

ID	Type	Cardinality	Description
cpu-arch	enum	1	Host CPU architecture. **Supported values:** • PREFER_X86 • REQUIRE_X86 • PREFER_X86_64 • REQUIRE_X86_64 • PREFER_I686 • REQUIRE_I686 • PREFER_IA64 • REQUIRE_IA64 • PREFER_ARMV7 • REQUIRE_ARMV7 • PREFER_ARMV8 • REQUIRE_ARMV8
cpu-vendor	enum	1	Host CPU vendor. **Supported values:** • PREFER_INTEL • REQUIRE_INTEL • PREFER_AMD • REQUIRE_AMD
cpu-socket-count	uint64	1	Number of sockets on the host.
cpu-core-count	uint64	1	Number of cores on the host.
cpu-core-thread-count	uint64	1	Number of threads per cores on the host.

ID	Type	Cardinality	Description
cpu-feature	list	0..1	List of CPU features. See "vnfr:cpu-feature" on page 264.
om-cpu-model-string	string	1	OpenMano CPU model string.
om-cpu-feature	list	0..n	OpenMano CPU features. See "vnfd:om-cpu-feature" on page 266.

vnfr:cpu-feature

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/host-epa/cpu-feature/PREFER_AES

ID	Type	Cardinality	Description

ID	Type	Cardinality	Description
feature	enum	1	**Enumeration for CPU features:** • AES: CPU supports advanced instruction set for AES (Advanced Encryption Standard). • CAT: Cache Allocation Technology (CAT) allows an operating system, hypervisor, or similar system management agent to specify the amount of L3 cache (currently the last-level cache in most server and client platforms) space an application can fill. **Note:** As a hint to hardware functionality, certain features, such as power management, may override CAT settings. • CMT: Cache Monitoring Technology (CMT) allows an Operating System, Hypervisor, or similar system management agent to determine the usage of cache based on applications running on the platform. The implementation is directed at L3 cache monitoring (currently the last-level cache in most server and client platforms). • DDIO: Intel Data Direct I/O (DDIO) enables Ethernet server NICs and controllers talk directly to the processor cache without a detour via system memory. This enumeration specifies if the VM requires a DDIO capable host. **Supported values:** • PREFER_AES • REQUIRE_AES • PREFER_CAT • REQUIRE_CAT • PREFER_CMT • REQUIRE_CMT • PREFER_DDIO • REQUIRE_DDIO • REQUIRE_VME • PREFER_VME • REQUIRE_DE • PREFER_DE • REQUIRE_PSE • PREFER_PSE • REQUIRE_TSC • PREFER_TSC

vnfd:om-cpu-feature

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/host-epa/om-cpu-feature/STRING

ID	Type	Cardinality	Description
feature	string	1	CPU feature.

vnfr:alarm

Information about alarms.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/alarm/STRING

Fields

ID	Type	Cardinality	Description
alarm-id	string	1	Reserved field for the identifier assigned by the **VIM** provider.
name	string	1	A human-readable string to identify the alarm.
description	string	1	Description of the alarm.
vdur-id	string	1	Identifier of the VDU record (VDUR) associated with this alarm.
actions	container	1	Actions related to the alarm. See "vnfr:actions" on page 269.
repeat	boolean	1	[Default *true*] Indicates whether the alarm should emit repeatedly after the associated threshold has been crossed.
enabled	boolean	1	[Default *true*] Indicates whether the alarm has been enabled or disabled.

ID	Type	Cardinality	Description
severity	enum	1	A measure of the important or urgency of the alarm. **Supported types:** • LOW • MODERATE • CRITICAL
metric	enum	1	Metric types that can be tracked by this alarm. **Supported values:** • CPU_UTILIZATION • MEMORY_UTILIZATION • STORAGE_UTILIZATION
statistic	enum	1	Type of statistic to use to measure a metric, which determines threshold crossing for an alarm. **Supported values:** • AVERAGE • MINIMUM • MAXIMUM • COUNT • SUM

ID	Type	Cardinality	Description
operation	enum	1	The relational operator used to define whether an alarm should be triggered when the metric statistic goes above or below a specified threshold value. **Supported values:** • GE — Greater than or equal to • LE — Less than or equal to • GT — Greater than • LT — Less than • EQ — Equal
value	decimal164	1	Defines the threshold (up to 4 fraction digits) that, if crossed, will trigger the alarm.
period	uint32	1	Defines the length of time (seconds) for which metric data are collected to evaluate the chosen statistic.
evaluation	uint32	1	Number of samples of the metric statistic used to evaluate threshold crossing. Each sample or evaluation is equal to the metric statistic obtained for a given period. **Note:** This value can be used to mitigate spikes in the metric that may skew the statistic of interest.

vnfr:actions

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/alarm/STRING/actions

ID	Type	Cardinality	Description
ok	list	0..n	See "vnfr:actions" on page 269.

ID	Type	Cardinality	Description
insufficient-data	list	0..n	See "vnfr:actions:insufficient-data" on page 270.
alarm	list	0..n	See "vnfr:actions:alarm" on page 270.

vnfr:actions:ok

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/alarm/STRING/actions/ok/STRING

ID	Type	Cardinality	Description
url	string	1	

vnfr:actions:insufficient-data

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/alarm/STRING/actions/insufficient-data/STRING

ID	Type	Cardinality	Description
url	string	1	

vnfr:actions:alarm

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/alarm/STRING/actions/alarm/STRING

ID	Type	Cardinality	Description
url	string	1	

vnfr:supplemental-boot-data

Container for custom **VIM** data.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/supplemental-boot-data

Fields

ID	Type	Cardinality	Description
config-file	list	0..n	List of configuration files to be mounted onto an additional drive. See "vnfr:config-file" on page 271.
custom-meta-data	list	0..n	List of metadata to be associated with the instance. See "vnfr:custom-meta-data" on page 272.
boot-data-drive	boolean	1	[Default *false*] Specifies whether the VIM should implement additional drives to host config-files or metadata.

vnfr:config-file

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/supplemental-boot-data/config-file/STRING

ID	Type	Cardinality	Description
source	string	1	Name of the configuration file.
dest	string	1	Full path of the destination in the guest.

vnfr:custom-meta-data

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/supplemental-boot-data/custom-meta-data/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the metadata parameter.
data-type	enum	1	Data type of the metadata parameter.
value	string	1	Value of the metadata parameter.

vnfr:internal-connection-point

List for internal connection points. Each VNFC has zero or more internal connection points. Internal connection points are used for connecting the VNF components internal to the VNF. If a VNF has only one VNFC, it may not have any internal connection points.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/internal-connection-point/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the connection point.
id	string	1	Identifier for the internal connection points.
short-name	string	1	Short name to use as a label in the UI.
type	enum	1	Type of connection point. **Supported values:** VPORT: Virtual Port
port-security-enabled	boolean		Specifies whether to enable port security for the port.
static-ip-address	union	1	Static IP address for the connection point
internal-vld-ref	leafref	1	This is a leafref to path: ../../../internal-vld/id

vnfr:internal-interface

Internal interfaces enable traffic between virtual network functions.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/internal-interface/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the internal interface inside the VDU. **Note:** This name has only local significance to the VDU.
vdu-internal-connection-point-ref	leafref	1	Reference to an internal connection point. This is a leafref to path: ../../internal-connection-point/id
virtual-interface	container	1	Container for the virtual interface properties. See "vnfr:virtual-interface" on page 275.

vnfr:virtual-interface

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/internal-interface/STRING/virtual-interface

ID	Type	Cardinality	Description
type	enum	1	Specifies the type of virtual interface between VM and host. **Supported values:** • VIRTIO: [Default] Use the traditional VIRTIO interface • PCI-PASSTHROUGH: Use PCI-PASSTHROUGH interface • SR-IOV: Use SR-IOV interface • E1000 : Emulate E1000 interface • RTL8139 : Emulate RTL8139 interface • PCNET : Emulate PCNET interface • OM-MGMT: Used to specify OpenMANO management internal-connection type
vpci	string	1	Specifies the virtual PCI address in format dddd:dd:dd.d. For example: 0000:00:12.0 This information can be used to pass as metadata during the VM creation.
bandwidth	uint64	1	Specifies the aggregate bandwidth requirement for the NIC.

vnfr:external-interface

External interfaces enable traffic between VNFs.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/external-interface/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the external interface inside the VDU. **Note:** This name has only local significance to the VDU.
vnfd-connection-point-ref	reference	1	Reference to an external connection point. This is a leafref to path: ../../../connection-point/name
virtual-interface	container	1	Virtual interface properties. See "vnfr:virtual-interface" on page 276.

vnfr:virtual-interface

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/external-interface/STRING/virtual-interface

ID	Type	Cardinality	Description

ID	Type	Cardinality	Description
type	enum	1	Specifies the type of virtual interface between VM and host. **Supported values:** • VIRTIO: [Default] Use the traditional VIRTIO interface • PCI-PASSTHROUGH: Use PCI-PASSTHROUGH interface • SR-IOV: Use SR-IOV interface • E1000 : Emulate E1000 interface • RTL8139 : Emulate RTL8139 interface • PCNET : Emulate PCNET interface • OM-MGMT: Used to specify OpenMANO management internal-connection type
vpci	string	1	Specifies the virtual PCI address in format dddd:dd:dd.d. For example: 0000:00:12.0 **Note:** This information can be used to pass as metadata during the VM creation.
bandwidth	uint64	1	Specifies the aggregate bandwidth requirement for the NIC.

vnfr:volumes

Defines disk volumes to be attached to the **VDU**, such as if a VNF requires multiple disks to boot the virtual machine.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/volumes/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the disk-volumes, such as vda, vdb.
description	string	1	Description for the volume.
size	uint64	1	Size of the disk, in GB.
ephemeral	empty		
image	string	1	Image name for the software image to be used. If the image name is found within the VNF package it will be uploaded to all cloud accounts during the onboarding process. Otherwise, the image must be added to the cloud account with the same name as entered in this field.
image-checksum	string	1	Image md5sum for the software image. The md5sum, if provided, along with the image name, uniquely identifies an image uploaded to the CAL.
volume-ref	string	1	Reference to the pre-existing volume in **VIM.**
boot-volume	boolean	1	If set to *true*, indicates that this is a boot volume

ID	Type	Cardinality	Description
boot-priority	int32		Boot priority associated with volume.
device_bus	enum	1	Type of disk-bus on which this disk is exposed to the guest operating system. **Supported values:** • IDE • USB • VIRTIO • SCSI
device_type	enum	1	Type of device as exposed to the guest operating system. **Supported values:** • DISK • CDROM • FLOPPY • LUN (logical unit number)
custom-meta-data	list	0..n	List of metadata to be associated with the instance. See "vnfr:custom-meta-data" on page 279

vnfr:custom-meta-data

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu/STRING/volumes/STRING/custom-meta-data/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the metadata parameter.

ID	Type	Cardinality	Description
data-type	enum	1	Data type of the metadata parameter.
value	string	1	Value of the metadata parameter.

vnfr:vdu-dependency

List of virtual deployment unit (**VDU**) dependencies, from which the orchestrator determines the order of startup among the VDUs.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/vdu-dependency/STRING

Fields

ID	Type	Cardinality	Description
vdu-source-ref	leafref	1	Identifier of the VDU. This is a leafref to path: ../../vdu/id
vdu-depends-on-ref	leafref	1	Reference to the VDU on which the source VDU depends. This is a leafref to path: ../../vdu/id

vnfr:http-endpoint

List of http endpoints to be used by monitoring parameters.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/http-endpoint/STRING

Fields

ID	Type	Cardinality	Description
path	string	1	The HTTP path on the management server.
https	boolean	1	[Default *false*] Pick HTTPS instead of HTTP.
port	uint16	1	HTTP port to connect to.
username	string	1	HTTP basic auth user name.
password	string	1	HTTP basic auth password.
polling_interval_secs	uint8	1	[Default 2] HTTP polling interval in seconds.
method	enum	1	Method to be performed at the URI. **Supported values:** • GET (default) • POST • PUT • GET • DELETE • PATCH • OPTIONS

ID	Type	Cardinality	Description
data	string	1	The data to be sent with POST.
headers	list	0..n	List of custom HTTP headers to put on the HTTP request. See "vnfr:headers" on page 283.

vnfr:headers

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/http-endpoint/STRING/headers/STRING

ID	Type	Cardinality	Description
key	string	1	HTTP header key.
value	string	1	HTTP header value.

vnfr:monitoring-param

List of monitoring parameters at the network service level.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/monitoring-param/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Identifier for the monitoring parameter.
name	string	1	Name of the monitoring parameter.
http-endpoint-ref	leafref	1	Reference to the HTTP endpoint. This is a leafref path: ../../../http-endpoint/path See "vnfd:http-endpoint" on page 215.
json-query-method	enum	1	The method to extract a value from a JSON response. **Supported values:** • NAMEKEY: [Default] Use the name as the key for a non-nested value. • JSONPATH: Use jsonpath-rw implementation to extract a value. • OBJECTPATH: Use objectpath implementation to extract a value.
json-query-params	container	1	Object for JSON query parameters. See "vnfr:json-query-params" on page 286.
description	string	1	Description of the monitoring parameter.

ID	Type	Cardinality	Description
group-tag	string	1	Tag to group monitoring parameters.
widget-type	enum	1	Type of the widget, typically used by the UI. **Supported values:** • HISTOGRAM • BAR • GAUGE • SLIDER • COUNTER • TEXTBOX
units	string	1	Units for the monitoring parameter, such as megabits per second.
value-type	enum	1	The type of the parameter value. **Supported values:** • INT (default) • DECIMAL • STRING
numeric-constraints	container	1	Constraints for the numbers. See "vnfr:numeric-constraints" on page 286.
text-constraints	container	1	Constraints for the text strings. See "vnfr:text-constraints" on page 287.
value-integer	int64	1	Current value for integer parameter.

ID	Type	Cardinality	Description
value-decimal	decimal164	1	Current value for decimal parameter, up to 4 fraction digits.
value-string	string	1	Current value for the string parameter.

vnfr:json-query-params

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/monitoring-param/STRING/json-query-params

ID	Type	Cardinality	Description
json-path	string	1	The JSON path used to extract value from the JSON structure.
object-path	string	1	The object path to use to extract value form the JSON structure.

vnfr:numeric-constraints

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/monitoring-param/STRING/numeric-constraints

ID	Type	Cardinality	Description
min-value	uint64	1	Minimum value for the parameter.
max-value	uint64	1	Maximum value for the parameter.

vnfr:text-constraints

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/monitoring-param/STRING/text-constraints

ID	Type	Cardinality	Description
min-length	uint8	1	Minimum string length for the parameter.
max-length	uint8	1	Maximum string length for the parameter.

vnfr:placement-groups

List of placement groups at VNF level. The placement group construct defines the compute resource placement strategy in a cloud environment.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/placement-groups/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Placement group name.
requirement	string	1	Describes the intent/rationale behind this placement group. Note: This free-text field is for human consumption only.
strategy	enum	1	Strategy associated with this placement group. **Supported values:** • COLOCATION: [Default] Share the physical infrastructure (hypervisor/network) among all members of this group. • ISOLATION: Do not share the physical infrastructure (hypervisor/network) among the members of this group
member-vdus	list	0..n	List of **VDUs** that are part of this placement group. See "vnfr:member-vdus" on page 289.

vnfr:member-vdus

/vnfr:vnfr-catalog/vnfr/STRING/vnfd/placement-groups/STRING/member-vdus/STRING

ID	Type	Cardinality	Description
member-vdu-ref	leafref	1	Reference to the VDU in the VNF. This is a leafref to path: ../../../vdu/id

vnfr:vnf-configuration

Information about the **VNF** configuration for the management interface.

Note: If the network service contains multiple instances of the same VNF, each VNF instance could have a different configuration.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration

Fields

ID	Type	Cardinality	Description
netconf	container	1	Use NETCONF for configuring the VNF. See "vnfr:netconf" on page 291.
rest	container	1	Use REST for configuring the VNF. See "vnfr:rest" on page 292.
script	container	1	Use a custom script for configuring the VNF. This script will be executed in the Orchestrator. All required dependencies for the script should be available in the Orchestrator system. See "vnfr:script" on page 292.
juju	container	1	Use Juju for configuring the VNF. See "vnfr:juju" on page 293.
config-access	container	1	IP address to be used to configure this VNF. See "vnfr:config-access" on page 293.

ID	Type	Cardinality	Description
config-attributes	container	1	Miscellaneous config attributes to be considered while processing the NSD to apply configuration. See "vnfr:config-atrributes" on page 293.
config-primitive	list	0..n	List of service primitives supported by the configuration agent for this VNF. See "vnfr:config-primitive" on page 294.
initial-config-primitive	list	0..n	Initial set of configuration primitives. See "vnfr:initial-config-primitive" on page 296.
config-template	string	1	Configuration template for each VNF.

vnfr:netconf

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration/netconf

ID	Type	Cardinality	Description
target	enum	1	NETCONF configuration target. **Supported values:** • RUNNING • CANDIDATE

ID	Type	Cardinality	Description
protocol	enum	1	Protocol to use for NETCONF. **Supported values:** • NONE • SSH
port	inet:port-number	1	Port for the NETCONF server.

vnfr:rest

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration/rest

ID	Type	Cardinality	Description
port	inet_port-number	1	Port for the REST server

vnfr:script

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration/script

ID	Type	Cardinality	Description
script-type	enum	1	Script type to use. **Supported values:** • BASH • EXPECT

vnfr:juju

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration/juju

ID	Type	Cardinality	Description
charm	string	1	Juju charm to use to use with the VNF.

vnfr:config-access

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration/config-access

ID	Type	Cardinality	Description
mgmt-ip-address	union	1	IP address to be used to configure this VNF. Note: This parameter is optional if it is possible to dynamically resolve the IP.
username	string	1	User name for configuration.
password	string	1	Password for configuration access authentication.

vnfr:config-atrributes

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration/config-attributes

ID	Type	Cardinality	Description
config-priority	uint64	1	Order of configuration priority to be applied to each VNF in this network service. A low number takes precedence over a high number.

ID	Type	Cardinality	Description
config-delay	uint64	1	Wait time (in seconds) before applying the configuration to this VNF.

vnfr:config-primitive

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration/config-primitive/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the service primitive.
parameter	list	0..n	List of parameters to the service primitive. See "vnfr:config-primitive:parameter" on page 294.
user-defined-script	string	1	A user-defined script. If a script is defined, the script will be executed using bash.

vnfr:config-primitive:parameter

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration/config-primitive/STRING

ID	Type	Cardinality	Description
name	string	1	Name of parameter

ID	Type	Cardinality	Description
data-type	enum	1	Data type associated with the name. **Supported values:** • STRING • INTEGER • BOOLEAN
mandatory	boolean	1	[Default *false*] Defines whether this field is mandatory.
default-value	string	1	The default value for the field. Note: This value is required to set the read-only and hidden parameters.
parameter-pool	string	1	NSD parameter pool name to use for this parameter.
read-only	boolean	1	If set to *true*, the value is dimmed in the UI. Note: This parameter can be set to true only on parameters that specify a default-value field.
hidden	boolean	1	If set to true, the value is hidden from view in the UI. Note: This parameter can be set to true only on parameters that specify a default-value field.
out	boolean	1	Specifies if this is an output of the primitive execution.

vnfr:initial-config-primitive

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration/initial-config-primitive/0

ID	Type	Cardinality	Description
seq	uint64	1	Sequence number for the configuration primitive.
name	string	1	Name of the configuration primitive.
parameter	list	0..n	List of parameters to the configuration primitive. See "vnfr:initial-config-primitive:parameter" on page 296.
user-defined-script	string	1	A user-defined script
config-primitive-ref	leafref	1	Reference to a config primitive name. Note: The referenced config primitive should have all the input paramaters predefined either with default values or dependency references.

vnfr:initial-config-primitive:parameter

/vnfr:vnfr-catalog/vnfr/STRING/vnf-configuration/initial-config-primitive/0/parameter/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the parameter.
value	string	1	Value of the parameter.

vnfr:mgmt-interface

Interface over which the **VNF** is managed.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/mgmt-interface

Fields

ID	Type	Cardinality	Description
ip-address	union	1	Specifies the static IP address for managing the VNF.
port	uint16	1	Port number for the management interface.
ssh-key	container	1	SSH key pair used for this VNF. See "vnfr:ssh-key" on page 297.

vnfr:ssh-key

/vnfr:vnfr-catalog/vnfr/STRING/mgmt-interface/ssh-key

ID	Type	Cardinality	Description
public-key	string	1	Public key configured on this VNF.
private-key-file	string	1	Path to the private key file.

vnfr:internal-vlr

References to a virtual link record (VLR) in the VLR catalog.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/internal-vlr/STRING

Fields

ID	Type	Cardinality	Description
vlr-ref	leafref	1	Reference to a virtual link record in the VLR catalog.
internal-connection-point	leafref	1	Reference to one or more internal connection points. This is a leadref path to: ../../vdur/internal-connection-point/id

vnfr:connection-point

List of external connection points, in which each VNF:

- Has one or more points that are used to connect a VNF to other VNFs or to external networks
- Exposes these connection points to the orchestrator (**NFVO**)

The orchestrator constructs network services by connecting the connection points between different VNFs.

The orchestrator uses **VLDs** and **VNFFGs** at the network service level to construct network services.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/connection-point/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the connection point.
id	string	1	Unique identifier of the connection points.
short-name	string	1	Short name of the connection point to use as a label in the UI.
type	enum	1	Type of connection point. **Supported values:** VPORT: Virtual Port
port-security-enabled	boolean	1	Enables the port security for the port.
static-ip-address	union	1	Static IP address for the connection point

ID	Type	Cardinality	Description
vlr-ref	leafref	1	Reference to the VLR associated with this connection point.
ip-address	union	1	IP address assigned to the external connection point.
mac-address	string	1	MAC address assigned to the external connection point.
connection-point-id	string	1	Identifier for the connection point.

vnfr:vdur

List of virtual deployment units (VDUR).

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Unique identifier for the VDU.
name	string	1	Unique name for the VDU.
vdu-id-ref	leafref	1	
vim-id	string	1	Allocated VM resource ID.
flavor-id	string	1	VIM assigned flavor ID.
image-id	string	1	VIM assigned flavor ID.
management-ip	union	1	Management IP address.
vm-management-ip	union	1	VM private management IP address.
console-url	string	1	Console URL for this VDU, if available.
vm-flavor	container	1	Flavor of the virtual machine (VM) instance. See "vnfr:vm-flavor" on page 254.

ID	Type	Cardinality	Description
guest-epa	container	1	EPA attributes for the guest operating system. See "vnfr:guest-epa" on page 255.
vswitch-epa	container	1	EPA attributes for Open vSwitch. See "vnfr:vswitch-epa" on page 260.
hypervisor-epa	container	1	EPA attributes for the hypervisor. See "vnfr:hypervisor-epa" on page 261.
host-epa	container	1	EPA attributes for the host operating system. See "vnfr:host-epa" on page 262.
supplemental-boot-data	container	1	Container for custom VIM data. See .
volumes	list	0..n	List of disk-volumes to be attached to VDU. See "vnfd:volumes" on page 211.
alarm	list	0..n	A list of alarms. See "vnfd:alarm" on page 200.
internal-connection-point	list	0..n	List for internal connection points. See "vnfd:internal-connection-point" on page 206.
internal-interface	list	0..n	List of internal interfaces for the VNF. See "vnfd:internal-interface" on page 207.

ID	Type	Cardinality	Description
external-interface	list	0..n	List of external interfaces for the VNF. See "vnfd:external-interface" on page 209.
operational-status	enum	1	The operational status of the VNFR instance init: • init : The VDU has just started • vm-init-phase : The VDUs in the VNF is being created in VIM • vm-alloc-pending : The VM alloc is pending in VIM • running : The VDU is active in VM • terminate : The VDU is being terminated • vm-terminate-phase: The VDU in the VNF is being terminated in VIM • terminated : The VDU is in the terminated state • failed : The VDU instantiation failed This element uses manotypes:monitoring-param
placement-groups-info	list	0..n	Placement groups to which this VDU belongs and its cloud construct

vnfr:vm-flavor

Flavor is an alternative term for a VM instance type.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/vm-flavor

Fields

ID	Type	Cardinality	Description
vpcu-count	uint16	1	Number of VCPUs for the VM.
memory-mb	uint64	1	Amount of memory in MB to allocate to the VM.
storage-gb	uint64	1	Amount of disk space in GB to allocate to the VM.

vnfr:guest-epa

EPA attributes for the guest operating system.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/guest-epa

Fields

ID	Type	Cardinality	Description
trusted-execution	boolean	1	If set to *true*, indicates this VM should be allocated from trusted pool.
mempage-size	enum	1	Memory page allocation size. **Supported values:** • LARGE: Require hugepages (either 2MB or 1GB) • SMALL: Doesn't require hugepages • SIZE_2MB: Requires 2MB hugepages • SIZE_1GB: Requires 1GB hugepages • PREFER_LARGE: Application prefers hugepages Note: If a VM requires hugepages, choose LARGE or SIZE_2MB or SIZE_1GB. If the VM prefers hugepages, choose PREFER_LARGE.
cpu-pinning-policy	enum	1	Describes the association between virtual CPUs in the guest and the physical CPUs in the host. **Supported values:** • DEDICATED: Virtual CPUs are pinned to physical CPUs • SHARED: Multiple VMs may share the same physical CPUs. • ANY: (Default) Any policy is acceptable for the VM

ID	Type	Cardinality	Description
cpu-thread-pinning-policy	enum	1	Describes how to place the guest CPUs when the host supports hyper threads. **Default values:** • AVOID: Avoids placing a guest on a host with threads. • SEPARATE: Places vCPUs on separate cores, and avoids placing two vCPUs on two threads of same core. • ISOLATE: Places each vCPU on a different core, and places no vCPUs from a different guest on the same core. • PREFER: Attempts to place vCPUs on threads of the same core.
pcie-device	list	0..1	List of PCIE passthrough devices. See "vnfr:pcie-device" on page 307
numa-unaware	empty		Details about the numa-node-policy are null.
numa-node-policy	container	1	Defines numa topology of the guest, specifying if the guest should run on a host with one numa node or multiple numa nodes. **Example:** A guest might need 8 VCPUs and 4 GB of memory with the VCPUs and memory distributed across multiple NUMA nodes. In this scenario, NUMA node 1 could run with 6 VCPUs and 3GB, and NUMA node 2 could run with 2 vcpus and 1GB. See "vnfr:numa-node-policy" on page 307

vnfr:pcie-device

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/guest-epa/pcie-device/STRING

ID	Type	Cardinality	Description
device-id	string	1	Device identifier.
count	uint64	1	Number of devices to attach to the VM.

vnfr:numa-node-policy

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/guest-epa/numa-node-policy

ID	Type	Cardinality	Description
node-cnt	uint16	1	Number of NUMA nodes to expose to the VM.
mem-policy	enum	1	Specifies how to allocate memory in a multi-node scenario. **Supported values:** • STRICT: The memory must be allocated from the memory attached to the NUMA node. • PREFERRED: The memory should be allocated from the memory attached to the NUMA node
node	list	0..n	List of NUMA nodes. See "vnfr:numa-node-policy:node" on page 308.

vnfr:numa-node-policy:node

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/guest-epa/numa-node-policy/node/0

ID	Type	Cardinality	Description
id	uint64	1	NUMA node identification. Typically 0 or 1.
vpcu	list	0..n	List of VPCUs to allocate on this NUMA node. See "vnfr:numa-node-policy:node:vcpu" on page 308
memory-mb	uint64	1	Memory size in MB for this NUMA node.
num-cores	uint8	1	Number of cores.
paired-threads	container	1	Container for paired threads. See "vnfr:numa-node-policy:node:paired-threads" on page 309.
num-threads	uint8	1	OpenMANO NUMA type selection.

vnfr:numa-node-policy:node:vcpu

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/guest-epa/numa-node-policy/node/0/vcpu/0

ID	Type	Cardinality	Description
id	uint64	1	List of VCPUs IDs to allocate on this NUMA node.

vnfr:numa-node-policy:node:paired-threads

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/guest-epa/numa-node-policy/node/0/paired-threads

ID	Type	Cardinality	Description
num-paired-threads	uint8	1	Number of paired-threads.
paired-thread-ids	list	0..n	List of thread paired to use in case of paired thread NUMA. See "vnfr:numa-node-policy:node:paired-threads:paired-thread-ids" on page 309

vnfr:numa-node-policy:node:paired-threads:paired-thread-ids

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/guest-epa/numa-node-policy/node/0/paired-threads/paired-thread-ids/0

ID	Type	Cardinality	Description
thread-a	uint8	1	Thread ID
thread-b	uint8	1	Thread ID

vnfr:vswitch-epa

EPA attributes for Open vSwitch.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/vswitch-epa

Fields

ID	Type	Cardinality	Description
ovs-acceleration	enum	1	Specifies Open vSwitch acceleration mode. **Supported values:** • MANDATORY: OVS acceleration is required • PREFERRED: OVS acceleration is preferred • DISABLED: OVS acceleration is disabled.
ovs-offload	enum	1	Specifies Open vSwitch hardware offload mode. **Supported values:** • MANDATORY: OVS offload is required • PREFERRED: OVS offload is preferred • DISABLED: OVS offload is disabled

vnfr:hypervisor-epa

EPA attributes for the hypervisor.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/hypervisor-epa

Fields

ID	Type	Cardinality	Description
type	enum	1	Specifies the type of hypervisor. For example, KVM, XEN. Value can be: • KVM: KVM • XEN: XEN
version	string	1	Version of the hypervisor.

vnfr:host-epa

Specifies the host-level EPA attributes.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/host-epa

Fields

ID	Type	Cardinality	Description
cpu-model	enum	1	Host CPU model. **Supported values:** • PREFER_WESTMERE • REQUIRE_WESTMERE • PREFER_SANDYBRIDGE • REQUIRE_SANDYBRIDGE • PREFER_IVYBRIDGE • REQUIRE_IVYBRIDGE • PREFER_HASWELL • REQUIRE_HASWELL • PREFER_BROADWELL • REQUIRE_BROADWELL • PREFER_NEHALEM • REQUIRE_NEHALEM • PREFER_PENRYN • REQUIRE_PENRYN • PREFER_CONROE • REQUIRE_CONROE • PREFER_CORE2DUO • REQUIRE_CORE2DUO

ID	Type	Cardinality	Description
cpu-arch	enum	1	Host CPU architecture. **Supported values:** • PREFER_X86 • REQUIRE_X86 • PREFER_X86_64 • REQUIRE_X86_64 • PREFER_I686 • REQUIRE_I686 • PREFER_IA64 • REQUIRE_IA64 • PREFER_ARMV7 • REQUIRE_ARMV7 • PREFER_ARMV8 • REQUIRE_ARMV8
cpu-vendor	enum	1	Host CPU vendor. **Supported values:** • PREFER_INTEL • REQUIRE_INTEL • PREFER_AMD • REQUIRE_AMD
cpu-socket-count	uint64	1	Number of sockets on the host.
cpu-core-count	uint64	1	Number of cores on the host.
cpu-core-thread-count	uint64	1	Number of threads per cores on the host.

ID	Type	Cardinality	Description
cpu-feature	list	0..1	List of CPU features. See "vnfr:cpu-feature" on page 314.
om-cpu-model-string	string	1	OpenMano CPU model string.
om-cpu-feature	list	0..n	OpenMano CPU features. See "vnfd:om-cpu-feature" on page 316.

vnfr:cpu-feature

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/host-epa/cpu-feature/PREFER_AES

ID	Type	Cardinality	Description

ID	Type	Cardinality	Description
feature	enum	1	**Enumeration for CPU features:** • AES: CPU supports advanced instruction set for AES (Advanced Encryption Standard). • CAT: Cache Allocation Technology (CAT) allows an operating system, hypervisor, or similar system management agent to specify the amount of L3 cache (currently the last-level cache in most server and client platforms) space an application can fill. Note: As a hint to hardware functionality, certain features, such as power management, may override CAT settings. • CMT: Cache Monitoring Technology (CMT) allows an Operating System, Hypervisor, or similar system management agent to determine the usage of cache based on applications running on the platform. The implementation is directed at L3 cache monitoring (currently the last-level cache in most server and client platforms). • DDIO: Intel Data Direct I/O (DDIO) enables Ethernet server NICs and controllers talk directly to the processor cache without a detour via system memory. This enumeration specifies if the VM requires a DDIO capable host. **Supported values:** • PREFER_AES • REQUIRE_AES • PREFER_CAT • REQUIRE_CAT • PREFER_CMT • REQUIRE_CMT • PREFER_DDIO • REQUIRE_DDIO • REQUIRE_VME • PREFER_VME • REQUIRE_DE • PREFER_DE • REQUIRE_PSE • PREFER_PSE • REQUIRE_TSC • PREFER_TSC

vnfd:om-cpu-feature

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/host-epa/om-cpu-feature/STRING

ID	Type	Cardinality	Description
feature	string	1	CPU feature.

V

nfr:supplemental-boot-data

Container for custom VIM data.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/supplemental-boot-data

Fields

ID	Type	Cardinality	Description
config-file	list	0..n	List of configuration files to be mounted onto an additional drive. See "vnfr:config-file" on page 317.
custom-meta-data	list	0..n	List of metadata to be associated with the instance. See "vnfr:custom-meta-data" on page 318.
boot-data-drive	boolean	1	[Default *false*] Specifies whether the VIM should implement additional drives to host config-files or metadata.

vnfr:config-file

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/supplemental-boot-data/config-file/STRING

ID	Type	Cardinality	Description
source	string	1	Name of the configuration file.
dest	string	1	Full path of the destination in the guest.

vnfr:custom-meta-data

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/supplemental-boot-data/custom-meta-data/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the metadata parameter.
data-type	enum	1	Data type of the metadata parameter.
value	string	1	Value of the metadata parameter.

vnfr:volumes

Defines disk volumes to be attached to the **VDU**, such as if a VNF requires multiple disks to boot the virtual machine.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/volumes/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the disk-volumes, such as vda, vdb.
description	string	1	Description for the volume.
size	uint64	1	Size of the disk, in GB.
ephemeral	empty		
image	string	1	Image name for the software image to be used. If the image name is found within the VNF package it will be uploaded to all cloud accounts during the onboarding process. Otherwise, the image must be added to the cloud account with the same name as entered in this field.
image-checksum	string	1	Image md5sum for the software image. The md5sum, if provided, along with the image name, uniquely identifies an image uploaded to the CAL.
volume-ref	string	1	Reference to the pre-existing volume in **VIM.**
boot-volume	boolean	1	If set to *true*, indicates that this is a boot volume

ID	Type	Cardinality	Description
boot-priority	int32		Boot priority associated with volume.
device_bus	enum	1	Type of disk-bus on which this disk is exposed to the guest operating system. **Supported values:** • IDE • USB • VIRTIO • SCSI
device_type	enum	1	Type of device as exposed to the guest operating system. **Supported values:** • DISK • CDROM • FLOPPY • LUN (logical unit number)
custom-meta-data	list	0..n	List of metadata to be associated with the instance. See "vnfr:custom-meta-data" on page 320

vnfr:custom-meta-data

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/volumes/STRING/custom-meta-data/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the metadata parameter.

ID	Type	Cardinality	Description
data-type	enum	1	Data type of the metadata parameter.
value	string	1	Value of the metadata parameter.

vnfr:alarms

vnfr:alarm

A list of the alarms that have been created for this VDU.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/alarms/STRING

Fields

ID	Type	Cardinality	Description
alarm-id	string	1	Reserved field for the identifier assigned by the **VIM** provider.
name	string	1	A human-readable string to identify the alarm.
description	string	1	Description of the alarm.
vdur-id	string	1	Identifier of the VDU record (VDUR) associated with this alarm.
actions	container	1	Actions related to the alarm. See "vnfr:actions" on page 324.
repeat	boolean	1	[Default *true*] Indicates whether the alarm should emit repeatedly after the associated threshold has been crossed.
enabled	boolean	1	[Default *true*] Indicates whether the alarm has been enabled or disabled.

ID	Type	Cardinality	Description
severity	enum	1	A measure of the important or urgency of the alarm. **Supported types:** • LOW • MODERATE • CRITICAL
metric	enum	1	Metric types that can be tracked by this alarm. **Supported values:** • CPU_UTILIZATION • MEMORY_UTILIZATION • STORAGE_UTILIZATION
statistic	enum	1	Type of statistic to use to measure a metric, which determines threshold crossing for an alarm. **Supported values:** • AVERAGE • MINIMUM • MAXIMUM • COUNT • SUM

ID	Type	Cardinality	Description
operation	enum	1	The relational operator used to define whether an alarm should be triggered when the metric statistic goes above or below a specified threshold value. **Supported values:** • GE — Greater than or equal to • LE — Less than or equal to • GT — Greater than • LT — Less than • EQ — Equal
value	decimal164	1	Defines the threshold (up to 4 fraction digits) that, if crossed, will trigger the alarm.
period	uint32	1	Defines the length of time (seconds) for which metric data are collected to evaluate the chosen statistic.
evaluation	uint32	1	Number of samples of the metric statistic used to evaluate threshold crossing. Each sample or evaluation is equal to the metric statistic obtained for a given period. Note: This value can be used to mitigate spikes in the metric that may skew the statistic of interest.

vnfr:actions

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/alarms/STRING/actions

ID	Type	Cardinality	Description
ok	list	0..n	See "vnfr:actions" on page 324.

ID	Type	Cardinality	Description
insufficient-data	list	0..n	See "vnfr:actions:insufficient-data" on page 325.
alarm	list	0..n	See "vnfr:actions:alarm" on page 325.

vnfr:actions:ok

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/alarms/STRING/actions/ok/STRING

ID	Type	Cardinality	Description
url	string	1	

vnfr:actions:insufficient-data

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/alarms/STRING/actions/insufficient-data/STRING

ID	Type	Cardinality	Description
url	string	1	

vnfr:actions:alarm

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/alarms/STRING/actions/alarm/STRING

ID	Type	Cardinality	Description
url	string	1	

vnfr:internal-connection-point

List for internal connection points. Each VNFC has zero or more internal connection points. Internal connection points are used for connecting the VNF components internal to the VNF. If a VNF has only one VNFC, it may not have any internal connection points.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/internal-connection-point/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the connection point.
id	string	1	Identifier for the internal connection points.
short-name	string	1	Short name to use as a label in the UI.
type	enum	1	Type of connection point. **Supported values:** VPORT: Virtual Port
port-security-enabled	boolean		Specifies whether to enable port security for the port.
static-ip-address	union	1	Static IP address for the connection point
internal-vld-ref	leafref	1	This is a leafref to path: ../../../internal-vld/id

vnfr:internal-interface

Internal interfaces enable traffic between virtual network functions.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/internal-interface/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the internal interface inside the VDU. Note: This name has only local significance to the VDU.
vdu-internal-connection-point-ref	leafref	1	Reference to an internal connection point. This is a leafref to path: ../../internal-connection-point/id
virtual-interface	container	1	Container for the virtual interface properties. See "vnfr:virtual-interface" on page 327.

vnfr:virtual-interface

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/internal-interface/STRING/virtual-interface

ID	Type	Cardinality	Description

ID	Type	Cardinality	Description
type	enum	1	Specifies the type of virtual interface between VM and host. **Supported values:** • VIRTIO: [Default] Use the traditional VIRTIO interface • PCI-PASSTHROUGH: Use PCI-PASSTHROUGH interface • SR-IOV: Use SR-IOV interface • E1000 : Emulate E1000 interface • RTL8139 : Emulate RTL8139 interface • PCNET : Emulate PCNET interface • OM-MGMT: Used to specify OpenMANO management internal-connection type
vpci	string	1	Specifies the virtual PCI address in format dddd:dd:dd.d. For example: 0000:00:12.0 This information can be used to pass as metadata during the VM creation.
bandwidth	uint64	1	Specifies the aggregate bandwidth requirement for the NIC.

vnfr:external-interface

External interfaces enable traffic between VNFs.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/external-interface/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Name of the external interface inside the VDU. Note: This name has only local significance to the VDU.
vnfd-connection-point-ref	reference	1	Reference to an external connection point. This is a leafref to path: ../../../connection-point/name
virtual-interface	container	1	Virtual interface properties. See "vnfr:virtual-interface" on page 329.

vnfr:virtual-interface

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/external-interface/STRING/virtual-interface

ID	Type	Cardinality	Description

ID	Type	Cardinality	Description
type	enum	1	Specifies the type of virtual interface between VM and host. **Supported values:** • VIRTIO: [Default] Use the traditional VIRTIO interface • PCI-PASSTHROUGH: Use PCI-PASSTHROUGH interface • SR-IOV: Use SR-IOV interface • E1000 : Emulate E1000 interface • RTL8139 : Emulate RTL8139 interface • PCNET : Emulate PCNET interface • OM-MGMT: Used to specify OpenMANO management internal-connection type
vpci	string	1	Specifies the virtual PCI address in format dddd:dd:dd.d. For example: 0000:00:12.0 Note: This information can be used to pass as metadata during the VM creation.
bandwidth	uint64	1	Specifies the aggregate bandwidth requirement for the NIC.

vnfr:placement-groups-info

Placement groups to which this VDU belongs and its cloud construct.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/placement-groups-info/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Placement group name.
requirement	string	1	Describes the intent/rationale behind this placement group. Note: This free-text field is for human consumption only.
strategy	enum	1	Strategy associated with this placement group. **Supported values:** • COLOCATION: [Default] Share the physical infrastructure (hypervisor/network) among all members of this group. • ISOLATION: Do not share the physical infrastructure (hypervisor/network) among the members of this group

ID	Type	Cardinality	Description
cloud-type	enum	1	Cloud account type: • aws • cloudsim • cloudsim_proxy • mock • openmano • openstack • vsphere • openvim • prop_cloud1
availability-zone	container	1	Name of the availability zone. See "vnfr:placement-groups-info" on page .
server-group	container	1	Name of the affinity/anti-affinity server group. See "vnfr:server-group" on page 333.
host-aggregate	list	0..n	Name of the host aggregate. See "vnfr:host-aggregate" on page 333.
aws-construct	empty		
openmano-construct	empty		
vsphere-construct	empty		
mock-construct	empty		

ID	Type	Cardinality	Description
cloudsim-construct	empty		

vnfr:availability-zone

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/placement-groups-info/STRING/availability-zone

ID	Type	Cardinality	Description
name	string	1	Name of the availability zone.

vnfr:server-group

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/placement-groups-info/STRING/server-group

ID	Type	Cardinality	Description
name	string	1	Name of the affinity/anti-affinity server group.

vnfr:host-aggregate

/vnfr:vnfr-catalog/vnfr/STRING/vdur/STRING/placement-groups-info/STRING/host-aggregate/STRING

ID	Type	Cardinality	Description
metadata-key	string	1	

ID	Type	Cardinality	Description
metadata-value	string	1	

vnfr:http-endpoint

List of http endpoints to be used by monitoring parameters.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/http-endpoint/STRING

Fields

ID	Type	Cardinality	Description
path	string	1	The HTTP path on the management server.
https	boolean	1	[Default *false*] Pick HTTPS instead of HTTP.
port	uint16	1	HTTP port to connect to.
username	string	1	HTTP basic auth user name.
password	string	1	HTTP basic auth password.
polling_interval_secs	uint8	1	[Default 2] HTTP polling interval in seconds.
method	enum	1	Method to be performed at the URI. **Supported values:** • GET (default) • POST • PUT • GET • DELETE • PATCH • OPTIONS

ID	Type	Cardinality	Description
data	string	1	The data to be sent with POST.
headers	list	0..n	List of custom HTTP headers to put on the HTTP request. See "vnfr:headers" on page 336.

vnfr:headers

/vnfr:vnfr-catalog/vnfr/STRING/http-endpoint/STRING/headers/STRING

ID	Type	Cardinality	Description
key	string	1	HTTP header key.
value	string	1	HTTP header value.

vnfr:monitoring-param

List of monitoring parameters at the network service level.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/monitoring-param/STRING

Fields

ID	Type	Cardinality	Description
id	string	1	Identifier for the monitoring parameter.
name	string	1	Name of the monitoring parameter.
http-endpoint-ref	leafref	1	Reference to the HTTP endpoint. This is a leafref path: ../../../http-endpoint/path See "vnfd:http-endpoint" on page 215.
json-query-method	enum	1	The method to extract a value from a JSON response. **Supported values:** • NAMEKEY: [Default] Use the name as the key for a non-nested value. • JSONPATH: Use jsonpath-rw implementation to extract a value. • OBJECTPATH: Use objectpath implementation to extract a value.
json-query-params	container	1	Object for JSON query parameters. See "vnfr:json-query-params" on page 339.

ID	Type	Cardinality	Description
description	string	1	Description of the monitoring parameter.
group-tag	string	1	Tag to group monitoring parameters.
widget-type	enum	1	Type of the widget, typically used by the UI. **Supported values:** • HISTOGRAM • BAR • GAUGE • SLIDER • COUNTER • TEXTBOX
units	string	1	Units for the monitoring parameter, such as megabits per second.
value-type	enum	1	The type of the parameter value. **Supported values:** • INT (default) • DECIMAL • STRING
numeric-constraints	container	1	Constraints for the numbers. See "vnfr:numeric-constraints" on page 339.
text-constraints	container	1	Constraints for the text strings. See "vnfr:text-constraints" on page 340.
value-integer	int64	1	Current value for integer parameter.

ID	Type	Cardinality	Description
value-decimal	decimal164	1	Current value for decimal parameter, up to 4 fraction digits.
value-string	string	1	Current value for the string parameter.

vnfr:json-query-params

/vnfr:vnfr-catalog/vnfr/STRING/monitoring-param/STRING/json-query-params

ID	Type	Cardinality	Description
json-path	string	1	The JSON path used to extract value from the JSON structure.
object-path	string	1	The object path to use to extract value form the JSON structure.

vnfr:numeric-constraints

/vnfr:vnfr-catalog/vnfr/STRING/monitoring-param/STRING/numeric-constraints

ID	Type	Cardinality	Description
min-value	uint64	1	Minimum value for the parameter.
max-value	uint64	1	Maximum value for the parameter.

vnfr:text-constraints

/vnfr:vnfr-catalog/vnfr/STRING/monitoring-param/STRING/text-constraints

ID	Type	Cardinality	Description
min-length	uint8	1	Minimum string length for the parameter.
max-length	uint8	1	Maximum string length for the parameter.

vnfr:placement-groups-info

Placement groups to which this VDU belongs and its cloud construct.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/placement-groups-info/STRING

Fields

ID	Type	Cardinality	Description
name	string	1	Placement group name.
requirement	string	1	Describes the intent/rationale behind this placement group. Note: This free-text field is for human consumption only.
strategy	enum	1	Strategy associated with this placement group. **Supported values:** • COLOCATION: [Default] Share the physical infrastructure (hypervisor/network) among all members of this group. • ISOLATION: Do not share the physical infrastructure (hypervisor/network) among the members of this group

ID	Type	Cardinality	Description
cloud-type	enum	1	Cloud account type: • aws • cloudsim • cloudsim_proxy • mock • openmano • openstack • vsphere • openvim • prop_cloud1
availability-zone	container	1	Name of the availability zone. See "vnfr:placement-groups-info" on page .
server-group	container	1	Name of the affinity/anti-affinity server group. See "vnfr:server-group" on page 343.
host-aggregate	list	0..n	Name of the host aggregate. See "vnfr:host-aggregate" on page 343.
aws-construct	empty		
openmano-construct	empty		
vsphere-construct	empty		
mock-construct	empty		

ID	Type	Cardinality	Description
cloudsim-construct	empty		

vnfr:availability-zone

/vnfr:vnfr-catalog/vnfr/STRING/placement-groups-info/STRING/availability-zone

ID	Type	Cardinality	Description
name	string	1	Name of the availability zone.

vnfr:server-group

/vnfr:vnfr-catalog/vnfr/STRING/placement-groups-info/STRING/server-group

ID	Type	Cardinality	Description
name	string	1	Name of the affinity/anti-affinity server group.

vnfr:host-aggregate

/vnfr:vnfr-catalog/vnfr/STRING/placement-groups-info/STRING/host-aggregate/STRING

ID	Type	Cardinality	Description
metadata-key	string	1	
metadata-value	string	1	

vnfr:cloud-config

List of public keys and users.

REST URI path

/vnfr:vnfr-catalog/vnfr/STRING/cloud-config

Fields

ID	Type	Cardinality	Description
key-pair	list	0..n	Used to configure the list of public keys to be injected as part of network service instantiation. See "vnfr:key-pair" on page 344.
user	list	0..n	List of users to be added through cloud-config. See "vnfr:cloud-config" on page .

vnfr:key-pair

/vnfr:vnfr-catalog/vnfr/STRING/cloud-config/key-pair/STRING

ID	Type	Cardinality	Description
name	string	1	Name of this key pair.
key	string	1	Key associated with this key pair.

vnfr:user

/vnfr:vnfr-catalog/vnfr/STRING/cloud-config/user/STRING

ID	Type	Cardinality	Description
name	string	1	Name of the user.
user-info	string	1	The user's real name.
key-pair	string	1	Used to configure the list of public keys to be injected as part of metwprk service instantiation. See "vnfr:key-pair" on page 345.

vnfr:key-pair

/vnfr:vnfr-catalog/vnfr/STRING/cloud-config/user/STRING/key-pair/STRING

ID	Type	Cardinality	Description
name	string	1	Name of this key pair.
key	string	1	Key associated with this key pair.

API Examples

This section provides API examples for configuring a cloud account, uploading an image, descriptor configuration, onboarding descriptors and starting, monitoring, and terminating network services. It also provides examples for executing config primitives on a network service and working with scaling instances.

Notes:

For some of the examples, you might want to refer to the following external resources:

- OpenVIM Usage Guide
- OpenVIM Northbound API (PDF)
- OpenVIM Install Guide
- OpenVIM Compute Node Configuration

Configure Cloud Account

Create cloud account

cURL

```
curl -i -X POST https://<orchestrator_ip/fqdn>:8008/api/config/cloud/account/
Grunt192.66.4.17 -u admin:admin --insecure -H "Accept: application/json" -d
@/tmp/cloud.txt
```

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/cloud/account/{name}
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

OpenStack data

The auth_url refers to your OpenStack system. The Orchestrator must be able to reach this URL.

Supported versions of the Openstack Nova API are v2, v2.1, v3.

```
{
       "account":[
          {
             "name":"Grunt192.66.4.17",
             "account-type":"openstack",
             "openstack":{
                "admin":"false",
                "mgmt-network":"private",
                "dynamic-flavor-support":"true",
                "plugin-name":"rwcal_openstack",
                "key":"",
                "tenant":"demo",
                "cert-validate":"false",
                "auth_url":"http://<keystone_url>/<version>",
                "secret":""
             }
          }
       ]
 }
```

OpenVIM data

"host" is the IP address of the host on which OpenVIM is running.

```
{
       "account":[
          {
             "name":"Grunt192.66.4.36",
             "account-type":"openvim",
             "openvim":{
                "host":"192.66.4.36",
                "mgmt-network":"default",
                "tenant-name":"demo",
               }
          }
       ]
 }
```

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Delete cloud account

cURL

```
curl -i -X DELETE
https://<orchestrator_ip/fqdn>:8008/api/config/cloud/account/Grunt192.66.4.17
-u admin:admin --insecure -H "Accept: application/json"
```

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/cloud/account/{name}
```

Method

DELETE

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

None

Response

On error: HTTP 405 with an error object

On success: HTTP 200 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Upload Image

Generate the MD5 checksum of the QCOW2 image and upload it to the cloud account from the "Configure Cloud Account" on page 347 example.

cURL

```
curl -X POST -H "Content-Type: application/vnd.yang.data+json" -H "Accept:
application/vnd.yang.data+json" -u admin:admin -d @/tmp/rpc2.txt --insecure
https:<orchestrator_ip/fqdn>:8008/api/operations/create-upload-job
```

URL

```
https://<orchestrator_ip/fqdn>:8008/api/operations/create-upload-job
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

```
{
   "input": {
    "cloud-account": "Grunt192.66.4.36",
     "external-url": {
         "image-url": "http://www.example.com/XN65E06D499AA624.qcow2",
         "disk-format": "qcow2",
         "container-format": "bare",
         "image-name": "XN65E06D499AA624.qcow2",
         "image-checksum": "c9dd9dad03d944858c5228ed492398f3"
     }
   }
 }
```

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Configure a VNF Descriptor

Create VNFD

cURL

```
curl -i -X POST https://<orchestrator_ip/fqdn>:8008/api/config/vnfd-
catalog/vnfd -u admin:admin --insecure -H "Accept:application/json" -d
@/tmp/vnfd.txt
```

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/vnfd-catalog/vnfd
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

```
{
       "vnfd": [
         {
           "vdu": [
             {
               "image": "XN65E06D499AA624.qcow2",
               "vm-flavor": {
                 "vcpu-count": 2,
                 "storage-gb": 32,
                 "memory-mb": 4096
               },
               "id": "iovdu_0",
               "count": 1,
               "name": "iovdu_0"
             }
           ],
           "mgmt-interface": {
             "vdu-id": "iovdu_0"
           },
           "version": "1.0",
           "vendor": "",
           "id": "XN65E06D499AA624",
           "short-name": "XN65E06D499AA624",
           "name": "XN65E06D499AA624",
           "description": " "
         }
       ]
 }
```

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Delete VNFD

cURL

```
curl -i -X DELETE https://<orchestrator_ip/fqdn>:8008/api/config/vnfd-
catalog/vnfd/XN65E06D499AA624 -u admin:admin  --insecure -H "Accept:
application/json"
```

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/vnfd-catalog/vnfd/{id}
```

Method

DELETE

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

None

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Configure a Network Service Descriptor

Create NSD

cURL

```
curl -i -X POST https://<orchestrator_ip/fqdn>:8008/api/config/nsd-
catalog/nsd -u admin:admin --insecure -H "Accept: application/json" -d
@/tmp/nsd.txt
```

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/nsd-catalog/nsd
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

```
{
      "nsd": [
         {
          "constituent-vnfd":
             [{
                "member-vnf-index": 1,
                "vnfd-id-ref": "XN65E06D499AA624"
              }],
          "vendor": "",
          "version": "1.0",
          "description": " ",
          "short-name": "XN65E06D499AA624_nsd",
          "name": "XN65E06D499AA624_nsd",
          "id": "XN65E06D499AA624_nsd"
          }
       ]
 }
```

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Delete NSD

cURL

```
curl -i -X DELETE https://<orchestrator_ip/fqdn>:8008/api/config/nsd-
catalog/nsd/XN65E06D499AA624_nsd -u admin:admin  --insecure -H "Accept:
application/json"
```

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/nsd-catalog/nsd/{id}
```

Method

DELETE

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

None

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Instantiate the Network Service

Create NSR

This API instantiates a network service.

cURL

```
curl --insecure --request POST
https://<orchestrator_ip/fqdn>:8008/api/config/ns-instance-config/nsr -u
admin:admin -H "Accept:application/vnd.yang.data+json" -H "Content-
type:application/vnd.yang.data+json" -d @/tmp/nsr.txt
```

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/ns-instance-config/nsr
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

```
{
   "nsr": [
     {
       "id": "90b024f4-79a3-4acb-818a-c93728220b3d",
       "name": "TestXN65E06D499AA624",
       "short-name": "Test",
       "description": " ",
       "admin-status": "ENABLED",
       "cloud-account": "Grunt192.66.4.36",
       "nsd": {
           "id": "XN65E06D499AA624.qcow2_nsd",
           "name": "XN65E06D499AA624.qcow2_nsd",
           "short-name": "XN65E06D499AA624.qcow2_nsd",
           "description": " ",
           "constituent-vnfd": [
              {
                 "vnfd-id-ref": "XN65E06D499AA624.qcow2",
                 "member-vnf-index": 1
              }
           ]
        }
     }
   ]
 }
```

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Delete NSR

To obtain an NSR ID, GET the network service records, identify the record to remove, and then run DELETE.

GET URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/ns-instance-config/nsr
```

DELETE cURL

```
curl -i -X DELETE https://<orchestrator_ip/fqdn>:8008/api/config/nsd-
catalog/nsr/XN65E06D499AA624_nsd -u admin:admin  --insecure -H "Accept:
application/json"
```

DELETE URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/ns-instance-config/nsr/{id}
```

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

None

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Manage the Network Service using Ping-Pong

The examples in this topic use the Ping-Pong sample network service. Download the Ping-Pong package from the RIFT.io website at https://open.riftio.com/download/.

Associating a VIM account with your Orchestrator instance

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/cloud
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

```
{
   "account": [
     {
       "name": "MyOpenstackAccount",
       "account-type": "openstack",
       "params": [
         {
           "label": "Key",
           "ref": "key"
         },
         {
           "label": "Secret",
           "ref": "secret"
         },
         {
           "label": "Authentication URL",
           "ref": "auth_url"
         },
         {
           "label": "Tenant",
           "ref": "tenant"
         },
         {
           "label": "Management Network",
           "ref": "mgmt-network"
         },
         {
           "label": "Floating IP Pool",
```

```
            "ref": "floating-ip-pool",
            "optional": true
          }
        ],
        "openstack": {
          "key": "key",
          "secret": "secret",
          "auth_url": "http://192.66.4.18:5000/v3",
          "tenant": "demo",
          "mgmt-network": "private"
        }
      }
    ]
 }
```

Note: Change the auth_url to your OpenStack system. The Orchestrator must be able to reach this URL. Supported versions of the Openstack Nova API are v2, v2.1, v3.

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Onboarding preconfigured VNF descriptors

Part 1: Onboard the ping VNFD

This example shows how to onboard the preconfigured `ping` VNFD using the Ping-Pong network service package.

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/vnfd-catalog/vnfd
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

```
{
     "vnfd": [
     {
         "description": "This is an example  VNF",
         "name": "ping_vnfd",
         "vendor": "",
         "id": "7faf9e0e-f196-11e5-a1e9-6cb3113b406f",
         "vdu": [
           {
             "count": 1,
             "name": "iovdu_0",
             "guest-epa": {
               "cpu-pinning-policy": "ANY"
             },
             "image-checksum": "a6ffaa77f949a9e4ebb082c6147187cf",
             "external-interface": [
               {
                 "name": "eth0",
                 "virtual-interface": {
                   "type": "VIRTIO"
                 },
                 "vnfd-connection-point-ref": "ping_vnfd/cp0"
               }
             ],
             "cloud-init": "#cloud-config\npassword: fedora\nchpasswd: {
expire: False }\nssh_pwauth: True\nruncmd:\n  - [ systemctl, daemon-reload
]\n  - [ systemctl, enable, ping.service ]\n  - [ systemctl, start, --no-
block, ping.service ]\n  - [ ifup, eth1 ]\n",
             "id": "7fb0223e-f196-11e5-a1e9-6cb3113b406f",
             "vm-flavor": {
               "memory-mb": 512,
               "vcpu-count": 1,
               "storage-gb": 4
             },
             "image": "Fedora-x86_64-20-20131211.1-sda-ping.qcow2"
           }
         ],
         "mgmt-interface": {
           "vdu-id": "7fb0223e-f196-11e5-a1e9-6cb3113b406f",
           "port": 18888,
```

```
          "dashboard-params": {
            "path": "/api/v1/pong/stats"
          }
        },
        "connection-point": [
          {
            "name": "ping_vnfd/cp0",
            "type": "VPORT"
          }
        ],
        "short-name": "ping_vnfd",
        "service-function-chain": "UNAWARE",
        "version": 1,
        "monitoring-param": [
          {
            "description": "number of ping requests",
            "value-type": "INT",
            "group-tag": "Group1",
            "units": "packets",
            "widget-type": "COUNTER",
            "name": "ping-request-tx-count",
            "json-query-method": "NAMEKEY",
            "id": 1,
            "http-endpoint-ref": "api/v1/ping/stats"
          },
          {
            "description": "number of ping responses",
            "value-type": "INT",
            "group-tag": "Group1",
            "units": "packets",
            "widget-type": "COUNTER",
            "name": "ping-response-rx-count",
            "json-query-method": "NAMEKEY",
            "id": 2,
            "http-endpoint-ref": "api/v1/ping/stats"
          }
        ],
        "http-endpoint": [
          {
            "path": "api/v1/ping/stats",
            "https": "false",
            "method": "GET",
            "port": 18888,
            "polling_interval_secs": 2
          }
        ]
      }]
}
```

Note: The image listed in the VNFD (Fedora-x86_64-20-20131211.1-sda-ping.qcow2 in the above example) must already be uploaded to your controller before you try to initiate the service.

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Part 2: Onboard the pong VNFD

This example shows how to onboard the preconfigured `pong` VNFD using the Ping-Pong network service package.

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/vnfd-catalog/vnfd
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

```
{
     "vnfd": [
     {
         "description": "This is an example  VNF",
         "name": "pong_vnfd",
         "vendor": "",
         "id": "7fb05722-f196-11e5-a1e9-6cb3113b406f",
         "vdu": [
           {
             "count": 1,
             "name": "iovdu_0",
             "guest-epa": {
               "cpu-pinning-policy": "ANY"
             },
             "image-checksum": "977484d95575f80ef8399c9cf1d45ebd",
             "external-interface": [
               {
```

```
                "name": "eth0",
                "virtual-interface": {
                  "type": "VIRTIO"
                },
                "vnfd-connection-point-ref": "pong_vnfd/cp0"
              }
            ],
            "cloud-init": "#cloud-config\npassword: fedora\nchpasswd: {
expire: False }\nssh_pwauth: True\nruncmd:\n  - [ systemctl, daemon-reload
]\n  - [ systemctl, enable, pong.service ]\n  - [ systemctl, start, --no-
block, pong.service ]\n  - [ ifup, eth1 ]\n",
            "id": "7fb0842c-f196-11e5-a1e9-6cb3113b406f",
            "vm-flavor": {
              "memory-mb": 512,
              "vcpu-count": 1,
              "storage-gb": 4
            },
            "image": "Fedora-x86_64-20-20131211.1-sda-pong.qcow2"
          }
        ],
        "mgmt-interface": {
          "vdu-id": "7fb0842c-f196-11e5-a1e9-6cb3113b406f",
          "port": 18889,
          "dashboard-params": {
            "path": "/api/v1/pong/stats"
          }
        },
        "connection-point": [
          {
            "name": "pong_vnfd/cp0",
            "type": "VPORT"
          }
        ],
        "short-name": "pong_vnfd",
        "service-function-chain": "UNAWARE",
        "version": 1,
        "monitoring-param": [
          {
            "description": "number of pong requests",
            "value-type": "INT",
            "group-tag": "Group1",
            "units": "packets",
            "widget-type": "COUNTER",
            "name": "ping-request-rx-count",
            "json-query-method": "NAMEKEY",
            "id": 1,
            "http-endpoint-ref": "api/v1/pong/stats"
          },
          {
            "description": "number of pong responses",
            "value-type": "INT",
            "group-tag": "Group1",
            "units": "packets",
            "widget-type": "COUNTER",
            "name": "ping-response-tx-count",
            "json-query-method": "NAMEKEY",
```

```
              "id": 2,
              "http-endpoint-ref": "api/v1/pong/stats"
            }
          ],
          "http-endpoint": [
            {
              "path": "api/v1/pong/stats",
              "https": "false",
              "method": "GET",
              "port": 18889,
              "polling_interval_secs": 2
            }
          ]
        }]
 }
```

Note: The image listed in the VNFD (Fedora-x86_64-20-20131211.1-sda-pong.qcow2 in the above example) must already be uploaded to your controller before you try to initiate the service

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Onboarding the preconfigured network service descriptor

This example shows how to onboard the Ping-Pong network service package, which has connectivity between the VNFs.

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/nsd-catalog/nsd
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
```

```
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

```
{
     "nsd": [{
         "description": "Toy NS",
         "input-parameter-xpath": [
           {
             "xpath": "/nsd:nsd-catalog/nsd:nsd/nsd:vendor"
           }
         ],
         "short-name": "ping_pong_nsd",
         "constituent-vnfd": [
           {
             "member-vnf-index": 1,
             "vnf-configuration": {
               "config-template": "\n#!/usr/bin/bash\n\n#
                 Rest API
config\nping_mgmt_ip='&lt;rw_mgmt_ip&gt;'\43nping_mgmt_port=18888\n\n# VNF
specific configuration\npong_server_ip='&lt;rw_connection_point_name
pong_vnfd/cp0&gt;'\nping_rate=5\nserver_port=5555\n\n# Make rest API calls to
configure VNF\ncurl -D /dev/stdout \\\n    -H \"Accept:
application/vnd.yang.data+xml\" \\\n    -H \"Content-Type:
application/vnd.yang.data+json\" \\\n    -X POST \\\n    -d
\"{\"ip\":\"$pong_server_ip\", \"port\":$server_port}\" \\\n
http://${ping_mgmt_ip}:${ping_mgmt_port}/api/v1/ping/server\nrc=$?\nif [ $rc
-ne 0 ]\nthen\n    echo \"Failed to set server info for ping!\"\n    exit
$rc\nfi\n\ncurl -D /dev/stdout \\\n    -H \"Accept:
application/vnd.yang.data+xml\" \\\n    -H \"Content-Type:
application/vnd.yang.data+json\" \\\n    -X POST \\\n    -d
\"{\"rate\":$ping_rate}\" \\\n
http://${ping_mgmt_ip}:${ping_mgmt_port}/api/v1/ping/rate\nrc=$?\nif [ $rc -
ne 0 ]\nthen\n    echo \"Failed to set ping rate!\"\n    exit
$rc\nfi\n\noutput=$(curl -D /dev/stdout \\\n    -H \"Accept:
application/vnd.yang.data+xml\" \\\n    -H \"Content-Type:
application/vnd.yang.data+json\" \\\n    -X POST \\\n    -d
\"{\"enable\":true}\" \\\n
http://${ping_mgmt_ip}:${ping_mgmt_port}/api/v1/ping/adminstatus/state)\nif
[[ $output == *\"Internal Server Error\"* ]]\nthen\n    echo $output\n
exit 3\nelse\n    echo $output\nfi\n\n\nexit 0\n        ",
               "script": {
                 "script-type": "bash"
               },
               "config-type": "script",
               "input-params": {
                 "config-priority": 2,
                 "config-delay": 0
               }
             },
             "vnfd-id-ref": "7faf9e0e-f196-11e5-a1e9-6cb3113b406f"
           },
           {
             "member-vnf-index": 2,
             "vnf-configuration": {
```

```
               "config-template": "\n#!/usr/bin/bash\n\n# Rest API
configuration\npong_mgmt_ip='&lt;rw_mgmt_ip&gt;'\npong_mgmt_port=18889\n#
username=&lt;rw_username&gt;\n# password=&lt;rw_password&gt;\n\n# VNF
specific configuration\npong_server_ip='&lt;rw_connection_point_name
pong_vnfd/cp0&gt;'\nserver_port=5555\n\n# Make Rest API calls to configure
VNF\ncurl -D /dev/stdout \\\n    -H \"Accept: application/vnd.yang.data+xml\"
\\\n    -H \"Content-Type: application/vnd.yang.data+json\" \\\n    -X POST
\\\n    -d \"{\"ip\":\"$pong_server_ip\", \"port\":$server_port}\" \\\n
http://${pong_mgmt_ip}:${pong_mgmt_port}/api/v1/pong/server\nrc=$?\nif [ $rc
-ne 0 ]\nthen\n    echo \"Failed to set server(own) info for pong!\"\n
exit $rc\nfi\n\ncurl -D /dev/stdout \\\n    -H \"Accept:
application/vnd.yang.data+xml\" \\\n    -H \"Content-Type:
application/vnd.yang.data+json\" \\\n    -X POST \\\n    -d
\"{\"enable\":true}\" \\\n
http://${pong_mgmt_ip}:${pong_mgmt_port}/api/v1/pong/adminstatus/state\nrc=$?
\nif [ $rc -ne 0 ]\nthen\n    echo \"Failed to enable pong service!\"\n
exit $rc\nfi\n\nexit 0\n        ",
               "script": {
                 "script-type": "bash"
               },
               "config-type": "script",
               "input-params": {
                 "config-priority": 1,
                 "config-delay": 60
               }
             },
             "vnfd-id-ref": "7fb05722-f196-11e5-a1e9-6cb3113b406f"
           }
         ],
         "config-primitive": [
           {
             "name": "ping config",
             "user-defined-script": "ping_config.py"
           }
         ],
         "name": "ping_pong_nsd",
         "vendor": "",
         "id": "7fb09dea-f196-11e5-a1e9-6cb3113b406f",
         "version": 1,
         "vld": [
           {
             "description": "Toy VL",
             "vnfd-connection-point-ref": [
               {
                 "vnfd-connection-point-ref": "ping_vnfd/cp0",
                 "member-vnf-index-ref": 1,
                 "vnfd-id-ref": "7faf9e0e-f196-11e5-a1e9-6cb3113b406f"
               },
               {
                 "vnfd-connection-point-ref": "pong_vnfd/cp0",
                 "member-vnf-index-ref": 2,
                 "vnfd-id-ref": "7fb05722-f196-11e5-a1e9-6cb3113b406f"
               }
             ],
             "short-name": "ping_pong_vld",
             "name": "ping_pong_vld",
```

```
            "vendor": "",
            "id": "7fb0cd42-f196-11e5-a1e9-6cb3113b406f",
            "version": 1,
            "type": "ELAN"
          }
        ]
      }]
 }
```

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Instantiating the network service

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/ns-instance-config/nsr
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

```
{
   "nsr": [
     {
       "id": "90b024f4-79a3-4acb-818a-c93728220b3d",
       "nsd-ref": "7fb09dea-f196-11e5-a1e9-6cb3113b406f",
       "name": "TestPingPongNS",
       "short-name": "TestPingPongNS",
       "description": "a description for 90b024f4-79a3-4acb-818a-
c93728220b3d",
       "admin-status": "ENABLED",
       "cloud-account": "MyOpenstackAccount"
     }
```

```
    ]
 }
```

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Monitoring the instantiated network service

URL

```
https://<orchestrator_ip/fqdn>:8008/api/operational/ns-instance-
opdata/nsr/<nsr_id>?deep
```

Method

GET

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

None

Response

On error: HTTP 405 with an error object

On success: HTTP 200 with payload similar to the following.

```
{
   "nsr:nsr": {
     "config-status": "configuring",
     "create-time": 1458833812,
     "vlr": [
       {
         "vlr-ref": "0aa96ce8-ceb4-43f4-83f7-43aa52d7f3fd",
         "vnfr-connection-point-ref": [
           {
```

```
            "connection-point": "ping_vnfd/cp0",
            "vnfr-id": "759a43dc-4596-4628-817d-f9c882470cf5"
          },
          {
            "connection-point": "pong_vnfd/cp0",
            "vnfr-id": "c6196a63-e175-43b2-9bf6-74536c3bc3ed"
          }
        ]
      }
    ],
    "rw-nsr:nfvi-metrics": {
      "internal-ports": {
        "label": "INTERNAL PORTS",
        "total": 0
      },
      "storage": {
        "utilization": 0,
        "used": 0,
        "label": "STORAGE",
        "total": 8000000000
      },
      "vcpu": {
        "utilization": 0,
        "label": "VCPU",
        "total": 2
      },
      "network": {
        "incoming": {
          "bytes": 0,
          "packets": 0,
          "label": "INCOMING NETWORK TRAFFIC",
          "packet-rate": 0,
          "byte-rate": 0
        },
        "outgoing": {
          "bytes": 0,
          "packets": 0,
          "label": "OUTGOING NETWORK TRAFFIC",
          "packet-rate": 0,
          "byte-rate": 0
        },
        "label": "NETWORK TRAFFIC"
      },
      "external-ports": {
        "label": "EXTERNAL PORTS",
        "total": 2
      },
      "vm": {
        "inactive-vm": 0,
        "label": "VM",
        "active-vm": 2
      },
      "memory": {
        "utilization": 0,
        "used": 0,
        "label": "MEMORY",
```

```
        "total": 1024000000
      }
    },
    "scaling-group-record": [
      {
        "scaling-group-name-ref": "ping_group"
      }
    ],
    "vnf-monitoring-param": [
      {
        "vnfr-id-ref": "c6196a63-e175-43b2-9bf6-74536c3bc3ed",
        "monitoring-param": [
          {
            "description": "no of ping responses",
            "value-type": "INT",
            "group-tag": "Group1",
            "units": "packets",
            "widget-type": "COUNTER",
            "value-integer": 0,
            "name": "ping-response-tx-count",
            "json-query-method": "NAMEKEY",
            "id": 2,
            "http-endpoint-ref": "api/v1/pong/stats"
          },
          {
            "description": "no of ping requests",
            "value-type": "INT",
            "group-tag": "Group1",
            "units": "packets",
            "widget-type": "COUNTER",
            "value-integer": 0,
            "name": "ping-request-rx-count",
            "json-query-method": "NAMEKEY",
            "id": 1,
            "http-endpoint-ref": "api/v1/pong/stats"
          }
        ]
      },
      {
        "vnfr-id-ref": "759a43dc-4596-4628-817d-f9c882470cf5",
        "monitoring-param": [
          {
            "description": "no of ping responses",
            "value-type": "INT",
            "group-tag": "Group1",
            "units": "packets",
            "widget-type": "COUNTER",
            "value-integer": 0,
            "name": "ping-response-rx-count",
            "json-query-method": "NAMEKEY",
            "id": 2,
            "http-endpoint-ref": "api/v1/ping/stats"
          },
          {
            "description": "no of ping requests",
            "value-type": "INT",
```

```
            "group-tag": "Group1",
            "units": "packets",
            "widget-type": "COUNTER",
            "value-integer": 0,
            "name": "ping-request-tx-count",
            "json-query-method": "NAMEKEY",
            "id": 1,
            "http-endpoint-ref": "api/v1/ping/stats"
          }
        ]
      }
    ],
    "operational-status": "running",
    "cloud-account": "MyOpenstackAccount",
    "nsd-name-ref": "ping_pong_nsd",
    "rw-nsr:operational-events": [
      {
        "id": 1,
        "description": "Instatiation Request Received NSR Id:90b024f4-79a3-
4acb-818a-c93728220b3d",
        "event": "instantiating",
        "timestamp": 1458833812
      },
      {
        "id": 2,
        "description": "Fetched NSD with descriptor id 7fb09dea-f196-11e5-
a1e9-6cb3113b406f",
        "event": "nsd-fetched",
        "timestamp": 1458833812
      },
      {
        "id": 3,
        "description": "Instantiating 1 external VLs for NSR id 90b024f4-
79a3-4acb-818a-c93728220b3d",
        "event": "begin-external-vls-instantiation",
        "timestamp": 1458833812
      },
      {
        "id": 4,
        "description": "Finished instantiating 1 external VLs for NSR id
90b024f4-79a3-4acb-818a-c93728220b3d",
        "event": "end-external-vls-instantiation",
        "timestamp": 1458833815
      },
      {
        "id": 5,
        "description": "Instantiating 2 VNFS for NSR id 90b024f4-79a3-4acb-
818a-c93728220b3d",
        "event": "begin-vnf-instantiation",
        "timestamp": 1458833815
      },
      {
        "id": 6,
        "description": "Finished instantiating 2 VNFs for NSR id 90b024f4-
79a3-4acb-818a-c93728220b3d",
        "event": "end-vnf-instantiation",
```

```
        "timestamp": 1458833815
      },
      {
        "id": 7,
        "description": "NSR in running state for NSR id 90b024f4-79a3-4acb-
818a-c93728220b3d",
        "event": "ns-running",
        "timestamp": 1458833815
      }
    ],
    "constituent-vnfr-ref": [
      {
        "vnfr-id": "c6196a63-e175-43b2-9bf6-74536c3bc3ed"
      },
      {
        "vnfr-id": "759a43dc-4596-4628-817d-f9c882470cf5"
      }
    ],
    "ns-instance-config-ref": "90b024f4-79a3-4acb-818a-c93728220b3d",
    "name-ref": "TestPingPongNS"
  }
 }
```

Monitoring underlying VNFs

URL

```
https://<orchestrator_ip/fqdn>:8008/api/operational/vnfr-catalog/vnfr?deep
```

Method

GET

HTTP headers

```
{
   "Accept": "application/vnd.yang.collection+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

None

Response

On error: HTTP 405 with an error object

On success: HTTP 200 with payload-like:

```
{
   "collection": {
```

```
    "vnfr:vnfr": [
      {
        "description": "This is an example  VNF",
        "vnf-configuration": {
          "config-access": {
            "mgmt-ip-address": "10.66.218.250"
          },
          "config-template": "\n#!/usr/bin/bash\n\n# Rest API
configuration\npong_mgmt_ip='<rw_mgmt_ip>'\npong_mgmt_port=18889\n#
username=<rw_username>\n# password=<rw_password>\n\n# VNF specific
configuration\npong_server_ip='<rw_connection_point_name
pong_vnfd/cp0>'\nserver_port=5555\n\n# Make Rest API calls to configure
VNF\ncurl -D /dev/stdout \\\n    -H \"Accept: application/vnd.yang.data+xml\"
\\\n    -H \"Content-Type: application/vnd.yang.data+json\" \\\n    -X POST
\\\n    -d \"{\"ip\":\"$pong_server_ip\", \"port\":$server_port}\" \\\n
http://${pong_mgmt_ip}:${pong_mgmt_port}/api/v1/pong/server\nrc=$?\nif [ $rc
-ne 0 ]\nthen\n    echo \"Failed to set server(own) info for pong!\"\n
exit $rc\nfi\n\ncurl -D /dev/stdout \\\n    -H \"Accept:
application/vnd.yang.data+xml\" \\\n    -H \"Content-Type:
application/vnd.yang.data+json\" \\\n    -X POST \\\n    -d
\"{\"enable\":true}\" \\\n
http://${pong_mgmt_ip}:${pong_mgmt_port}/api/v1/pong/adminstatus/state\nrc=$?
\nif [ $rc -ne 0 ]\nthen\n    echo \"Failed to enable pong service!\"\n
exit $rc\nfi\n\nexit 0\n        ",
          "script": {
            "script-type": "bash"
          },
          "config-type": "script",
          "input-params": {
            "config-priority": 1,
            "config-delay": 60
          }
        },
        "config-status": "configuring",
        "connection-point": [
          {
            "name": "pong_vnfd/cp0",
            "ip-address": "11.0.0.2",
            "vlr-ref": "0aa96ce8-ceb4-43f4-83f7-43aa52d7f3fd",
            "connection-point-id": "3c6ffdad-d450-4f3a-9a7d-9499cec9bb9e"
          }
        ],
        "vendor": "",
        "id": "c6196a63-e175-43b2-9bf6-74536c3bc3ed",
        "mgmt-interface": {
          "ip-address": "10.66.218.250",
          "port": 18889
        },
        "name": "TestPingPongNS._.pong_vnfd.2",
        "vnfd-ref": "7fb05722-f196-11e5-a1e9-6cb3113b406f",
        "vdur": [
          {
            "vm-management-ip": "10.0.218.233",
            "guest-epa": {
              "cpu-pinning-policy": "ANY"
            },
```

```
            "management-ip": "10.66.218.250",
            "vdu-id-ref": "7fb0842c-f196-11e5-a1e9-6cb3113b406f",
            "external-interface": [
              {
                "name": "pong_vnfd/cp0",
                "vnfd-connection-point-ref": "pong_vnfd/cp0"
              }
            ],
            "id": "a186ec29-598d-4eef-baa4-70fb69baf490",
            "flavor-id": "500d30e6-41ce-40d4-ad7d-fe14b113e6bf",
            "vim-id": "2f51ce1a-0c3a-43ea-878b-21c8efce8217",
            "vm-flavor": {
              "memory-mb": 512,
              "vcpu-count": 1,
              "storage-gb": 4
            },
            "operational-status": "running",
            "image-id": "d96eda6d-4a48-4f9d-bd8c-3a5a26a64b71"
          }
        ],
        "short-name": "pong_vnfd",
        "operational-status": "running",
        "cloud-account": "MyOpenstackAccount",
        "dashboard-url": "http://10.66.218.250:80/api/v1/pong/stats",
        "version": 1,
        "member-vnf-index-ref": 2,
        "monitoring-param": [
          {
            "description": "no of ping requests",
            "value-type": "INT",
            "group-tag": "Group1",
            "units": "packets",
            "widget-type": "COUNTER",
            "value-integer": 0,
            "name": "ping-request-rx-count",
            "json-query-method": "NAMEKEY",
            "id": 1,
            "http-endpoint-ref": "api/v1/pong/stats"
          },
          {
            "description": "no of ping responses",
            "value-type": "INT",
            "group-tag": "Group1",
            "units": "packets",
            "widget-type": "COUNTER",
            "value-integer": 0,
            "name": "ping-response-tx-count",
            "json-query-method": "NAMEKEY",
            "id": 2,
            "http-endpoint-ref": "api/v1/pong/stats"
          }
        ]
      },
      {
        "description": "This is an example  VNF",
        "vnf-configuration": {
```

```
           "config-access": {
             "mgmt-ip-address": "10.66.218.251"
           },
           "config-template": "\n#!/usr/bin/bash\n\n# Rest API
config\nping_mgmt_ip='<rw_mgmt_ip>'\nping_mgmt_port=18888\n\n# VNF specific
configuration\npong_server_ip='<rw_connection_point_name
pong_vnfd/cp0>'\nping_rate=5\nserver_port=5555\n\n# Make rest API calls to
configure VNF\ncurl -D /dev/stdout \\\n    -H \"Accept:
application/vnd.yang.data+xml\" \\\n    -H \"Content-Type:
application/vnd.yang.data+json\" \\\n    -X POST \\\n    -d
\"{\"ip\":\"$pong_server_ip\", \"port\":$server_port}\" \\\n
http://${ping_mgmt_ip}:${ping_mgmt_port}/api/v1/ping/server\nrc=$?\nif [ $rc
-ne 0 ]\nthen\n    echo \"Failed to set server info for ping!\"\n    exit
$rc\nfi\n\ncurl -D /dev/stdout \\\n    -H \"Accept:
application/vnd.yang.data+xml\" \\\n    -H \"Content-Type:
application/vnd.yang.data+json\" \\\n    -X POST \\\n    -d
\"{\"rate\":$ping_rate}\" \\\n
http://${ping_mgmt_ip}:${ping_mgmt_port}/api/v1/ping/rate\nrc=$?\nif [ $rc -
ne 0 ]\nthen\n    echo \"Failed to set ping rate!\"\n    exit
$rc\nfi\n\noutput=$(curl -D /dev/stdout \\\n    -H \"Accept:
application/vnd.yang.data+xml\" \\\n    -H \"Content-Type:
application/vnd.yang.data+json\" \\\n    -X POST \\\n    -d
\"{\"enable\":true}\" \\\n
http://${ping_mgmt_ip}:${ping_mgmt_port}/api/v1/ping/adminstatus/state)\nif
[[ $output == *\"Internal Server Error\"* ]]\nthen\n    echo $output\n
exit 3\nelse\n    echo $output\nfi\n\n\nexit 0\n        ",
           "script": {
             "script-type": "bash"
           },
           "config-type": "script",
           "input-params": {
             "config-priority": 2,
             "config-delay": 0
           }
         },
         "config-status": "configuring",
         "connection-point": [
           {
             "name": "ping_vnfd/cp0",
             "ip-address": "11.0.0.3",
             "vlr-ref": "0aa96ce8-ceb4-43f4-83f7-43aa52d7f3fd",
             "connection-point-id": "114b5015-6006-4cd0-8e29-683fd886ae1a"
           }
         ],
         "vendor": "",
         "id": "759a43dc-4596-4628-817d-f9c882470cf5",
         "mgmt-interface": {
           "ip-address": "10.66.218.251",
           "port": 18888
         },
         "name": "TestPingPongNS._.ping_vnfd.1",
         "vnfd-ref": "7faf9e0e-f196-11e5-a1e9-6cb3113b406f",
         "vdur": [
           {
             "vm-management-ip": "10.0.218.236",
             "guest-epa": {
```

```
            "cpu-pinning-policy": "ANY"
          },
          "management-ip": "10.66.218.251",
          "vdu-id-ref": "7fb0223e-f196-11e5-a1e9-6cb3113b406f",
          "external-interface": [
            {
              "name": "ping_vnfd/cp0",
              "vnfd-connection-point-ref": "ping_vnfd/cp0"
            }
          ],
          "id": "a6e5c57d-890c-469f-82d7-5808ed523c31",
          "flavor-id": "500d30e6-41ce-40d4-ad7d-fe14b113e6bf",
          "vim-id": "aaf88b0e-8577-4b09-b8c0-c3316010e36c",
          "vm-flavor": {
            "memory-mb": 512,
            "vcpu-count": 1,
            "storage-gb": 4
          },
          "operational-status": "running",
          "image-id": "1b38b7d1-1146-424e-a831-eb765bc89a07"
        }
      ],
      "short-name": "ping_vnfd",
      "operational-status": "running",
      "cloud-account": "MyOpenstackAccount",
      "dashboard-url": "http://10.66.218.251:80/api/v1/pong/stats",
      "version": 1,
      "member-vnf-index-ref": 1,
      "monitoring-param": [
        {
          "description": "no of ping requests",
          "value-type": "INT",
          "group-tag": "Group1",
          "units": "packets",
          "widget-type": "COUNTER",
          "value-integer": 0,
          "name": "ping-request-tx-count",
          "json-query-method": "NAMEKEY",
          "id": 1,
          "http-endpoint-ref": "api/v1/ping/stats"
        },
        {
          "description": "no of ping responses",
          "value-type": "INT",
          "group-tag": "Group1",
          "units": "packets",
          "widget-type": "COUNTER",
          "value-integer": 0,
          "name": "ping-response-rx-count",
          "json-query-method": "NAMEKEY",
          "id": 2,
          "http-endpoint-ref": "api/v1/ping/stats"
        }
      ]
    }
  ]
```

```
    }
 }
```

Terminating the Ping-Pong NS

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/ns-instance-
config/nsr/{nsr_id}
```

Method

DELETE

HTTP request headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

None

HTTP response headers

On error: HTTP 405 with an error object

On success: HTTP 200 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Trigger Config Primitives on Network Service

If your network services have configuration primitives defined in the descriptor, you can trigger those primitives with the **RPC** API described in this topic.

Note: Primitives can be triggered only after the initial configuration has completed. Check the status by looking at the `"config-status"` parameter. See the response payload for "Monitoring the instantiated network service" in "Manage the Network Service using Ping-Pong" on page 357.

URL

```
https://<orchestrator_ip/fqdn>:8008/api/operations/exec-ns-config-primitive
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   "Authorization": "Basic YWRtaW46YWRtaW4="
 }
```

Payload

Example to update IMS network service configuration through configuration primitives:

```
{
   "input": {
     "name": "Update Domain",
     "nsr_id_ref": "07497feb-4f01-411b-a0da-bc258ce6a30b",
     "vnf-list": [
       {
         "member_vnf_index_ref": "1",
         "vnfr-id-ref": "e8c65cc4-5e81-40b2-b2b0-87092defeb3d",
         "vnf-primitive": [
           {
             "name": "config",
             "index": "0",
             "parameter": [
               {
                 "name": "base_number",
                 "value": "1234567890"
               },
               {
                 "name": "home_domain",
                 "value": "abc.com"
```

```
                },
                {
                  "name": "number_count",
                  "value": "1000"
                },
                {
                  "name": "password",
                  "value": "cw-aio"
                }
              ]
            }
          ]
        }
      ]
    }
  }
```

Response

On error: HTTP 405 with an error object

On success: HTTP 200 with the following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Manage Scaling Group Instance

This topic describes how to use API calls to create and delete a scaling instance for a VNF. For more information, see Scaling Group Descriptor and NFV Scaling.

Creating a scaling group instance

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/ns-instance-config/nsr/<nsr-id>/scaling-group/<scaling_group_name>/instance HTTP/1.1
```

Method

POST

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   “Connection: keep-alive”,
   “Cache-Control: no-cache”
   "Authorization": "Basic YWRtaW46YWRtaW4=“,
 }
```

Payload

```
{
     "instance": [
         {
             "index": <instance_index_number>
         }
     ]
 }
```

Response

On error: HTTP 405 with an error object

On success: HTTP 201 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```

Deleting a scaled instance

URL

```
https://<orchestrator_ip/fqdn>:8008/api/config/ns-instance-config/nsr/<nsr-
id>/scaling-group/<scaling_group_name>/instance/<instance_index_number>
HTTP/1.1
```

Method

DELETE

HTTP headers

```
{
   "Accept": "application/vnd.yang.data+json",
   "Content-Type": "application/vnd.yang.data+json",
   “Connection: keep-alive”,
   “Cache-Control: no-cache”
   "Authorization": "Basic YWRtaW46YWRtaW4=“,
 }
```

Payload

None

Response

On error: HTTP 405 with an error object

On success: HTTP 200 with following payload:

```
{
   "rpc-reply": {
     "ok": ""
   }
 }
```