# Oracle® Communications

## Diameter Signaling Router

Virtual Network Functions Manager Installation and User Guide

Release 8.3

**E93569-02**

December 2018

Oracle Communications Diameter Signaling Router VNFM Installation and User Guide, Release 8.3.

⚠ **CAUTION**:  Use only the Upgrade procedure included in the Upgrade Kit.

Before upgrading any system, please access My Oracle Support (MOS) (https://support.oracle.com) and review any Technical Service Bulletins (TSBs) that relate to this upgrade.

My Oracle Support (MOS) (https://support.oracle.com) is your initial point of contact for all product support and training needs.  A representative at Customer Access Support (CAS) can assist you with MOS registration.

Call the CAS main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html.

See more information on My Oracle Support (MOS) in Appendix A.

## Table of Contents

## List of Tables

## List of Figures

## 1. Introduction

This document defines and describes the DSR Virtual Network Functions Manager (DSR VNFM). DSR VNFM is an application that helps to deploy virtual DSRs quickly by automating the entire deployment process and making it ready to use in the shortest possible time.

The VNFM is responsible for the lifecycle management of virtual network functions (VNFs) under the control of the network function virtualization orchestrator (NFVO).

### 1.1 References

- DSR Cloud Benchmarking Guide
- Or-VNFm Interface defined by ETSI NFV-SOL 003
- Import a Swagger Specification/Swagger UI
- OpenStack prerequisites

### 1.2 Acronyms

An alphabetized list of acronyms used in the document.

**Table 1. Acronyms**

| Acronym | Definition |
|---|---|
| APIGW | Application Program Interface Gateway |
| DA-MP | Diameter Agent Message Processor |
| DB | Database |
| DSR | Diameter Signaling Router |

| Acronym | Definition |
|---|---|
| ETSI | European Telecommunications Standards Institute |
| GUI | Graphical User Interface |
| HA | High Availability |
| IP | Internet Protocol |
| IDIH | Integrated Diameter Intelligence Hub |
| MP | Message Processing or Message Processor |
| NFVO | Network Functions Virtualization Orchestrator |
| NOAM | Network Operations and Maintenance |
| OAM | Operations, Administration, and Maintenance |
| REST | Representational State Transfer |
| SOAM | System Operations and Maintenance |
| STP-MP | Signaling Transfer Point Message Processor |
| UDR | Usage Detail Records |
| UI | User Interface |
| NFVO | Network Function Virtualization Orchestrator |
| VDSR | Virtual Diameter Signaling Router |
| VM | Virtual Manager |
| VNFM | Virtual Network Functions Manager |
| VNF | Virtual Network Functions |
| XMI | External Management Interface |

## 1.3 Terminology

This section describes terminology as it is used within this document.

**Table 2. Terminology**

| Term | Definition |
|---|---|
| OpenStack controller | OpenStack controller controls the selected OpenStack instance. |
| Postman | A tool for creating REST requests. |
| Swagger UI | Swagger UI allows the users to interact with the API resources. |
| VNF instances | VNF instances are represented by the resources. Using this resource, the client can create individual VNF instance resources, and to query VNF instances. |

# 2. Virtual Network Functions Manager Overview

A VNFM automates lifecycle operations for VNFs. Since, each VNF is managed independently, to deploy a DSR it requires creating and instantiating at least two VNFs (one for the network OAM VNF and one for the signaling VNF). Signaling VNFs can be instantiated any time after the network OAM has been instantiated.

The main objective of the DSR VNFM is to provide an ETSI-compliant VNFM. The VNFM would be helpful by:

- Automating lifecycle management (LCM) operations for DSR VNFs. Automation of these operations can reduce their execution time.
- Providing a standardized interface to easily integrate with automation clients, especially ETSI-compliant NFVOs. The DSR VNFM provides a REST API that complies with ETSI NFV-SOL 003.

The VNFM is also helpful in responding quickly to changing customer requirements and delivers solutions for those requirements in a very short time.

The following figure illustrates the interaction between various components of DSR VNFM:

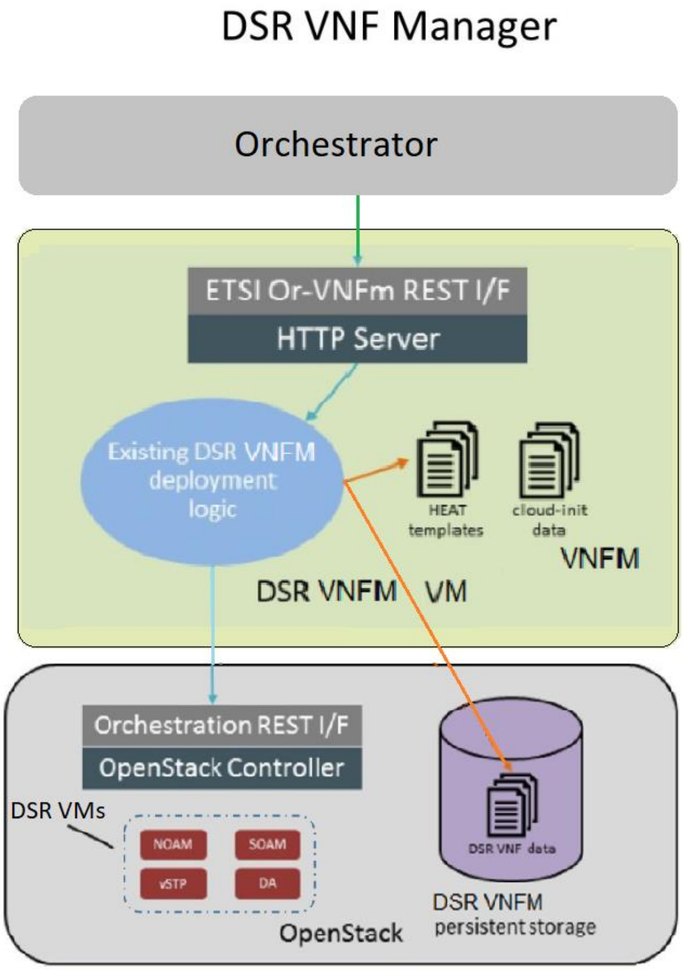

**Figure 1. DSR VNFM Manager**

## 3. Advantage of Using VNFM

Deployment of Virtual DSR (VDSR) was performed using the following methods that required manual processing:

- VM creation and installation process
- HEAT Template based installation (HEAT templates require manual updates)

The manual deployment consumes multiple hours to deploy a fully operational DSR and the HEAT template based installation needed more caution since it requires more manual work.

Using DSR VNFM, users can now deploy a fully operational DSR on OpenStack in less than 15 minutes!

This application benefits both the internal and external customers by reducing operating expenses associated with the implementation and by reducing human errors by eliminating manual intervention.

## 4. DSR VNFM Lifecycle Management Interfaces

The ETSI NFV-SOL 003 VNFM Lifecycle Management interface includes multiple operations. Currently the DSR VNFM supports the following operations:

- Create VNF Identifier
- Instantiate VNF

## 5. DSR VNFM OpenStack Prerequisites

To use the DSR VNFM, you need the following:

1. An OpenStack instance, Mitaka version.
2. One OpenStack tenant per DSR Signaling VNF. The DSR network OAM VNF may share a tenant with one of the signaling VNFs, if allowed.

   ***Note:*** In a production environment, each site would be geographically separate, and therefore on a separate OpenStack instance. For lab setups, separate tenants in the same OpenStack instance are sufficient.
3. A DSR VM image in vmdk format, and named as

   `DSR-8.3.0.0.0_83.15.0.vmdk`

   Where `DSR-8.3.0.0.0_83.15.0.ova` is the name of the OVA image that was delivered with the DSR build. This image must be accessible from every tenant where DSR VMs are deployed.
4. DSR-specific flavors. VNFM assumes the following flavors are defined on each OpenStack tenant on which the DSR VMs are deployed:

   - dsr.noam
   - dsr.soam
   - dsr.da
   - dsr.ipfe
   - dsr.sbr
   - dsr.ss7
   - dsr.vstp
   - spf
   - dsrapigw.admin
   - dsrapigw.app
   - udr.noam
   - appl-idih
   - med-idih
   - db-idih

   For more details about flavor, see the *DSR Cloud Benchmarking Guide*.

5. The specific flavor SPF is defined on each OpenStack tenant on which the SPF VMs are deployed.

   ***Note:*** The user must configure the firewall to allow traffic on required ports such as HTTPS, SCTP, ICMP, FTP, and so on.  For example:

   In Security groups for vSTP, add the following rules to enable traffic on ICMP and SCTP:

   - For ICMP, select Rule as ALL ICMP.
   - For SCTP, select Rule as Other Protocol and IP Protocol as 132.

# 6. Install and Configure the DSR VNFM

Perform the steps below to install and configure the DSR VNFM:

1. Identify an OpenStack instance.

   ***Note:*** The identified OpenStack instance must meet the DSR VNFM OpenStack Prerequisites.

2. Download the HEAT templates.

   ***Note:*** Download the DSR VNFM (Automatic release 2.0) HEAT templates to your local disk from OHC.

3. Upload the image file to OpenStack:

   a. From the OpenStack GUI, navigate to **Projects > Compute-Image**.

      ***Note:*** The OpenStack GUI must be Mitaka version or higher.

   b. Click **Create Image**.

   c. In the Create Image box, perform these suggested options for the following fields:

      i. In the Image Source field, select **Image File**.

      ii. In the Image File field, select the **VNFM 2.0 VM** image. The VNFM Image can be obtained from Oracle Software Delivery Cloud (OSDC) Portal. For example:

      ```
      DSRVNFM_2.0.0.0.0-20.6.0.qcow2
      ```

      The Minimum Disk and Minimum RAM fields can be left blank.

   d. The VNFM flavors must be provided with the appropriate values. For information about flavors, see the *DSR Cloud Benchmarking Guide*.

4. Create the VNFM volume, using the OpenStack CLI:

   a. Create the VNFM volume to use as a part of the OpenStack. The VNFM 2.0 supports a volume with the following specifications:

      Volume size = 8 GB

      Availability-zone = nova

      For example:

      ```
      openstack volume create --size 8 --availability-zone nova <Name of the
      volume Id>
      ```

      The command displays the ID assigned to the newly created volume.

   b. Get the ID of the volume and update the **dsrVnfmVolumeId** parameter with it in the dsrVnfmParams.yaml file.

5. Modify the input parameters.

   a. Edit the **dsrVnfmParams.yaml** HEAT template file.

      ***Note***:

      - The input parameters are given as key/value pairs. Only modify the values (the part to the right side of the colon).
      - The formatting is very important in a YAML file. Do not remove any leading spaces or add any lines to the file.

   b. Edit the values as per the guidelines provided in Table 3.

**Table 3. Parameters and Definitions**

| Parameter | Value |
|---|---|
| dsrVnfmVmName | Enter a name for the VM. Alphanumeric plus "-" and "_" are allowed. |
| dsrVnfmImage | Enter the name of the image uploaded in the previous step. |
| dsrVnfmFlavor | Enter the name of a flavor that is loaded onto OpenStack. |
| xmiPublicNetwork | Enter the name of a network that external clients can use to talk to the VNFM. |
| ntpServer | Enter the IP address of an NTP server with which the VNFM synchronizes the time. The OpenStack controller hosts an NTP server so the IP address of the OpenStack controller is usually a good value. |
| dsrVnfmAZ | Enter the availability zone to place the VNFM. The "nova" is the default availability zone and is usually the right value. |
| dsrVnfmVolumeId | Enter the volume name to use as persistence storage for the VNFM. |

c. Once editing is done, save the file.

6. Deploy the VNFM. The VNFM can be deployed using either of these methods:

   a. Use the OpenStack CLI (recommended):

      Execute the following command:

      ```
      openstack stack create -t dsrVnfmVm.yaml -e dsrVnfmParams.yaml
      <stackName>
      ```

      Example for naming a stack:

      ```
      stack-name
      ```

   b. Using the OpenStack GUI (optional):

      i. Navigate to **Projects->Orchestration->Stacks**.

      ii. Click **Launch Stack**.

## 6.1 Access DSR VNFM Using the REST Interface

The DSR VNFM is accessible using a REST interface. There is no provision to access the REST interface through CLI, or GUI, however it can be accessed through a Swagger specification provided for the REST interface. There are many other compatible interfaces that can be used with popular REST testing tools. Some of the most widely used tools that can be used with the REST testing tool are:

- Swagger UI

  With the Swagger UI, a GUI can be generated from the Swagger specification.

  Swagger specifications can be found post VNFM installation at, (http://<VNFM IP>:8080/docs/vnfm/).

- Postman

  Another popular tool for creating REST requests is the Postman tool. It is available as a standalone app and as a Chrome browser plugin. You can import a Swagger specification to allow Postman to understand the VNFM REST API in detail, which allows it to assist you while creating requests and interpreting responses.

## 6.2 Supported VNF's by the DSR VNFM

**Table 4. Supported VNFs and VMs**

| Supported VNFs | Supported VMs |
|---|---|
| NOAM | NOAM (Active/Standby) |
| Signaling | SOAM (Active/Standby), DA-MP, STP-MP, IPFE, SBR, UDR |
| SPF | SPF |
| APIGW | DB Server ( Active/Standby ) , Admin Server, Application Server(s) |
| IDIH | APP, MEDIATION, DB Server |

# 7. Deploying DSR VNFs

**Prerequisites**: A virtual infrastructure satisfying the DSR VNFM OpenStack Prerequisites.

## 7.1 Create a VNF Instance

The following procedure creates the VNF Instance:

1. Before a DSR VNF is instantiated, the user must first issue a request to create a VNF instance by using the command **create VNF instance**.
2. Creating a VNF instance informs the VNFM that a user has requested to instantiate a VNF at some point in the future.
3. The VNFM returns a VNF ID that must be saved for future use while performing operations on the same VNF.

   ***Note:*** Each VNF has its own VNF ID, so if it is required to create a DSR with two signaling VNFs, then issue the request to create a VNF instance three times, once for the network OAM VNF, and once for each signaling VNFs.

For more information about the full list of all inputs and possible outputs of the **create VNF instance** command, see ETSI NFV-SOL 003, section 5.4.2.3.1, or the DSR VNFM Swagger specification. Swagger specifications can be found post VNFM installation at (http://<VNFM IP>:8080/docs/vnfm/).

The following image illustrates the VNF instance creation:

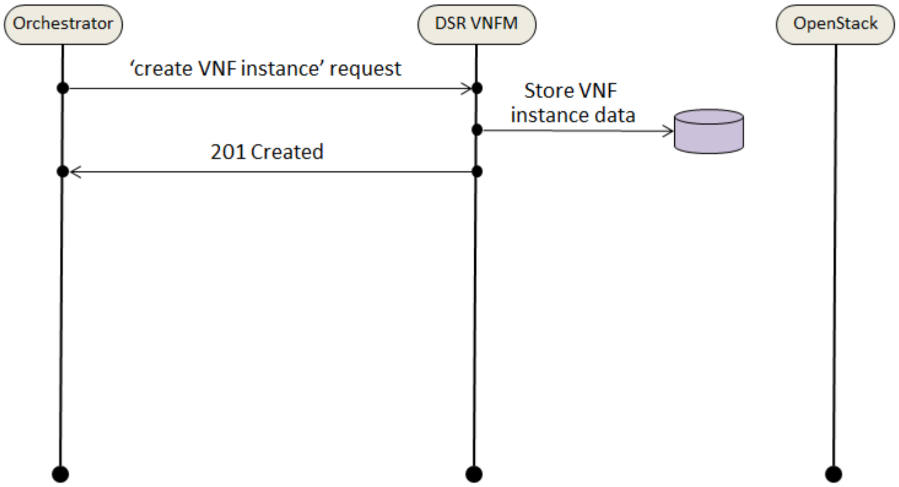

**Figure 2. VNF Create Instance Request**

**Sample Request**

Create VNF instance request generated.

Resource URL:  http://<<VNFM HOST IP>>:8080/vnfm/v1/vnf_instances

Accept:  application/json

Content-Type:  application/json

Example for **NOAM**:

```
{
  "vnfdId": "dsrNetworkOam",
  "vnfInstanceName": "DemoNoam",
  "vnfInstanceDescription": "DemoNoam "
}
```

Example for **Signaling**:

```
{
  "vnfdId": "dsrSignaling",
  "vnfInstanceName": "DemoSoam",
  "vnfInstanceDescription": "Description"
}
```

Example for **SPF**:

```
{
  "vnfdId": "SPF",
  "vnfInstanceName": "DemoSPF",
  "vnfInstanceDescription": "Description for SPF VNF"
}
```

Example for **APIGW**:

```
{
  "vnfdId": "dsrApiGw",
  "vnfInstanceName": "DemoApiGw",
  "vnfInstanceDescription": "Description for APIGW VNF"
}
```

Example for **IDIH**:

```
{
  "vnfdId": "dsrIdih",
  "vnfInstanceName": "DemoIdih",
  "vnfInstanceDescription": "Description for IDIH VNF"
}
```

**Sample Response**

`201 Created`

Create VNF Instance Response

Content-Type: application/json

Resource URL: `http://<<myhost-IP>>:8080/vnfm/v1/vnf_instances/dsrNetworkOam-38e2c734-2e1f-4ed8-b18b-08d6c30c60d2`

```
{
 "id": "dsrNetworkOam-cdf2d110-ac13-4c54-b87e-c49935cd8b33",
 "vnfdId": "dsrNetworkOam",
 "instantiationState": "NOT_INSTANTIATED",
 "vnfInstanceName": "DemoNoam"
}
```

***Note:*** VNFM supports both the secured and the unsecured URL (HTTPS with port 8443 and HTTP with port 8080).

Table 5 describes the parameters used for sending request to VNFM:

**Table 5. Parameters and Definitions**

| Parameter | Definitions |
|---|---|
| vnfdId | Identifier of the VNF instance deployment ID to be created |
| vnfInstanceName | Name of the VNF instance to be created |
| vnfInstanceDescription | Description of the VNF instance |

## 7.2 Query VNF Instance

The diagram describes a sequence for querying/reading information about a VNF instance.

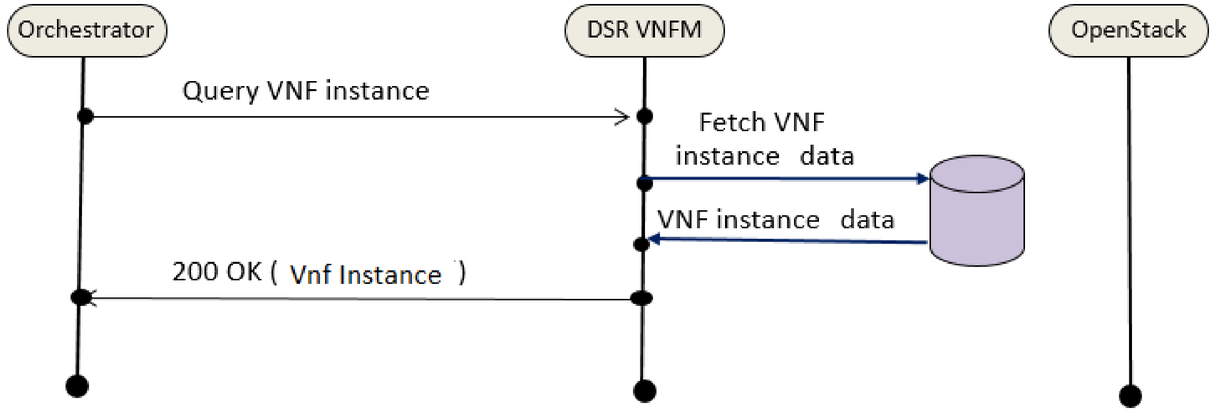

**Figure 3. Query VNF Instance**

VNF instance query, as illustrated in Figure 3. Query VNF Instance, consists of the following steps:

1. If the NFVO intends to read information about a particular VNF instance, it sends a GET request to the **Individual VNF instance** resource, addressed by the appropriate VNF instance identifier (Vnf Id) in its resource URI.

2. The VNFM returns a **200 OK** response to the NFVO, and includes specific data structure of type **VnfInstance** related to the VNF instance identifier (Vnf Id) in the payload body.
3. If the NFVO intends to query all VNF instances, it sends a GET request to the **VNF instances** resource.
4. The VNFM returns a **200 OK** response to the NFVO, and includes zero or more data structures of type **VnfInstance** in the payload body.

### 7.2.1 Query Individual VNF Instance

**Sample Request for Single VNF Instance:**

URL: GET: https://<<VNFM HOST IP>>:8443/vnfm/v1/vnf_instances/<<VNF Instance ID>>

**Sample Response for Single VNF Instances:**

Accept: application/json

Content-Type: application/json

```
{
   "id": "dsrNetworkOam-793a2420-adab-4347-9667-489ae671b767",
   "vnfdId": "dsrNetworkOam",
   "instantiationState": "NOT_INSTANTIATED",
   "vnfInstanceName": "string",
   "vnfInstanceDescription": "string",
   "vnfProvider": "Oracle",
   "vnfProductName": "DSR",
   "vnfSoftwareVersion": "8.3",
   "vnfdVersion": "1.0",
   "onboardedVnfPkgInfoId": "N/A",
   "links": {
             "self": {
                      "href":
"https://localhost:8443/vnflcm/v1/vnf_instances/dsrNetworkOam-793a2420-adab-
4347-9667-489ae671b767"
                     },
             "instantiate": {
                             "href":
"https://localhost:8443/vnflcm/v1/vnf_instances/dsrNetworkOam-793a2420-adab-
4347-9667-489ae671b767/instantiate"
                            }
            }
}
```

### 7.2.2 Query All VNF Instance

**Sample Request for all VNF Instances:**

URL: GET: https://<<VNFM HOST IP>>:8443/vnfm/v1/vnf_instances

**Sample Response for all VNF Instances:**

Accept: application/json

Content-Type: application/json

Response Body for No VNF Instances

```
[]
```

Response Body for VNF Instances

```
[
  {
    "id": "dsrNetworkOam-793a2420-adab-4347-9667-489ae671b767",
    "vnfdId": "dsrNetworkOam",
    "instantiationState": "NOT_INSTANTIATED",
    "vnfInstanceName": "string",
    "vnfInstanceDescription": "string",
    "vnfProvider": "Oracle",
    "vnfProductName": "DSR",
    "vnfSoftwareVersion": "8.3",
    "vnfdVersion": "1.0",
    "onboardedVnfPkgInfoId": "N/A",
    "links": {
              "self": {
                       "href":
"https://localhost:8443/vnflcm/v1/vnf_instances/dsrNetworkOam-793a2420-adab-
4347-9667-489ae671b767"
                      },
              "instantiate": {
                              "href":
"https://localhost:8443/vnflcm/v1/vnf_instances/dsrNetworkOam-793a2420-adab-
4347-9667-489ae671b767/instantiate"
                             }
              }
  }
]


[
  {
    "id": "dsrSignaling-715fd05f-d9ea-4bee-8ab8-5b921e52efde",
    "vnfdId": "dsrSignaling",
    "instantiationState": "NOT_INSTANTIATED",
    "vnfInstanceName": "api",
    "vnfInstanceDescription": "string",
    "vnfProvider": "Oracle",
    "vnfProductName": "DSR",
    "vnfSoftwareVersion": "8.3",
    "vnfdVersion": "1.0",
    "onboardedVnfPkgInfoId": "N/A",
    "links": {
              "self": {
                       "href":
"https://localhost:8443/vnflcm/v1/vnf_instances/dsrSignaling-715fd05f-d9ea-4bee-8ab8-5b921e52efde"
                       },
              "instantiate": {
                              "href":
"https://localhost:8443/vnflcm/v1/vnf_instances/dsrSignaling-715fd05f-d9ea-4bee-8ab8-5b921e52efde/instantiate"
                             }
              }
  }
]
```

## 7.3 Instantiate the Network OAM VNF

To start a DSR deployment, it is required to instantiate a DSR network OAM VNF. Before deploying the VNF, make sure the following information is available:

- The VNF ID for a previously created DSR Network OAM VNF instance
- Information about the OpenStack instance on which the VNF must be deployed:
    - OpenStack Controller URI
    - Domain name
    - Username
    - Password
    - Tenant name
- The name of a Public Network in your chosen OpenStack instance that will carry OAM traffic.
- The IP of an NTP server accessible by VMs within the selected OpenStack instance. The OpenStack controller that controls the selected OpenStack instance normally hosts an NTP server, and is often a good choice.

For more information about the full list of all inputs and possible outputs of the **create VNF instance** command, see ETSI NFV-SOL 003, section 5.4.2.3.1, or the DSR VNFM Swagger specification. Swagger specifications can be found post VNFM installation at (http://<VNFM IP>:8080/docs/vnfm/).

**Sample Request**

Instantiating NOAM Request generated.

Resource URL:  http://<<myhost-IP>>:8080/vnfm/v1/vnf_instances/<VNF ID received from create request>/instantiate

Accept:  application/json

Content-Type:  application/json

```
{
        "flavourId": "DSR NOAM",
    "instantiationLevelId": "HA",
    "extVirtualLinks": "extVirtualLinks",
                "extManagedVirtualLinks": [],
    "vimConnectionInfo":[ {
        "id": "vimid",
        "vimType": "OpenStack",
        "interfaceInfo": {
          "controllerUri": "http://oortcloud.us.oracle.com:5000/v3"
        },
        "accessInfo": {
            "username": "dsrci.user",
            "password": "xxxxx",
            "domain": "default",
            "tenant": "DSR CI"
        }
    }],
    "localizationLanguage": "localizationLanguage",
    "additionalParams": {
        "xmiNetwork": {
            "name": "ext-net3",
            "ipVersion": "IPv4"
        },
        "ntpServerIp": "10.250.32.10"
    }
}
```

**Sample Response**

Instantiating NOAM Request.

```
202 Accepted
```

***Notes***:

- The 202 response means that the request was accepted for processing. The VNF might take up to 15 minutes to become fully operational. Use the DSR GUI to determine when the VNF is operational.
- If the VNFM creates a VNF that is operational, but has no Signaling VNFs, then it is required to deploy one or more Signaling VNF, and create the DIAMETER configuration data (peers, connections, etc.) for those VNFs to perform DIAMETER routing.
- The supported NOAM Flavor is DSR NOAM.

Table 6 describes the parameters used for sending request to VNFM.

**Table 6. Parameters and Definitions**

| Parameter | Definitions |
|---|---|
| flavourId | Identifier of the VNF deployment flavour to be instantiated |
| id | Unique ID of the Vim |
| vimType | Virtual Infrastructure Manager (OpenStack) |
| controllerUri | VIM URI |
| xmiNetwork | Network for talking to external devices |
| ntpServerIp | IP of the NTP server |

## 7.4 Instantiate the First Signaling VNF

To deploy the first signaling VNF, the following must be available:

- A previously instantiated DSR Network OAM VNF.
- The VNF ID for a previously created DSR Signaling VNF instance.
- Information about the OpenStack instance on which you want to deploy the VNF:
  - OpenStack Controller URI
  - Domain name
  - Username
  - Password
  - Tenant name
- The name of a Public Network in your chosen OpenStack instance that will carry OAM traffic.
- The name of a Public Network in your chosen OpenStack instance that will carry Signaling traffic.

  ***Note:*** This should be a different network than the one that carries OAM traffic.
- The IP address of the NTP server accessible by VMs within the selected OpenStack instance. The OpenStack controller that controls your chosen OpenStack instance normally hosts an NTP server, and is often a good choice.
- OpenStack resource IDs for the XMI IPs from both NOAM VMs.

***Note:*** The resource IDs can be obtained by examining the network OAM stack to which the identified signaling VNF would be attached.

- Name of the active NOAM VM.
- Name of the NOAM SG.

Figure 1 illustrates the VNF instantiation:

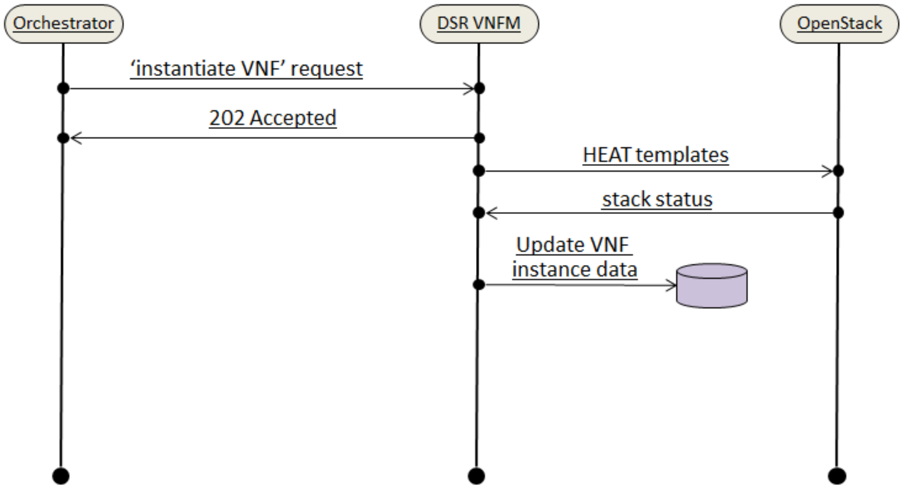

**Figure 4. VNF Instantiate Request**

Table 7 contains the supported Instantiation levels to instantiate a VNF resource for the DSR signaling VNF.

***Note:*** The definition of Instantiation Level can be modified through the configuration file provided in VNFM. The values defined in the table are the default parameters that come with VNFM deployment.

**Table 7. Supported Instantiation Levels for DSR Signaling VNF**

| Signaling Flavors supported by VNFM | Small | | | | | Medium | | | | | Large | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DAMP | IPFE | STP | SBR | UDR | DAMP | IPFE | STP | SBR | UDR | DAMP | IPFE | STP | SBR | UDR |
| Diameter | 2 | 2 | 0 | 0 | 0 | 4 | 2 | 0 | 0 | 0 | 8 | 2 | 0 | 0 | 0 |
| SS7 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 8 | 0 | 0 |
| Diameter+ SS7 | 2 | 2 | 2 | 0 | 0 | 4 | 2 | 4 | 0 | 0 | 8 | 2 | 8 | 0 | 0 |
| Diameter+ SBR | 2 | 2 | 0 | 3 | 0 | 4 | 2 | 0 | 6 | 0 | 8 | 2 | 0 | 9 | 0 |
| Diameter+ SS7+SBR | 2 | 2 | 2 | 3 | 0 | 4 | 2 | 4 | 6 | 0 | 8 | 2 | 8 | 9 | 0 |
| Diameter+ UDR | 2 | 2 | 0 | 0 | 2 | 4 | 2 | 0 | 0 | 2 | 8 | 2 | 0 | 0 | 2 |
| SS7+UDR | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 4 | 0 | 2 | 0 | 0 | 8 | 0 | 2 |
| Diameter+ UDR | 2 | 2 | 2 | 3 | 2 | 4 | 2 | 4 | 6 | 2 | 8 | 2 | 8 | 9 | 2 |

### 7.4.1 Determine the NOAM XMI Resource IDs

From the OpenStack GUI:

1. Change your view to the tenant on which the DSR Network OAM VNF was deployed.
2. Navigate to **Orchestration->Network->Network Topology**.

   A diagram of all VMs in the tenant displays.

   ***Note:*** The diagram may take a few minutes to display.
3. Click on one of the NOAM VMs.

   A screen displays with information about the specific NOAM VM.
4. Save the resource ID for the XMI port provided in the IP addresses section of the screen.

   ***Note:*** The IP Addresses section of the screen contains information about the network ports and resource IDs assigned to the VM.
5. Repeat the previous step for the other NOAM VM.

For more information about the full list of all inputs and possible outputs of the **create VNF instance** command, see ETSI NFV-SOL 003, section 5.4.2.3.1, or the DSR VNFM Swagger specification. Swagger specifications can be found post VNFM installation at (http://<VNFM IP>:8080/docs/vnfm/).

**Sample Request**

Instantiating the first signaling VNF request generated.

Resource URL: http://<<myhost-IP>>:8080/vnfm/v1/vnf_instances/<VNF ID received from create request>/instantiate

Accept: application/json

Content-Type: application/json

```
{
                "flavourId": "DIAMETER+SS7", ( Supported are DIAMETER,SS7 &
DIAMETER+SS7)
                "instantiationLevelId": "small",
                "extVirtualLinks": "extVirtualLinks",
                "extManagedVirtualLinks": [{
                                                "id": "",
                                                "virtualLinkDescId": "",
                                                "resourceId": "8a4d1ec6-367a-
4b1a-978d-2c4eae3daec3"
                                },
                                {
                                                "id": "",
                                                "virtualLinkDescId": "",
                                                "resourceId": "2bed5886-8c97-
4623-8da3-9c500cce71e3"
                                }
                ],
                "vimConnectionInfo":[ {
        "id": "vimid",
        "vimType": "OpenStack",
        "interfaceInfo": {
          "controllerUri": "http://oortcloud.us.oracle.com:5000/v3"
        },
        "accessInfo": {
            "username": "dsrci.user",
            "password": "xxxx",
            "domain": "default",
            "tenant": "DSR CI"
        }
    }],
                "localizationLanguage": "localizationLanguage",
                "additionalParams": {
                                "xmiNetwork": {
                                                "name": "ext-net3",
                                                "ipVersion": "IPv4"
                                },
                                "xsiNetwork": {
                                                "name": "ext-net2",
                                                "ipVersion": "IPv4"
                                },
                                "ntpServerIp": "10.250.32.10",
                                "primaryNoamVmName": "NOAM00-32cd6138",
                                "noamSgName":
"dsrNetworkOam_NOAM_32cd6138_SG"
                }
}
```

**Sample Response**

```
202 Accepted
```

**Sample Request**

Instantiating the signaling VNF request with SBR generated

Resource URL: http://<<myhost-IP>>:8080/vnfm/v1/vnf_instances/<VNF ID received from create request>/instantiate

Accept: application/json

Content-Type: application/json

```
{
                "flavourId": "DIAMETER+SS7", ( Supported are DIAMETER,SS7 &
DIAMETER+SS7)
                "instantiationLevelId": "small",
                "extVirtualLinks": "extVirtualLinks",
                "extManagedVirtualLinks": [{
                                                "id": "",
                                                "virtualLinkDescId": "",
                                                "resourceId": "8a4d1ec6-367a-
4b1a-978d-2c4eae3daec3"
                                },
                                {
                                                "id": "",
                                                "virtualLinkDescId": "",
                                                "resourceId": "2bed5886-8c97-
4623-8da3-9c500cce71e3"
                                }
                ],
                "vimConnectionInfo":[ {
        "id": "vimid",
        "vimType": "OpenStack",
        "interfaceInfo": {
          "controllerUri": "http://oortcloud.us.oracle.com:5000/v3"
        },
        "accessInfo": {
            "username": "dsrci.user",
            "password": "xxxx",
            "domain": "default",
            "tenant": "DSR CI"
        }
    }],
                "localizationLanguage": "localizationLanguage",
                "additionalParams": {
                                "xmiNetwork": {
                                                "name": "ext-net3",
                                                "ipVersion": "IPv4"
                                },
                                "xsiNetwork": {
```

```
                                                "name": "ext-net2",
                                                "ipVersion": "IPv4"
                                },
                                "sbrNetwork": {
                                                "name": "ext-net3",
                                                "ipVersion": "IPv4"
                                },
                                "ntpServerIp": "10.250.32.10",
                                "primaryNoamVmName": "NOAM00-32cd6138",
                                "noamSgName":
"dsrNetworkOam_NOAM_32cd6138_SG"
                }
}
```

**Sample Response**

Instantiating the signaling VNF with SBR response

`202 Accepted`

***Notes***:

- The 202 response means that the request was accepted for processing. The VNF might take up to 15 minutes to become fully operational. Use the DSR GUI to determine when the VNF is operational.
- If the VNFM creates a VNF that is operational, but has no DIAMETER configuration data, then create the necessary configuration data (peers, connections, etc.) to perform DIAMETER routing.
- Supported Signaling flavors are DIAMETER, SS7, and DIAMETER+SS7.

Table 8 describes the parameters used for sending request to VNFM.

**Table 8. Parameters and Definitions**

| Parameter | Definitions |
|---|---|
| flavourId | Identifier of the VNF deployment flavour to be instantiated |
| instantiationLevelId | Identifier of the instantiation level of the deployment flavour to be instantiated. If not present, the default instantiation level as declared in the VNFD is instantiated. |
| resourceId | The identifier of the resource in the scope of the VIM or the resource provider |
| id | Unique ID of the Vim |
| vimType | Virtual Infrastructure Manager (OpenStack) |
| controllerUri | VIM URI |
| xmiNetwork | Network for talking to external devices |
| name | Network name, for example; ext-net |
| ipVersion | IP version IPv4 or IPv6 |
| xsiNetwork | Network for talking to DIAMETER peers |
| ntpServerIp | IP of the NTP server |

| Parameter | Definitions |
|---|---|
| primaryNoamVmName | Name of primary NOAM VM on which the configured XML is loaded |
| noamSgName | The server group of the NOAM VM |

## 7.5 Instantiate More Signaling VNFs

To instantiate more Signaling VNFs, simply repeat the above procedures. You would need to create another DSR Signaling VNF instance, and you must deploy each Signaling VNF on a separate OpenStack instance.

Note: For lab installations, a separate tenant on the same OpenStack instance is acceptable.

## 7.6 Instantiate the APIGW VNF

To start APIGW deployment, it is required to instantiate an APIGW VNF. Before deploying the VNF, make sure the following information is available:

- The VNF ID for a previously created APIGW VNF instance.
- Information about the OpenStack instance on which the VNF must be deployed:
  - OpenStack Controller URI
  - Domain name
  - Username
  - Password
  - Tenant name
- The name of a public network in the selected OpenStack instance that will carry APIGW traffic.
- The name of a public network in the selected OpenStack instance that will carry signaling traffic.

  ***Note:*** This should be a different network than the one that carries APIGW traffic
- The IP of an NTP server accessible by VMs within the selected OpenStack instance. The OpenStack controller that controls the selected OpenStack instance normally hosts an NTP server, and is often a good choice.

For more information about the full list of all inputs and possible outputs of the **create VNF instance** command, see ETSI NFV-SOL 003, section 5.4.2.3.1, or the DSR VNFM Swagger specification. Swagger specifications can be found post VNFM installation at (http://<VNFM IP>:8080/docs/vnfm/).

Table 9 contains the supported Instantiation levels to instantiate the VNF resource for DSR APIGW VNF.

**Table 9. Supported Instantiation levels for DSR APIGW VNF**

| APIGW Flavors supported by VNFM | Small | | | Medium | | | Large | | |
|---|---|---|---|---|---|---|---|---|---|
| | ADMIN | APP | DB | ADMIN | APP | DB | ADMIN | APP | DB |
| **APIGW** | 1 | 1 | Active/ Standby | 1 | 2 | Active/ Standby | 1 | 3 | Active/ Standby |

**Sample Request**

Instantiating APIGW Request generated.

URL: http://<<VNFM HOST IP>>:8080/vnfm/v1/vnf_instances/< VNF ID received from create request>/instantiate

Accept: application/json

Content-Type: application/json

```
{
  "flavourId": "APIGW",
  "instantiationLevelId": "small",
  "extVirtualLinks": "extVirtualLinks",
  "extManagedVirtualLinks": [],
  "vimConnectionInfo": [
    {
      "id": "vimid",
      "vimType": "OpenStack",
      "interfaceInfo": {
        "controllerUri": "http://oortcloud.us.oracle.com:5000/v3"
      },
      "accessInfo": {
        "username": "dsrat.user",
        "password": "xxxx",
        "domain": "default",
        "tenant": "DSR AT Dev 2"
      }
    }
  ],
  "localizationLanguage": "localizationLanguage",
  "additionalParams": {
    "ntpServerIp": "10.250.32.10",
    "keyName": "apiGwKey",
    "xmiNetwork": {
      "name": "ext-net3",
      "ipVersion": "IPv4"
    },
    "xsiNetwork": {
      "name": "ext-net2",
      "ipVersion": "IPv4"
    },
    "externalLoadBalancer": "10.10.10.10",
    "dsrMPList": "10.10.10.4:49152"
  }
}
```

**Sample Response**

Instantiating APIGW Request

```
202 Accepted
```

***Notes***:

- The 202 response means that the request was accepted for processing. The VNF might take up to 6 minutes to become fully operational. Use the DSR GUI to determine when the VNF is operational.
- The supported flavor is APIGW.
- The keyName must be generated in OpenStack before instantiating APIGW VNF.

Table 10 describes the parameters used for sending request to VNFM.

**Table 10. Parameters and Definitions**

| Parameter | Definitions |
|---|---|
| flavourId | Identifier of the VNF deployment flavour to be instantiated |
| instantiationLevelId | Identifier of the instantiation level of the deployment flavour to be instantiated. If not present, the default instantiation level as declared in the VNFD is instantiated. |
| id | Unique ID of the Vim |
| vimType | Virtual Infrastructure Manager (OpenStack) |
| controllerUri | VIM URI |
| xmiNetwork | Network for talking to external devices |
| xsiNetwork | Network for talking to DIAMETER peers |
| ntpServerIp | IP of the NTP server |
| keyName | Name of key-pair to be used for compute instance |

## 7.7 Instantiate the IDIH VNF

To start IDIH deployment, it is required to instantiate a signaling VNF. Before deploying the VNF, make sure the following information is available:

- The VNF ID for a previously created IDIH VNF instance.
- Information about the OpenStack instance on which the VNF must be deployed:
  - OpenStack Controller URI
  - Domain name
  - Username
  - Password
  - Tenant name

- The name of a public network in the selected OpenStack instance that will carry the IDIH traffic.
- The IP of an NTP server accessible by VMs within the selected OpenStack instance. The OpenStack controller that controls the selected OpenStack instance normally hosts an NTP server, and is often a good choice.
- The network ID of the private network in the selected OpenStack instance that will carry OAM traffic. A signaling stack must be brought up first and then the ID of the internal network generated from this stack must be used for instantiating IDIH.
- The name of the internal private network in the selected OpenStack instance that will allow communication between Application, Mediation, and Database servers.

For more information about the full list of all inputs and possible outputs of the **create VNF instance** command, see ETSI NFV-SOL 003, section 5.4.2.3.1, or the DSR VNFM Swagger specification. Swagger specifications can be found post VNFM installation at (http://<VNFM IP>:8080/docs/vnfm/).

**Sample Request**

Instantiating IDIH Request generated

URL: http://<<VNFM HOST IP>>:8080/vnfm/v1/vnf_instances/<VNF ID received from create request>/instantiate

Accept: application/json

Content-Type: application/json

```
{
   "flavourId":"IDIH",
   "instantiationLevelId":"small",
   "extVirtualLinks":"extVirtualLinks",
   "extManagedVirtualLinks":[
 {
         "id":"id1",
         "virtualLinkDescId":"",
         "resourceId":"aae72b3d-d189-4464-a217-58bb0320065b"
 }
   ],
   "vimConnectionInfo":[
      {
         "id":"vimid",
         "vimType":"OpenStack",
         "interfaceInfo":{
            "controllerUri":"http://oortcloud.us.oracle.com:5000/v3"
         },
         "accessInfo":{
            "username":"dsrat.user",
            "password":"xxxx",
            "domain":"default",
            "tenant":"DSRAT_Feature_Test4"
         }
      }
   ],
   "localizationLanguage":"localizationLanguage",
   "additionalParams":{
      "ntpServerIp":"10.250.32.10",
      "xmiNetwork":{
         "name":"ext-net3",
         "ipVersion":"IPv4"
      },
      "idihIntNetwork":{
         "idihIntPrivateNetwork":"test",
         "idihIntPrivateSubnet":"test-sub"
      }
   }
}
```

**Sample Response**

Instantiating APIGW Request

```
202 Accepted
```

***Notes***:

- The 202 response means the request was accepted for processing.  The VNF might take up to 6 minutes to become fully operational.  Use the DSR GUI to determine when the VNF is operational.
- The supported flavor is IDIH.

Table 11 describes the parameters used for sending request to VNFM.

**Table 11.  Parameters and Definitions**

| Parameter | Definitions |
|---|---|
| flavourId | Identifier of the VNF deployment flavour to be instantiated |
| instantiationLevelId | Identifier of the instantiation level of the deployment flavour to be instantiated.  If not present, the default instantiation level as declared in the VNFD is instantiated. |
| resourceId | The Identifier of the Private network (imi) |
| id | Unique ID of the Vim |
| vimType | Virtual Infrastructure Manager (OpenStack) |
| controllerUri | VIM URI |
| xmiNetwork | Network for talking to external devices |
| IdihIntNetwork | Private network for communication between application, mediation and database servers |

| Parameter | Definitions |
|---|---|
| ntpServerIp | IP of the NTP server |

## 7.8 Instantiate the SPF VNF

To start a DSR deployment, it is required to instantiate a SPF VNF. Before deploying the VNF, make sure the following information is available:

- The VNF ID for a previously created SPF VNF instance.
- Information about the OpenStack instance on which the VNF must be deployed:
  - OpenStack Controller URI
  - Domain name
  - Username
  - Password
  - Tenant name
- The name of a Public Network in the selected OpenStack instance that will carry the SPF traffic.
- The IP of an NTP server accessible by VMs within the selected OpenStack instance. The OpenStack controller that controls the selected OpenStack instance normally hosts an NTP server, and is often a good choice.

For more information about the full list of all inputs and possible outputs of the **create VNF instance** command, see ETSI NFV-SOL 003, section 5.4.2.3.1, or the DSR VNFM Swagger specification. Swagger specifications can be found post VNFM installation at (http://<VNFM IP>:8080/docs/vnfm/).

**Sample Request**

Instantiating SPF VNF request generated

Resource URL: http://<<VNFM HOST IP>>:8080/vnfm/v1/vnf_instances/< VNF ID received from create request>/instantiate

Accept: application/json

Content-Type: application/json

```
{
    "flavourId": "SPF",
    "instantiationLevelId": "HA",
    "extVirtualLinks": "extVirtualLinks",
    "extManagedVirtualLinks": [],
    "vimConnectionInfo":[ {
        "id": "vimid",
        "vimType": "OpenStack",
        "interfaceInfo": {
          "controllerUri": "http://oortcloud.us.oracle.com:5000/v3"
        },
        "accessInfo": {
            "username": "xxxx.user",
            "password": "xxxx",
            "domain": "default",
            "tenant": "<Tenant Name>"
        }
    }],
    "localizationLanguage": "localizationLanguage",
    "additionalParams": {
        "networks": [{
            "network": "ext-net3",
            "fixed_ip": "10.196.12.248"
        }],
        "ntpServerIp": "10.250.32.10",
        "keyName": "spf",
        "httpListenIpv4": "10.196.12.248",
        "httpListenPort": 9999,
        "timeZone": "GMT",
        "dbVolumeSize": 5
    }
}
```

**Sample Response**

```
202 Accepted
```

***Notes***:

- The 202 response means that the request was accepted for processing. The VNF might take up to 15 minutes to become fully operational. Use the DSR GUI to determine when the VNF is operational.
- The supported Flavor is **SPF.**

The following table describes the parameters used for sending request to VNFM:

**Table 12. Parameters and Definitions**

| Parameter | Definitions |
|---|---|
| flavourId | Identifier of the VNF deployment flavour to be instantiated |
| id | Unique ID of the Vim |
| vimType | Virtual Infrastructure Manager (OpenStack) |
| controllerUri | VIM URI |
| xmiNetwork | Network for talking to external devices |
| ntpServerIp | IP of the NTP server |
| fixed ip | The IP to be assigned to SPF VM |
| httplistenIpv4 | IPv4 listen address for HTTP connection |
| httplistenport | Listen port for HTTP connection |

| Parameter | Definitions |
|---|---|
| timezone | Timezone of the VM instance |
| dbvolumesize | Database size in GB |

## 7.9 Import HTTPS/SSL Certificate into VNFM

### 7.9.1 Recombine Existing PEM Keys and Certificates into VNFM

If you have an existing private key and certificates for your server's domain in PEM format, combine them into a PKCS keystore, then convert the PKCS keystore into a Java keystore.

Execute the following command:

```
cat <midfile.1.cert.pem> <midfile.2.cert.pem> > intermediates.cert.pem
```

Where <`midfile.1.cert.pem`> and <`midfile.2.cert.pem`> are the names of intermediate certificate files.

***Note:*** If you have multiple intermediate certificates, combine them in any order.

1.
   ```
   openssl pkcs12 -export -in <dsrVnfm.pem> -inkey <dsrVnfm.key> -
   certfile <intermediate.cert.pem> -passin pass:<existingpassword> -
   passout pass: xxxx -out vnfm_default.p12 -name "<yourDomainName>"
   ```

   For example:

   ```
   openssl pkcs12 -export -in dsrVnfm.pem -inkey dsrVnfm.key -passin
   pass: xxxx -passout pass:xxxx -out vnfm_default.p12 -name dsrvnfm
   ```

2.
   ```
   keytool -importkeystore -srckeystore vnfm_default.p12 -srcstorepass
   xxxx -srcstoretype PKCS12 -destkeystore vnfm_default.jks -
   deststorepass xxxx -alias dsrVnfm
   ```

   For example:

   ```
   keytool -importkeystore -srckeystore vnfm_default.p12 -srcstorepass
   xxxx -srcstoretype PKCS12 -destkeystore vnfm_default.jks -
   deststorepass xxxx -alias dsrVnfm
   ```

   ***Note:*** keytool is the java key and certificate management utility provided by Java. It exist in `jre/bin/keytool`.

Where,

- <`dsrVnfm.pem`>: The existing signed certificate file that matches your existing private key.
- <`dsrVnfm.key`>: The existing private key file.
- <`intermediate.cert.pem`>: The existing intermediate certificates that complete the chain from your certificate to a root CA.
- <`yourDomainName`>: The complete domain name of your server.
- <`existingpassword`>: The password that allows access to the existing key file.
- <`yourpassword`>: The password that allows access to your new keystore. Provide at least six characters.

***Notes***:

- `destkeystore` file name should be same as mention in the command (`vnfm_default.jks`).

- `srcstorepass` is the password that is given in first command (`-passout pass: xxxx`) and it should also be same as mention in the command (`xxxx`)
- `deststorepass` is the password that is used to open the certificate file (`vnfm_default.jks`) and should also be same as mention in the command (`xxxx`), because the same file name and password is used in Tomcat Apache to access the SSL certificate.

### 7.9.2 Copy Created Certificate (vnfm_default.jks) into VNFM

Once vnfm box is installed, a self-signed certificate is created by VNFM and is placed in the `/var/vnfm/certificate/vnfm_default.jks` directory by default. This certificate is valid for 365 days.

The client must copy their created certificate with same name as `vnfm_default.jks` into `/var/vnfm/certificate/` directory and override the existing `vnfm_default.jks` certificate.

***Note:*** After the making the certificate changes, client must restart the apache tomcat server to reflect the updated certificate in VNFM.

Run the following command to restart the apache tomcat server:

1. `sudo /usr/share/vnfm/apache-tomcat-9.0.6/bin/shutdown.sh`
2. `sudo /usr/share/vnfm/apache-tomcat-9.0.6/bin/startup.sh`

### 7.9.3 VNFM Self Signed Certificate Generation

1. Create a `vnfmCert.conf` configuration file as shown in the example below (provide your own details in the respective fields):

```
[ req ]
default_bits = 2048
default_md = sha256
distinguished_name = req_distinguished_name
req_extensions = req_ext
[ req_distinguished_name ]
countryName = Country Name (2-letter code)
stateOrProvinceName = State or Province Name (full name)
localityName = Locality (e.g. city name)
organizationName = Organization (e.g. company name)
commonName = Common Name (your.domain.com)
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
DNS.1 = *.localhost
DNS.2 = 127.0.0.1
DNS.3 = *.oracle.com
DNS.4 = *.oraclecorp.com
```

2. Generate a key pair and a signing request by executing:

```
openssl req -new -keyout dsrVnfm.key -out dsrVnfm.csr -newkey rsa:2048 -
config vnfmCert.conf
```

It will ask password to create private key file.

***Note:*** To skip passphrase in private key, just add -nodes (read: "No DES encryption") parameter from the command.

Check if CSR contains the SAN by executing:

```
openssl req -noout -text -in sslcert.csr | grep DNS
```

3. Generating a self-signed certificate:

   To generate a temporary certificate, which is acceptable for 365 days, by executing:

   ```
   openssl x509 -req -days 365 -in dsrVnfm.csr -signkey dsrVnfm.key -sha256 -
   out dsrVnfm.crt -extfile ca.cnf -extensions req_ext
   ```

   Enter pass phrase for dsrVnfm.key: <type pass phrase of private key>

   Check if CSR contains the SAN by executing:

   ```
   openssl req -noout -text -in sslcert.csr | grep DNS
   ```

4. Convert the CRT to PEM format:

   Use the `openssl` tool to convert the CRT to a PEM format that is readable by the reporter:

   ```
   openssl x509 -in dsrVnfm.crt -out dsrVnfm.pem -outform PEM
   ```

5. To convert the PEM-format keys to Java KeyStores:

   ```
   openssl pkcs12 -export -in dsrVnfm.pem -inkey dsrVnfm.key -passin
   pass:4srVN6M -passout pass:4srVN6M -out vnfm_default.p12 -name dsrvnfm
   ```

6. Convert the `vnfm_default.p12` to a Java `keystore vnfm_default.jks`, by executing:

   ```
   keytool -importkeystore -srckeystore vnfm_default.p12 -srcstorepass
   4srVN6M -srcstoretype PKCS12 -destkeystore vnfm_default.jks -deststorepass
   4srVN6M -alias dsrVnfm
   ```

   ***Note:*** After importing certificate into java keystore, it is a good practice to check if the certificate information is correct or not. Keytool is the java jdk tool, which exists in `jdk/bin`.

   ```
   keytool -list -v -keystore [enter keystore name] -storepass [enter
   keystore password]
   ```

   ***Note:*** The `vnfm_default.jks` is the ssl certification file which is being used in VNFM https to establish the ssl connection.

   While importing certificate into java keystore, provide `-alias dsrVnfm`. If it prompts to override, type YES.

   Use the password "`xxxx`".

   ***Note:*** Certificate file name (`vnfm_default.jks`) and alias name (`dsrVnfm`) must be the same as mentioned above.

## 7.10 Terminating a VNF

The DSR VNFM does not yet support terminating a VNF. The user must directly delete the stack from the OpenStack that holds the virtual resources for the VNF.

## 8. Discover Stack

1. It is an N/B LCM Discover Rest I/F.
2. It is used to discover the created stack in OpenStack and save the stack information (parameter file and VNF instance) in the VNFM persistent directory. This information can be used for further requests by the orchestrator. For example, to scale out the stack.
3. Before discovering the stack, make sure the following information is available:
   - The Stack ID of the previously created stack.
   - The following information about the OpenStack instance on which the Stack must be discovered:
     - OpenStack Controller URI
     - Domain name
     - Username
     - Password
     - Tenant name
4. The Interface discovers the stack and performs the following operations:
   a. Download the parameter file of the discovered stack.
   b. Create the Instance file of the discovered stack.
   c. These two files are saved in `/var/vnfm/instances/<autoDiscovery InstanceId>/` directory.

**Sample Request for Discover Interface**

```
Request URL : POST:
https://<<VNFM HOST IP>>:8443/vnflcm/v1/discover/<<discover stack id>>
For example:
https://localhost:8443/vnflcm/v1/discover/b30ac203-5fe1-4007-a3ba-
078f3422708b
Accept: application/json
Content-Type: application/json
Request Body:
{
  "vimConnectionInfo": [
    {
      "id": "vimid",
      "vimType": "OpenStack",
      "interfaceInfo": {
        "controllerUri": "http://oortcloud.us.oracle.com:5000/v3"
      },
      "accessInfo": {
        "username": "dsrat.user",
        "password": "xxxx",
        "domain": "default",
        "tenant": "DSR AT Dev 1"
      }
    }
  ]
}
```

**Sample Response for Discover Interface**

```
Response Code : 200
{ "vnfInstanceId": "dsrApiGw-b5597bce0f1fe0a9-17f463f7-2c32-4b05-a04d-
23d345ea0f5d", "autoDiscoverStackId": "b30ac203-5fe1-4007-a3ba-078f3422708b"
}
```

***Note:*** Discover VNF stack supports only those stacks that are created by the VNFM templates.

## Appendix A. My Oracle Support (MOS)

MOS (https://support.oracle.com) is your initial point of contact for all product support and training needs. A representative at Customer Access Support (CAS) can assist you with MOS registration.

Call the CAS main number at **1-800-223-1711** (toll-free in the US), or call the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html. When calling, make the selections in the sequence shown on the Support telephone menu:

1. Select 2 for New Service Request.
2. Select 3 for Hardware, Networking, and Solaris Operating System Support.
3. Select one of the following options:

   For technical issues such as creating a new Service Request (SR), select 1.

   For non-technical issues such as registration or assistance with MOS, select 2.

You are connected to a live agent who can assist you with MOS registration and opening a support ticket. MOS is available 24 hours a day, 7 days a week, and 365 days a year.

**Emergency Response**

In the event of a critical service situation, emergency response is offered by the CAS main number at 1-800-223-1711 (toll-free in the US), or by calling the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html. The emergency response provides immediate coverage, automatic escalation, and other features to ensure that the critical situation is resolved as rapidly as possible.

A critical situation is defined as a problem with the installed equipment that severely affects service, traffic, or maintenance capabilities, and requires immediate corrective action. Critical situations affect service and/or system operation resulting in one or several of these situations:

- A total system failure that results in loss of all transaction processing capability
- Significant reduction in system capacity or traffic handling capability
- Loss of the system’s ability to perform automatic system reconfiguration
- Inability to restart a processor or the system
- Corruption of system databases that requires service affecting corrective actions

- Loss of access for maintenance or recovery operations
- Loss of the system ability to provide any required critical or major trouble notification

Any other problem severely affecting service, capacity/traffic, billing, and maintenance capabilities may be defined as critical by prior discussion and agreement with Oracle.

**Locate Product Documentation on the Oracle Help Center**

Oracle Communications customer documentation is available on the web at the Oracle Help Center (OHC) site, http://docs.oracle.com. You do not have to register to access these documents. Viewing these files requires Adobe Acrobat Reader, which can be downloaded at http://www.adobe.com.

1. Access the **Oracle Help Center** site at http://docs.oracle.com.
2. Click **Industries**.
3. Under the **Oracle Communications** subheading, click the **Oracle Communications documentation** link. The Communications Documentation page displays. Most products covered by these documentation sets display under the headings **Network Session Delivery and Control Infrastructure** or "**Platforms**."
4. Click on your Product and then the Release Number. A list of the entire documentation set for the selected product and release displays. To download a file to your location, right-click the PDF link, select `Save target as` (or similar command based on your browser), and save to a local folder.