HP NFV Director

HP NFV Director

Version 3.0

Integration Guide

Edition: 1.0

For Red Hat Enterprise Linux Server 6.6

June 2015

Legal Notices

Contents

Tables

Figures

Preface

In this Guide

This document is a developer guide of NFV Director and describes various interfaces for integration and customization.

The document provides the following information:

- It describes the HP NFV Director and its role in OpenNFV program.
- It describes the functional and component architecture of the NFV Director.
- Functionality of each component of the NFV Director.
- Interaction between components of the NFV Director.
- NFV Data modeling for on-boarding, deploying, and activating the Virtual Network Functions (VNF).
- North Bound interface provided by the NFV Director.
- Customization of various NFV components to adapt and integrate any new VNF.

Audience

This guide is intended for the following users:

- VNF Creators
- Solution Developers
- Software Development Engineers

Software Versions

The software versions referred to in this document are as follows:

Product Version	Supported Operating systems
NFV Director 3.0	Red Hat Enterprise Linux Server release 6.6

Table 1 Software versions

Typographical Conventions

`Courier` Font:

- Source code and examples of file contents.
- Commands that you enter on the screen.
- Pathnames
- Keyboard key names

Italic Text:

- Filenames, programs and parameters.
- The names of other documents referenced in this manual.

Bold Text:

- To introduce new terms and to emphasize important words.

References

- *HP NFV Director User Guide - Overview*
- *HP NFV Director User Guide - Advanced*
- *HP NFV Director API - Reference guide*
- *HP NFV Director Workflows – Reference Guide*
- *HP NFV Director VNF On-boarding Guide*
- *Unified Correlation Analyzer for Event Based Correlation Reference Guide*
- *Unified Correlation Analyzer for Event Based Correlation Value Pack Development Guide*
- *Unified Correlation Analyzer for Event Based Correlation – Clustering and HA Guide*
- *OSS Open Mediation Installation and Configuration Guide*
- *SiteScope User Guide*
- *HPSA User Guide*

Chapter 1

Introduction

HP Network Functions Virtualization (NFV) Director is an ETSI MANO compliant NFV orchestrator that is responsible for lifecycle management of network services (NS) across the entire operator's domain, like multiple datacenters.

The NFV Director performs the following functions:

- Manages network services.
- Manages NS and VNF packages (functions such as on-boarding new NS and VNF packages).
- Creates new NS during run-time.
- Lifecycle management of NS instantiation and NS instance—includes functions such as update, scale-out/in, event collection and correlation, and termination.
- Manages integrity and visibility of NS instances through their lifecycle and also manages the relationship between the NS instances and the VNF instances.
- Manages NS instance topology across the entire operator domain.
- Monitors NS instances.
- Manages NS instance automation across the entire Operator domain.
- Manages policies and enforces them for the NS instances.

1.1 HP OpenNFV Program

The NFV Director is part of the HP OpenNFV Program, operationalizing NFV through an NFV orchestrator with embedded VNF manager capabilities.

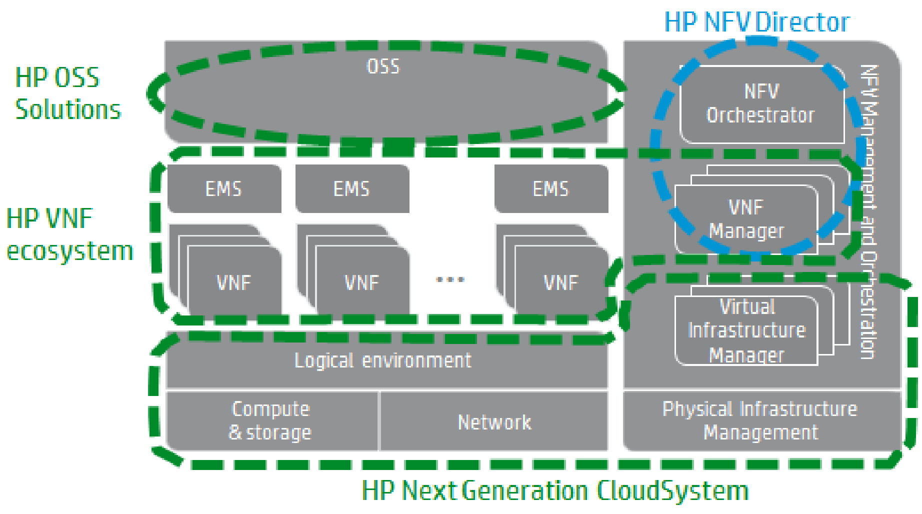


Figure 1 HP NFV Director

1.2 HP NFV Director Functionality

From a high level view, HP NFV Director has four main modules:

- Fulfillment module, based on HP Service Activator (HPSA)
- Assurance module, composed of several components like HP Assurance Gateway, HP SiteScope and HP Open Mediation, among others.
- Data and Integration module, which main components are the Workflow Manager and the object inventory.
- NFV Director product graphical user interfaces, dedicated to NFV Director Operations

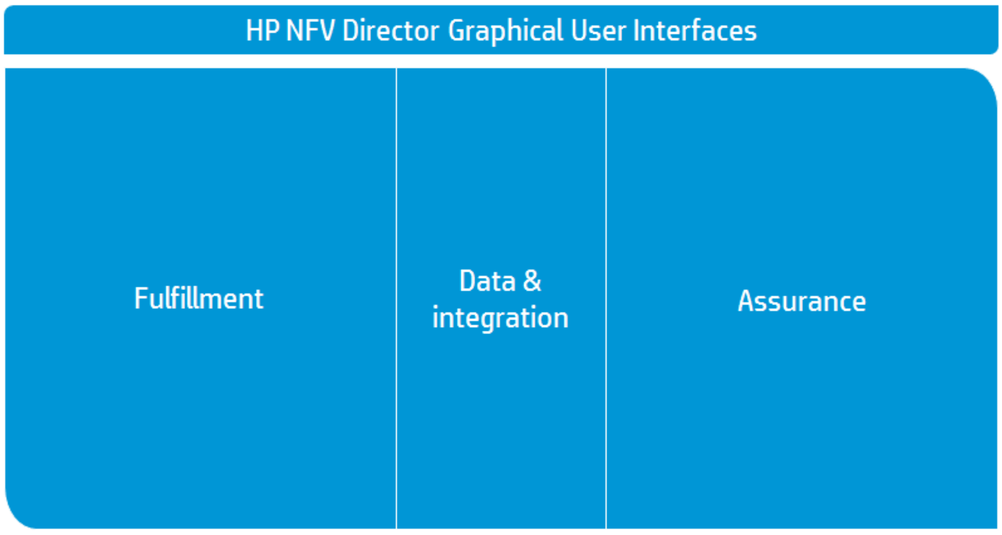


Figure 2 HP NFV Director Modules

Fulfillment, Assurance and Data & Integration modules are divided at the same time in four main functional planes.

Every plane interacts with the other planes allowing the correct performance and management of all VNFs and NSs.

These planes are:

- Model-based coordination and control plane, responsible of overall coordination of the Management and Orchestration layer.
- Agent-less monitoring plane, responsible of collecting virtual infrastructure information.
- Dynamic topology-based correlation plane, responsible of correlation across the different levels introduced by NFV abstractions.
- Rules-based autonomous actions plane, responsible of autonomous action based on rules and external information

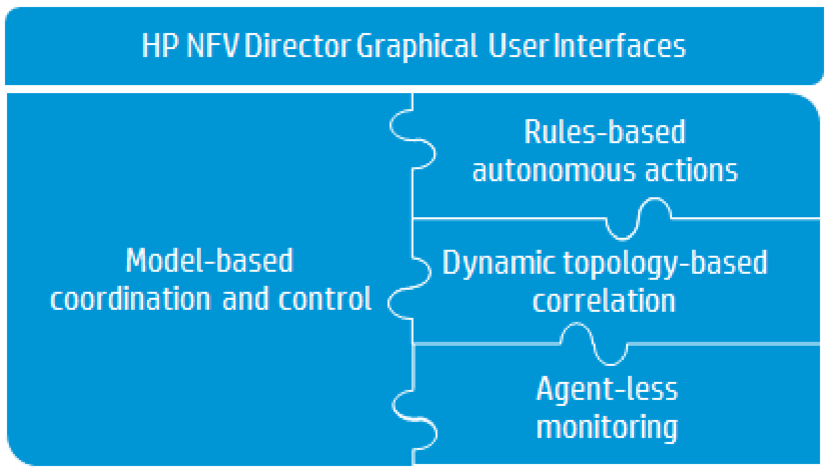

HP NFV Director Graphical User Interfaces
Model-based coordination and control
Rules-based autonomous actions
Dynamic topology-based correlation
Agent-less monitoring

Figure 3 HP NFV Director Functionality

In the component architecture view, these functional planes and the Graphical User Interfaces are included in one of the HP NFV Director main components:

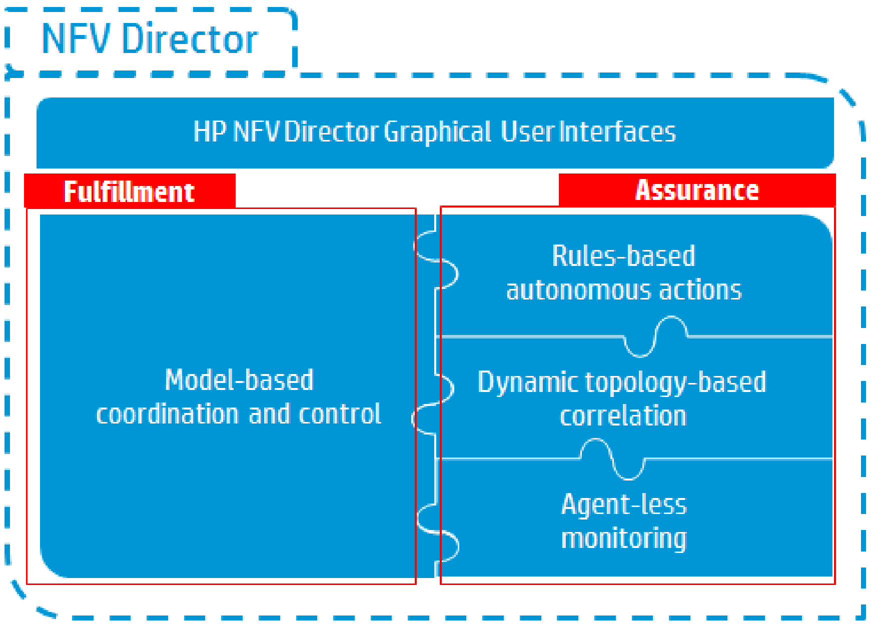

NFV Director
HP NFV Director Graphical User Interfaces
Fulfillment
Assurance
Model-based coordination and control
Rules-based autonomous actions
Dynamic topology-based correlation
Agent-less monitoring

Figure 4 HP NFV Director Component Architecture

Finally, the main functions that the HP NFV Director performs are:

- Orchestrates the allocation and release of resources to be used by VNF.
- Automatically discovers NFV Infrastructure resources from VIM.
- Applies policies for orchestration and resource allocation.
- Installs and instantiates VNF software images on virtual and physical machines.
- Configures virtual or physical networks to connect instantiated machines.
- Assumes the role of a VNF Manager in the absence of a VNF Manager for a VNF.
- Monitors and accounts NFVI resources for a specific VNF and network service.

- Checks the usage of a resource against policies.
- Collects performance data for NFVI compute, storage, and network resources.
- Exports collected KPIs and topology as csv files for other application integrations.
- Detects and handles faults in NFVI.
- Provides the ability to consume fault data for a VNF and fault management.
- Proactively monitors NFVI resources and generates fault in the absence of fault reporting from NFVI.
- Interfaces with the VNF Manager to provide NFVI resources to be consumed by VNF managers for VNFs.
- Provides a VNF and NS Catalog for the operator to instantiate and manage VNF and NS.
- Provides a framework for fault correlation and root-cause analysis process to determine the reason for fault conditions and their impact on VNFs and network services.
- Provides a framework to determine corrective actions. It triggers such actions at one or more action points within the NFV framework or to the OSS.
- Defines and provides an interface to external entities (such as OSS and NMS) for the following functions:
    - VNF on-boarding
    - Lifecycle management of VNF instances (in co-ordination with VNF Managers)
    - NFV Policy Management
    - Performance Data of VNF and NS
    - NFVI resource usage accounting

For features supported in version 3.0, refer to the *NFV Director Release Notes*.

1.3 HP NFV Director Component Architecture

In the previous section, we described the high level view of HP NFV Director, as *From a high level view,* HP NFV Director has four main modules:

- Fulfillment module, based on HP Service Activator (HPSA)
- Assurance module, composed of several components like HP Assurance Gateway, HP SiteScope and HP Open Mediation, among others.
- Data and Integration module, which main components are the Workflow Manager and the object inventory.
- NFV Director product graphical user interfaces, dedicated to NFV Director Operations

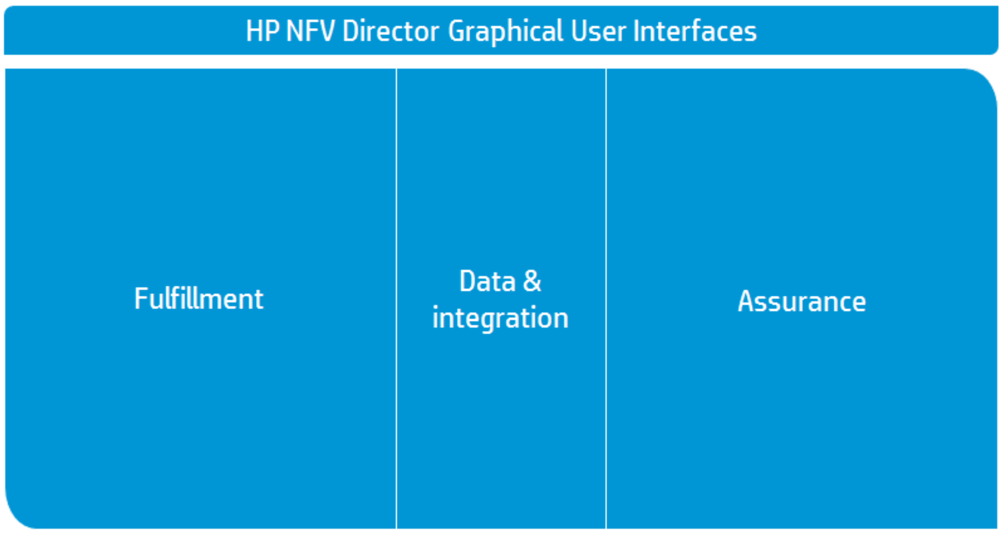


Figure 2 HP NFV Director Modules shows.

A more detailed view of these modules is described in next figure:

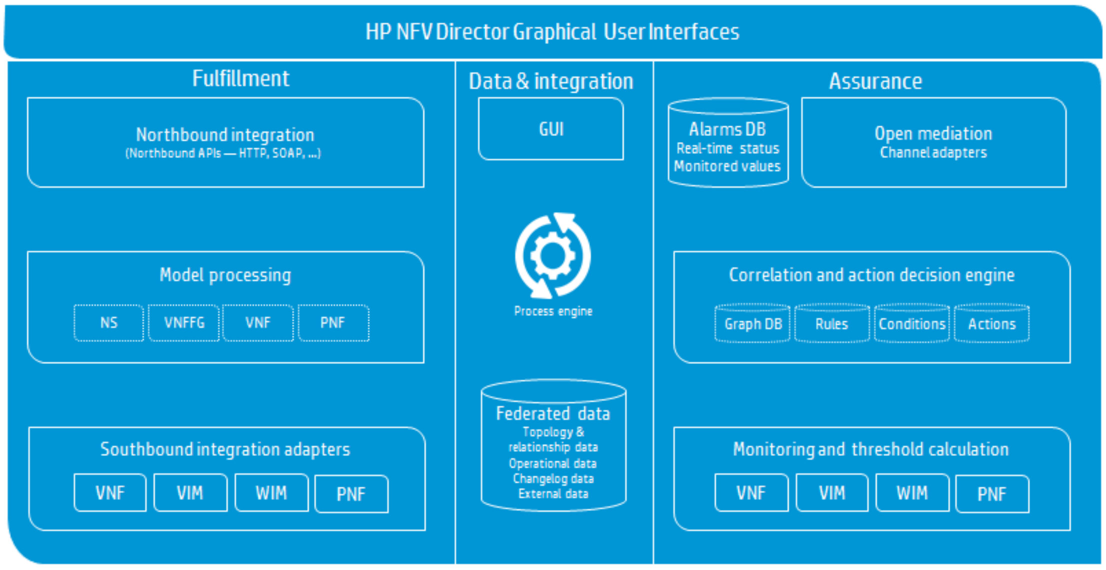


Figure 5 HP NFV Director Detailed Modules

1.3.1 Fulfillment

Next figure shows the Fulfillment Architecture in depth:

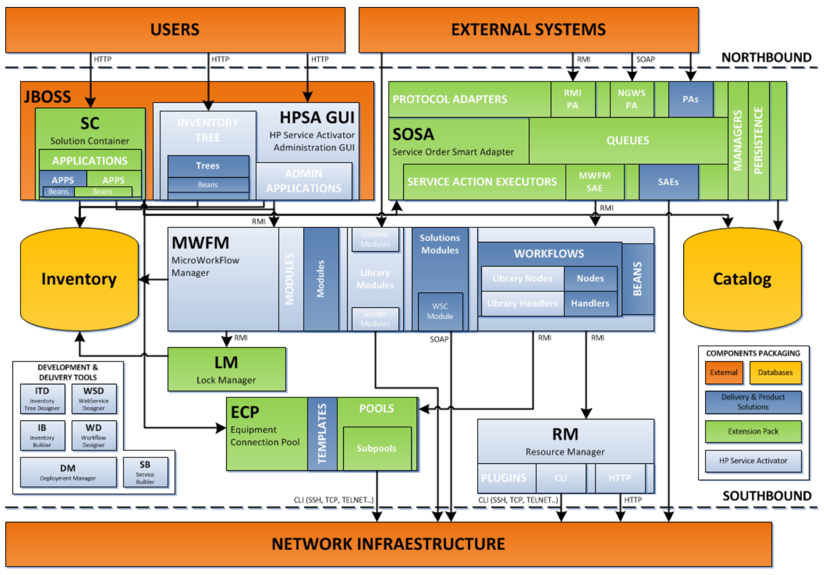


Figure 6 HP NFV Director Detailed Modules

1.3.1.1 Co-ordination and Control plane

In the NFV Director Fulfillment module we find three main sections:

- Configuration adaptation layer
- Model processing layer
- Configuration Adapters

A short description of every section is explained in next paragraphs.

Configuration adaptation layer

The configuration adaptation layer provides a proven, adaptable and modular Northbound interface to easily integrate HP NFV Director with the Opertation Service Systems (OSS).

The Northbound architecture is based on HP Service Activator Sosa Module and its main function is managing requests using queues.

The two principal adapters in this layer are:

- Web Service Protocol Adapter. It allows execution of task
- REST Protocol Adapter. It allows access to data model.

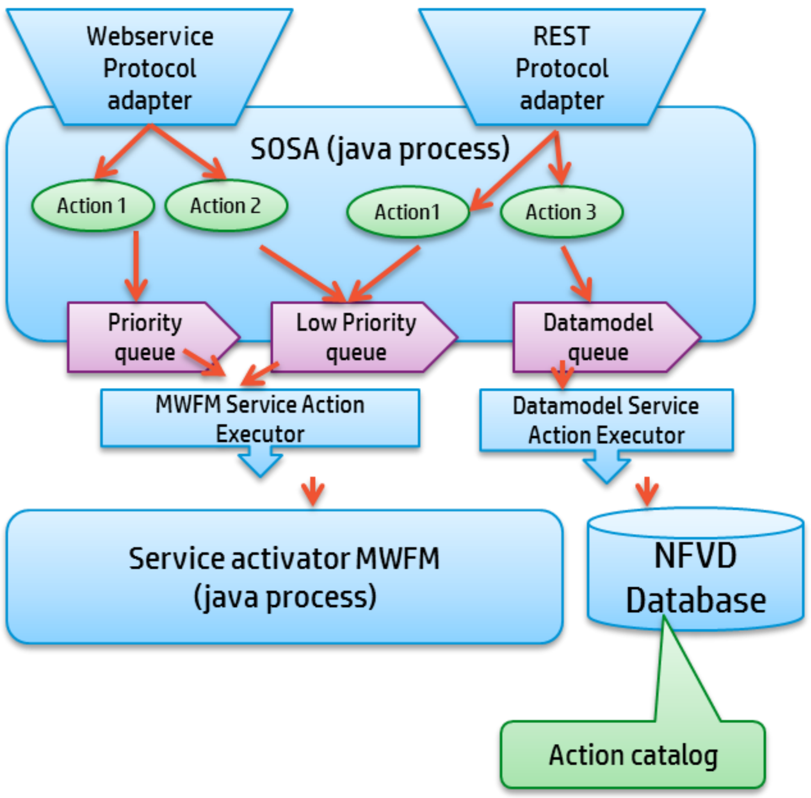


Figure 7 HP SOSA Mapping behaviour

Model processing layer

It is built on proven HP Service Activator technology and its goal is coordinate the NFV Orchestrator functionality.

Configuration Adapters

They are plugins (generic or vendor-specific) for full multi-vendor support. The whole set of plugins we refer to it as Southbound Interface.

Each plug-in allows to connect in a certain way (web services, SSH, Telnet, REST, and so on) and execute a set of instructions/commands.

The instructions/commands are stored in the database in the form of templates and allow variables for providing the reusability of those templates.

Each template can be organized in sections and can optionally have rollback instructions for each section

1.3.2 Assurance

1.3.2.1 Monitoring plane

The monitoring plane consists of three parts:

- An integration component called *Assurance Gateway*
- *Monitoring* that is agentless
- *Mediation* to collect events from external sources

The *Assurance Gateway* receives and processes the VNF topology information from Fulfillment, and updates UCA-EBC graph database for correlation.

It also receives and processes monitoring notifications from Fulfillment and delegates appropriate action with all the data to agentless monitoring component.

Assurance gateway provides lifecycle notifications for artifacts. Assurance gateway exports topology and KPI data for integrations with OSS, analytics, and other applications.

The agentless *monitoring* component, built using HP SiteScope, can monitor a wide variety of monitoring points, issuing events or executing commands when pre-defined thresholds are crossed. As different VNFs have different monitoring needs, the monitoring points and thresholds are automatically configured by NFV Director as the VNF is provisioned or modified. As an agentless solution, the NFV Director does not require installation of monitoring agents on the target systems.

The individual VNF managers are responsible for monitoring their own internal faults and performance, possibly augmented with traps and key performance indicators (KPIs) provided by the NFV Director. When VNF management functionality is provided out of the NFV Director, through an embedded VNF manager, it might be necessary to collect application-specific monitoring information. HP Sitescope provides the capability to develop custom monitors for VNF resources and VNF applications.

HP SiteScope templates are used to standardize a set of monitor types and configurations into a single structure. This structure can then be repeatedly deployed as a group of monitors targeting multiple elements of the monitored environments.

Templates accelerate the deployment of monitors across the enterprise through a single-operation deployment of groups, monitors, alerts, remote servers, and configuration settings.

Finally, HP NFV Director includes HP Open *Mediation*, which provides the following capabilities:

- Allows the integration of multiple products.
- Defines common communication patterns:
  - Alarm flow
  - Resynchronization
  - Action invocation
  - Topology notification
- Provides numerous connectivity features:
  - Web Services—SOAP/HTML, SOAP/JMS, HTML/REST
  - Files—Local file access, FTP/FTPS/SFTP
  - Database—JDBC
  - Enterprise Java—JMS , JMX , RMI
  - Other—TCP/UDP, HTTP/HTTPS, IRC, LDAP, SMTP/POP3/IMAP, RSS, SMPP, SNMP, XMPP

In the context of the NFV Director, OpenMediation is used to integrate with Sitescope, EMS, VNF Manager, or any other source of events.

1.3.2.2 Correlation plane

The Unified Correlation Analyzer for Event Based Correlation product (also known as UCA Expert by analogy with the legacy TeMIP Expert software) offers a new and generalized event based correlation solution.

Based on the JBoss Drools 5.5.0.Final, rule engine, UCA for EBC offers the capability to create comprehensive functional correlation sets called Value packs that implement the correlation logic. This correlation is performed by running certain rules (the rules are written in a Java-based language). Any Value Pack can support or use predefined functionalities such as Alarm collection, filtering, lifecycle, as well as Generic Actions.

The UCA for EBC can perform the following functions:

- Collect alarms and map them into Operator Alarm model (Alarm).
- The Alarm model is based on a mix of X733 and OSS/J Fault Management Model.
- Run several scenarios (rule engines) in parallel and in sequence to implement complex correlation algorithms. Each set of scenarios implementing a single correlation solution is grouped inside a UCA for EBC Value Pack.
- Dispatch Alarm objects for different scenarios.
- Execute rules based on the scenario input stream and generate suitable output. For example, actions to external systems.
- Control the scenario input stream using an alarm-based filtering layer.
- Execute actions such as storing in a database, creating a Trouble Ticket, creating a new Alarm, group alarms, forward an alarm to another scenario, or execute a Generic Action through the OSS Open Mediation v6.2 (NOM v6.2) layer.

Rule files are JBoss Rule files. Hence, both JBoss Expert and Fusion rules are supported in the UCA for EBC rule files. JBoss Drools Expert and JBoss Drools Fusion are JBoss Drools basic modules.

Over this basic functionality, UCA for EBC also provides a software development kit (SDK) that allows solution developers to easily build UCA for EBC Value Packs (Functional Correlation block). Administration tools (both command line and a GUI) are also available to manage, monitor, and troubleshoot the product.

UCA for EBC can be connected with a mediation bus (OSS Open Mediation v6.2), providing the capability to collect alarms from any source (NMS) and performing actions in return.

1.3.2.3 Autonomous Action plane

UCA Automation software, which is a combination of both business rule engine and workflow engine, enables a clear separation of what to automate and how to automate. All complexities of automation, such as how to access a network resource (can be a network element, an element component, an EMS, or NMS), what its credentials are, which specific transport mechanism should be used to connect to the resource, what specific OS versions of the device should be supported, what specific commands need to be sent, are abstracted from the business rules.

This software enables administrators to create, update, and read the business rules with utmost clarity and to maintain them efficiently. It also empowers the administrators to store the knowledge gained regarding the automation in the form of business rules focusing on what part without bothering about the how part.

Another advantage of UCA Automation software is that for most of the resolution automations, the operator should know only the business rules and not the business rules technologies to implement day-to-day operational changes to the decisions.

Thus, UCA Automation System is a platform for building value added resolution automations based on a judicious combination of business rules and workflows.

1.3.3 HP NFV Director and OSS

As of V3.0 HP NFV Director deliver brand new Operator interfaces dedicated to Operators. These native NFV D Operators’ user interfaces have been developed using HP Unified OSS

Console, a common and secure Web UI, highly customizable platform used as a foundation to providing and unifying the native graphical user interfaces of several HP OSS products, inclusive of HP NFV Director and HP NFV Analytics. These products' user interfaces benefit from the Unified OSS Console modern and responsive WEB UI technology (HTML5/CSS3/full JavaScript stack), built with a client/server architecture and using a JavaScript technology stack, with a highly performant and scalable presentation server.

In this architecture, the NFV Director Web client exposes the NFV Director user interfaces and communicates with NFV Director capabilities via the GUI presentation server and a dedicated NFV Director plug-in. NFV Directors user interfaces consist of a set of product views and operations made available to the users to manage and monitor network services and VNFs across their lifecycle.

NFV Director Web client runs in most popular web browsers (Microsoft Internet Explorer Version 10 or later, Mozilla Firefox V32 or later and Google ChromeV37 or later). The web client communicates with the GUI presentation server (a Linux server) which in turn interact with the NFV Director northbound APIs.

HP NFV Analytics native graphical user interfaces have also been developed using HP Unified OSS Console technology, integrated in this architecture with a dedicated plug-in (the OSS-A plugin). When both NFV Director & NFV Analytics are deployed as one solution, these share the same GUI presentation server, as illustrated in the following figure.

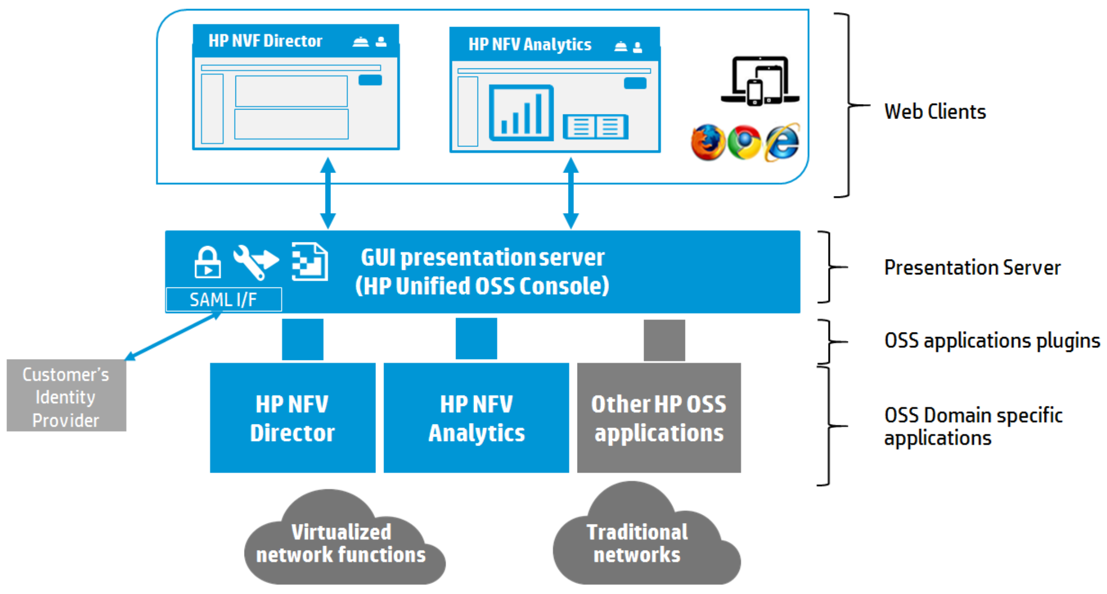


Figure 8 NFV Director and OSS

HP NFV Director graphical user interfaces are primarily aimed at Operations, facilitating Operator's work in the day to day operations and monitoring of the virtualized network services and functions managed and orchestrated by NFV Director. HP NFV Director Operator interfaces cover 3 main areas:

- The NFV Director catalog, which provides the user with a set of views and actions on the on boarded NS & VNFs packages, easy navigation into the NS and VNF components, and a set of operations to manage and deploy network services and VNFs easily

- The NFV Director instances which provides the user with a set of monitoring views and actions, on the deployed Network services and VNF instances, easy navigation into the NS and VNF instance components and their health status.

- The NFV Director resources, which provides the user with a set of views and actions on Datacenter resources, VIMs and VNF managers.

Further to providing easy to use Operator interfaces, NFV Director deployments also benefit from the Unified OSS console architecture openness to further customize, enrich and extent the by default provided views. This includes the ability to change the HP theme /branding, the ability to contextually launch an external application, role based access control, the configuration of different workspaces for different types of users, the modification of the views layout and the content of the views. For user authentication, an SAML interface supports the integration with the existing Customer identity provider through the SAML V2.0 protocol in order to support a single Sign-on across applications.

More details on HP NFV Director Product interfaces are described in further sections of this document.

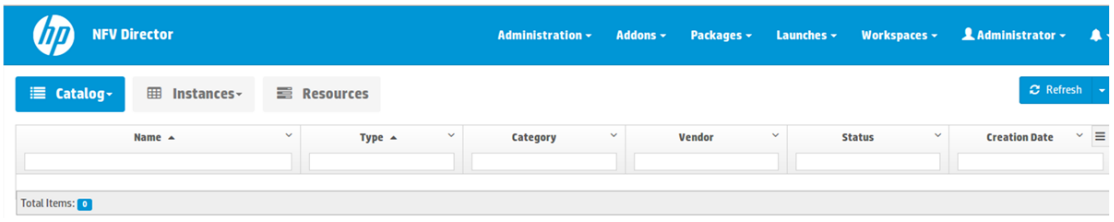


Figure 9 HP NFV Director Operators' Graphical User Interfaces

1.3.3.1 HP NFV Director and traditional OSS

Network Functions Virtualization (NFV) handles the lifecycle of Network Services and VNFs. HP NVF Director brings a new set of management capabilities which can easily be integrated with the OSS managing traditional networks, using the Unified OSS console integration capabilities.

It provides the required operational support, to enable you (among others) to:

- Safely transition your network from physical to virtualized mode
- Easily integrate virtualized network functions into your OSS and IT environment
- Grow from simple virtualization use cases to complex use cases
- Enable easier and quicker deployment of network functions for faster innovation and increased revenue

1.3.4 Interaction between HP NFV Director components

A global vision of the interaction between HP NFV Director components is shown in next figure:

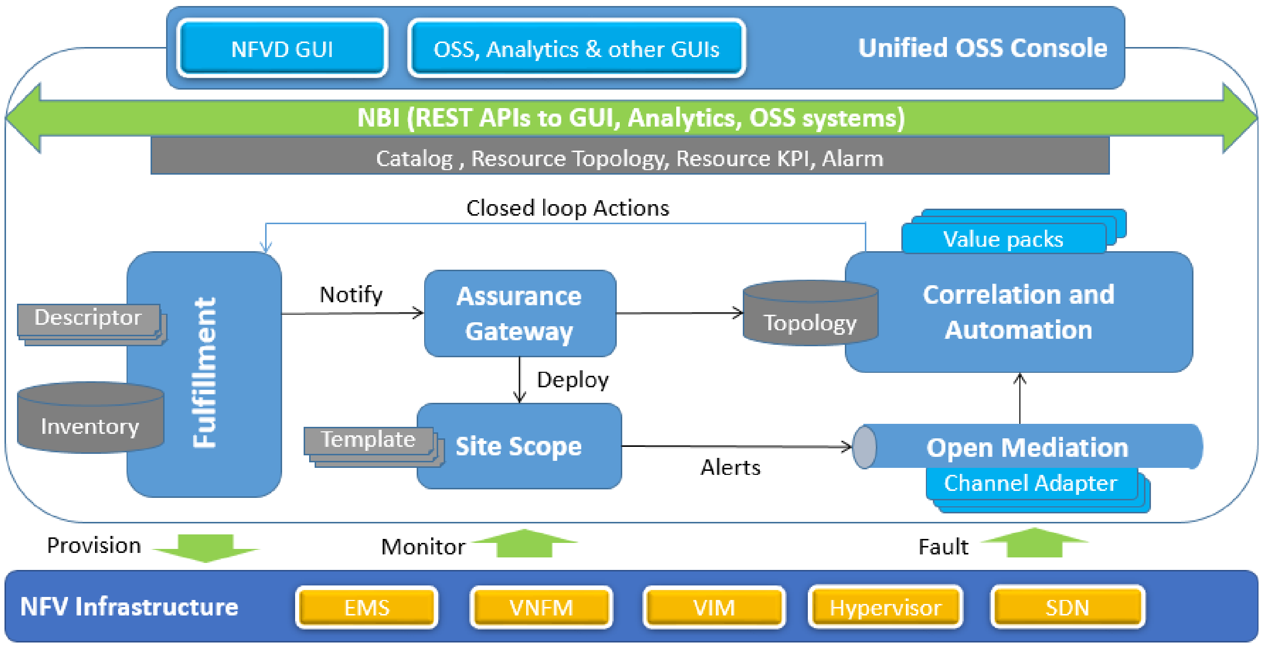


Figure 10 Interaction between HP NFV Director Components – Single Server

And the following picture shows interactions between components in a distributed environment:

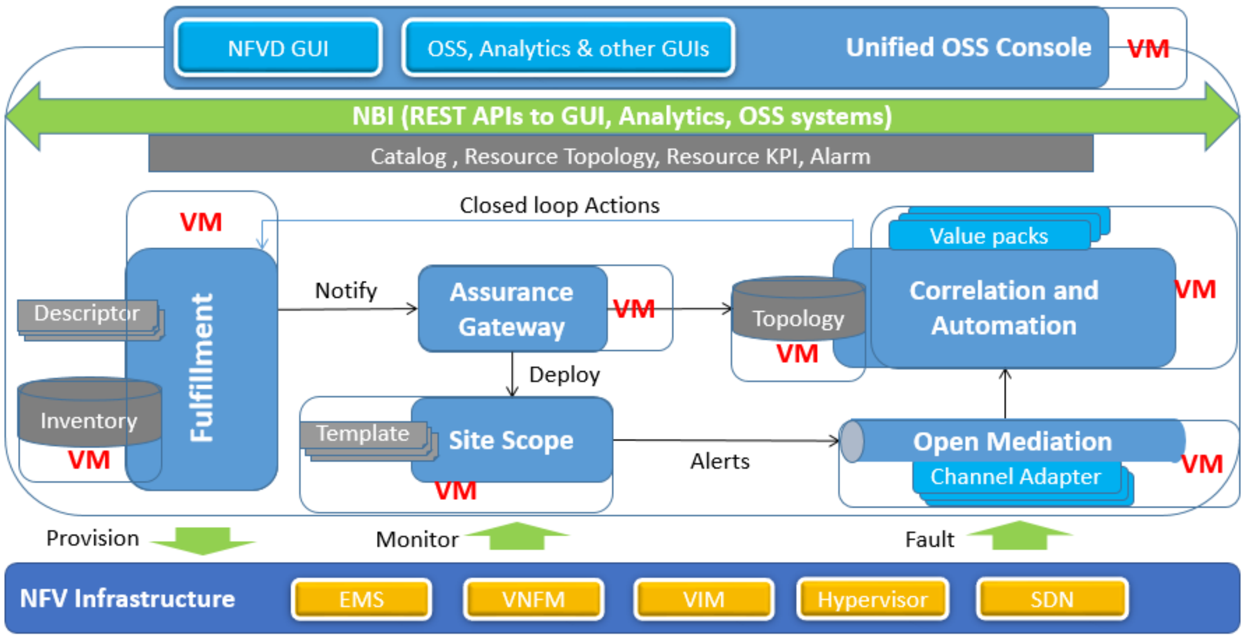


Figure 11 Interaction between HP NFV Director Components – Distributed Server

- On instantiation of VNF, the fulfillment component processes the VNF descriptor, applies policies and creates the required VNF and its subcomponents (for example, virtual machines) using Virtual Infrastructure Manager (VIM).
- Fulfillment also sends topology information of the created VNF to the monitoring component.

- The monitoring component processes this information and stores the topology information in the topology database for later correlation of events.
- At the behest of the fulfillment component the monitoring component deploys and activates the monitors associated to a VNF, its subcomponents, or both (for example, VM).
- The monitoring component continuously monitors the key performance indicators of the VNFs and upon threshold violation sends an event notification to the correlation component.
- The correlation component with the help of topology database correlates all the incoming events and takes the necessary action. For example, ScaleIn or ScaleOut of the VNF resources.

Another representation of NFV Director components and their function planes is shown next figure:

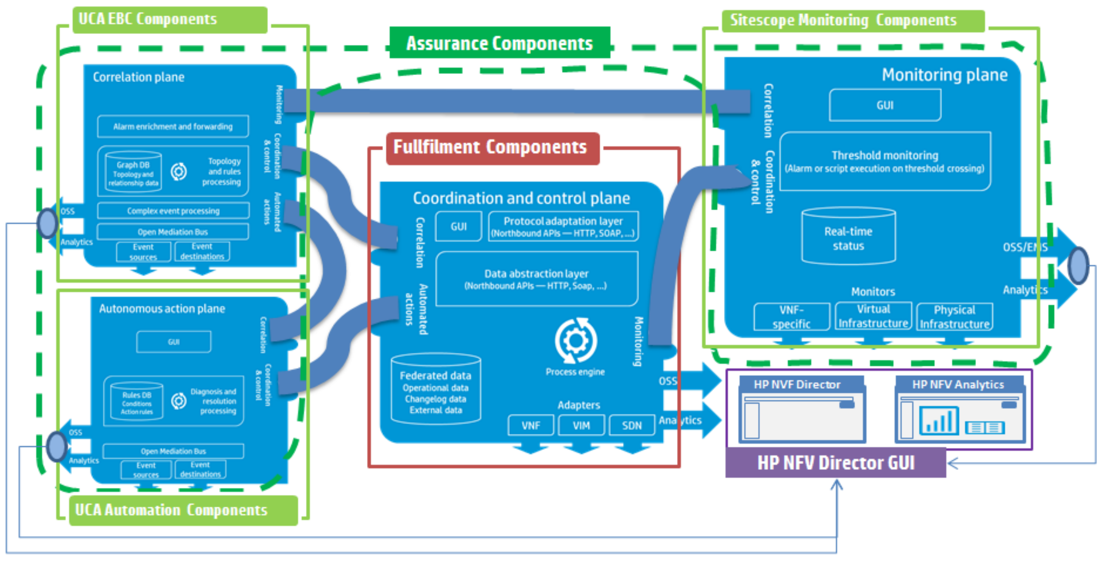


Figure 12 Architecture components from functional plane viewpoint

1.4 NFVD customization and integration points

The NFV Director is highly customizable and extensible to incorporate the type of VNF. It also provides different integration points to integrate with any external component.

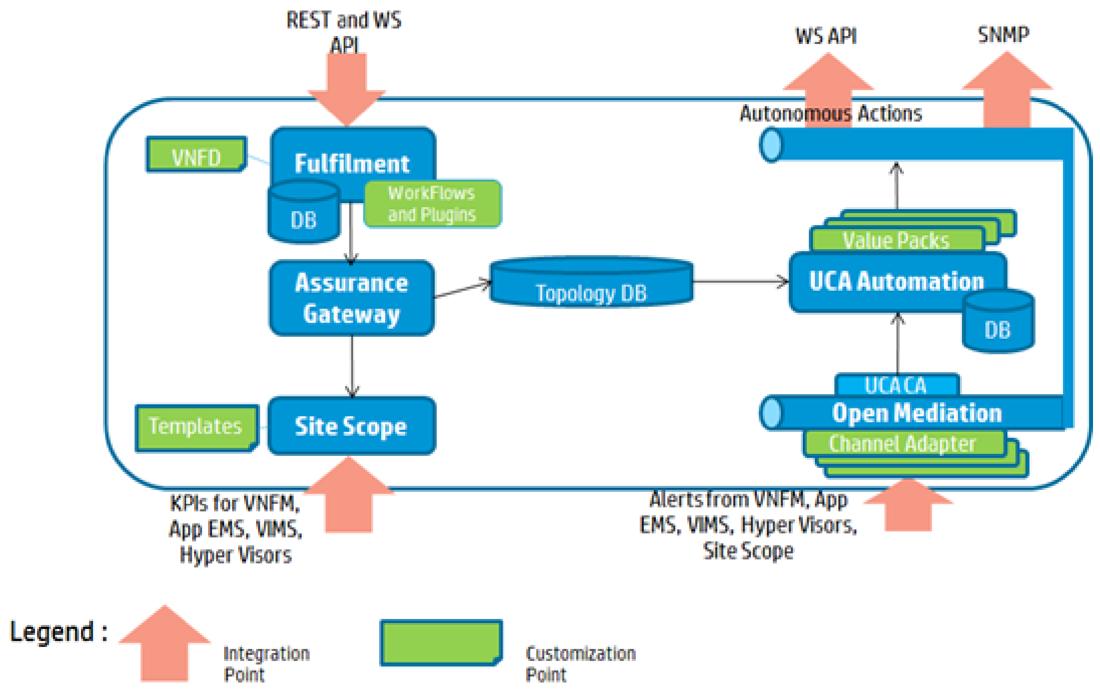


Figure 13 Customization and Integration points for NFV Director

1.4.1 Integration points

1.4.1.1 Southbound Integration

- NFV Director provides REST interface for VNF management.
- Read the *HP NFV Director documents* for more detailed information.
- The NFV Director also provides basic OpenStack Rest based API.
- The NFV Director is extensible and any new interface can be developed on demand to integrate with external components and other products.

1.4.1.2 Northbound Integration

NFV Director can integrate events from various sources such as VNFM, EMS, hypervisors, SiteScope, and external applications.

By default, NFV Director provides integration of events from the monitoring component with SiteScope.

1.4.2 Customization

1.4.2.1 VNFD

HP NFV Director provides an open-XML based extensible data model to model, deploy, and instantiate any type of VNF.

1.4.2.2 Workflows and plugins

HP NFV Director provides a framework to write new workflows to deploy, activate and configure any type of VNF and its constituent sub-components.

Using plugins, HP NFV Director is extensible to interface with different kinds of Virtual Infrastructure Managers.

1.4.2.3 Monitoring templates

For effective monitoring of any NFV resources, HP NFV Director provides a framework to develop and integrate new customized monitoring templates. New monitoring templates are developed using SiteScope SDK.

1.4.2.4 Open Mediation channel adaptors

The OpenMediation channel adaptors provide a flexible way to integrate events from different sources of NFV ecosystem for correlation.

1.4.2.5 Value Packs

HP NFV Director provides capability to create comprehensive functional correlation sets called Value packs that implement the correlation logic that can take necessary actions.

Note

See Chapter 5 and Chapter 6 for more information on Customization of NFV Director components.

Chapter 2

NFV Modeling

2.1 Generic NFV data model

HP NFV Director provides a generic approach to model any VNF or NS.

In order to model NS, VNF, or both and its subcomponents, HP NFV Director provides a XML based modeling.

Basically NFV Director provides:

- ARTIFACT
  - An artifact is an entity of any type.
  - An artifact can have any number of attributes.
- RELATIONSHIP
  - A relationship is a parent child relationship from one artifact to another.
  - A relationship can have any number of attributes as well.

For example, if you want to model a server having a card, and card has a port, then by using the ARTIFACT and RELATIONSHIP we can model the following example.

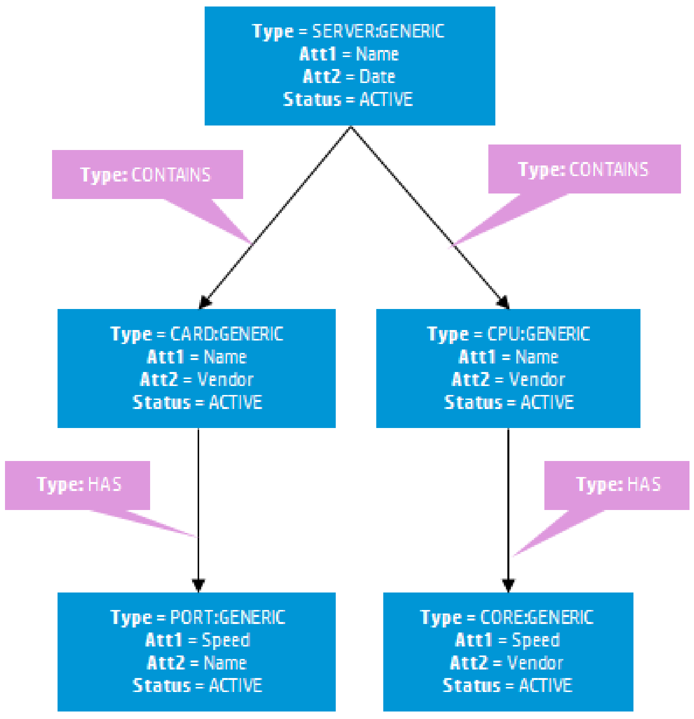


Figure 14 Generic Model Example

The advantages of this NFV Director generic modeling is:

- Flexibility

  - Anything can be modeled.
  - Modifications on the model can be done on real time.
  - Modifications on the model can be done from GUI without compiling or restarting.
- Speed
  - Tables, beans and JSP are already there out of the box.
  - Reduces the number of tables.
  - Logic, queries, or visualizations are much more reusable as the table structure does not change from project to project.
  - Optimization only needs to be done for a fixed set of tables.
- Re-use
  - Allows the copy and paste of data, definitions or templates in an independent way.
  - Is language independent.
  - Artifacts can be used in different places for different purposes.
  - Out of the box import and export tools allow recreating definitions, templates or data independently.

Three different flavors of artifacts and relationships:

- Definitions (types)

The possible types of artifacts and their attributes along with the possible types of relationships and their attributes are defined in the Definition tables. The artifact that can be a parent of another is also defined here.

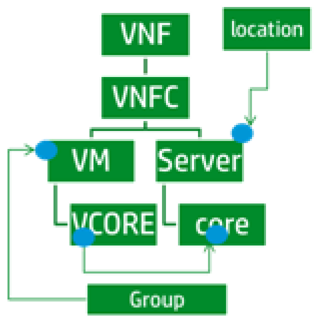


Figure 15 Artifact and Relationship Definition

- Template (catalog)

A template is an almost finished instance, that may have policies and ruling to fix behavior while creating an instance based on a template.

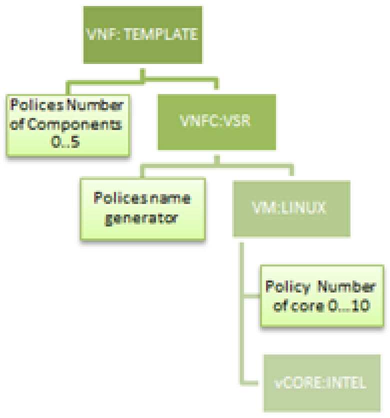

VNF: TEMPLATE
Polices Number of Components 0..5
VNFC:VSR
Policesname generator
VM:LINUX
Policy Number of core 0...10
vCORE:INTEL

Figure 16 Artifact and Relationship Template

- Instance (data)

Each definition can have any number of instances with different or equal values for each attribute.

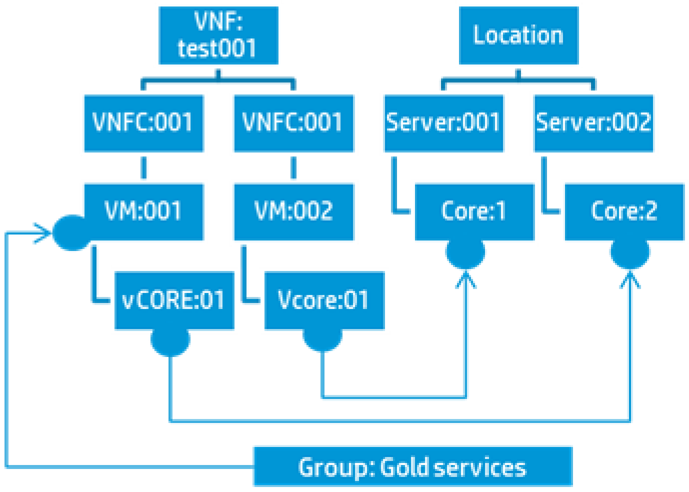


Figure 17 Artifact and Relationship template

You can get more info about NFV Modeling in *HP NFV Director v3.0 Data Model Guide* document.

2.2 NorthBound API Provided by HP NFV Director

The complete list of the automation processes provided by HP NFV Director is shown in the next table:

	Service Order Name	Service Order Service	Service Order Operation
1	NFVD	ASSIGNMENT	GREATER_CAPACITY
2	NFVD	INVENTORY	CREATE_FROM_TEMPLATE
3	NFVD	INVENTORY	SCALE_DOWN
4	NFVD	INVENTORY	SCALE_IN
5	NFVD	MONITOR	DEPLOY

6	NFVD	MONITOR	START
7	NFVD	MONITOR	STOP
8	NFVD	MONITOR	UNDEPLOY
9	NFVD	NETWORK_SERVICE	CREATE_AND_CONNECT
10	NFVD	NETWORK_SERVICE	SCALE_DOWN
11	NFVD	NETWORK_SERVICE	SCALE_IN
12	NFVD	NETWORK_SERVICE	SCALE_OUT_AND_CONNECT
13	NFVD	NETWORK_SERVICE	SCALE_UP
14	NFVD	NS	CREATE_AND_CONNECT
15	NFVD	NS	SCALE_IN
16	NFVD	NS	SCALE_OUT_AND_CONNECT
17	NFVD	VIRTUAL_CORE	SCALE_DOWN
18	NFVD	VIRTUAL_CORE	SCALE_UP
19	NFVD	VIRTUAL_MEMORY	SCALE_DOWN
20	NFVD	VIRTUAL_MEMORY	SCALE_UP
21	NFVD	VNF	ACTIVATE
22	NFVD	VNF	CREATE
23	NFVD	VNF	CREATE_AND_CONNECT
24	NFVD	VNF	CREATE_WITHOUT_ACTIVATE
25	NFVD	VNF	DEACTIVATE
26	NFVD	VNF	DELETE
27	NFVD	VNF	DELETE_INSTANCE_TREE
28	NFVD	VNF	PREPARE_ACTIVATION
29	NFVD	VNF	PROCESS_MONITORS
30	NFVD	VNF	SCALE_DOWN
31	NFVD	VNF	SCALE_IN
32	NFVD	VNF	SCALE_OUT_AND_CONNECT
33	NFVD	VNF	SCALE_UP
34	NFVD	VNF	START_VM
35	NFVD	VNF	STOP_VM
36	NFVD	VNFMANAGER	ASSIGNMENT
37	NFVD	VNFMANAGER	CHANGE_STATUS_BY_MANAGER
38	NFVD	VNFMANAGER	CREATE_AND_DEPLOY_MANAGER
39	NFVD	VNFMANAGER	CREATE_INSTANCE_FROM_DESCRIPTOR
40	NFVD	VNFMANAGER	CREATE_VNF_THROUGH_MANAGER
41	NFVD	VNFMANAGER	DELETE_VNF
42	NFVD	VNFMANAGER	DO_NOTHING
43	NFVD	VNFMANAGER	GET_MANAGER_JOB_STATUS
44	NFVD	VNFMANAGER	GET_MANAGER_VNF_DETAILS
45	NFVD	VNFMANAGER	GET_VNF_GRANT
46	NFVD	VNFMANAGER	REGISTER_MANAGER

Table 2 Automation processes

You can get more information about Northbound Interfaces in *HP NFV Director Interfaces. Northbound Guide* document.

Chapter 3

VNF on-boarding

Note

For more comprehensive details on VNF on-boarding, refer to the *HP NFV Director VNF On-boarding Guide*.

The VNF on-boarding process is defined in this document as the process that allows the deployment of VNF in NFV Director.

The process is composed of the following high level steps:

1. Create a VNF descriptor (formally a template in NFV director product).
2. Create all the necessary monitor templates.
3. Define all the necessary actions (like scale in/out).
4. Configure the resource data (available resources, hypervisors and VIMS).
5. Load/Configure all the necessary resources on the available VIMs or hypervisors (like images referenced on the template, the networks described on the template, etc.).

Once the previous steps are completed, it is possible in one operation to:

1. Create an instance of the VNF (VNF model representation at database level).
2. Assign/Allocate the selected resources to a target resource group (for example, Vcores of the VMs of the VNF to Cores of servers of a datacenter).
3. Create/activate those virtual machines on a target VIM or hypervisor.

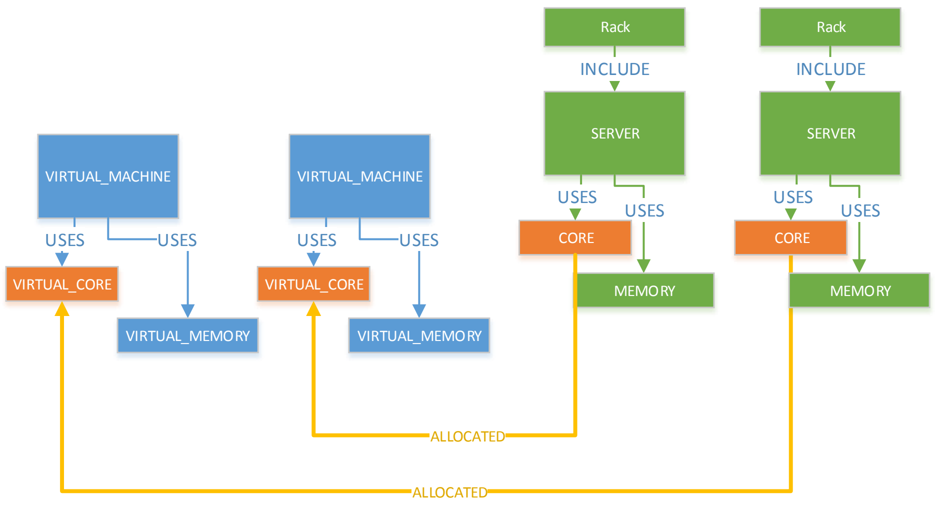


Figure 18 Virtual machine Vcores assigned (allocated) to Cores of Servers

3.1 VNF descriptor

A VNF descriptor is modeled like a template.

A template is an almost finished instance. Policies and rules are fixed when an instance is created based on a template.

The needed information in a VNF descriptor is related to:

- General info
    - VNF descriptor file with all the information, if not at very least all the information required (the information is needed even if we receive the descriptor from them)
    - VM template (at very least VMDK files for all machines)
    - API documentation
- VNF info
    - Number of VM types
    - If the VMs will be using vlans or just ip address
    - Type of VM (linux /windows) users and password for it apis or interfaces (ssh, telnet, webservice , etc …)
    - Flavours allowed for each vm (range of ram, disk, cores for each vm)
    - KPIs to monitor and scripts to so if those are inside the VM
- Networking info
    - Rules of how many VMs of each type it should be (min max)
    - Rules about if the vms can be together or not
    - How the VMs connect between them
    - If any IP is needed or DHCP can be used

3.1.1 Virtual Network Function model

The VNF model is composed of a set of definitions of artifacts and their relationships.

A high level VNF Model is described in the following illustration.

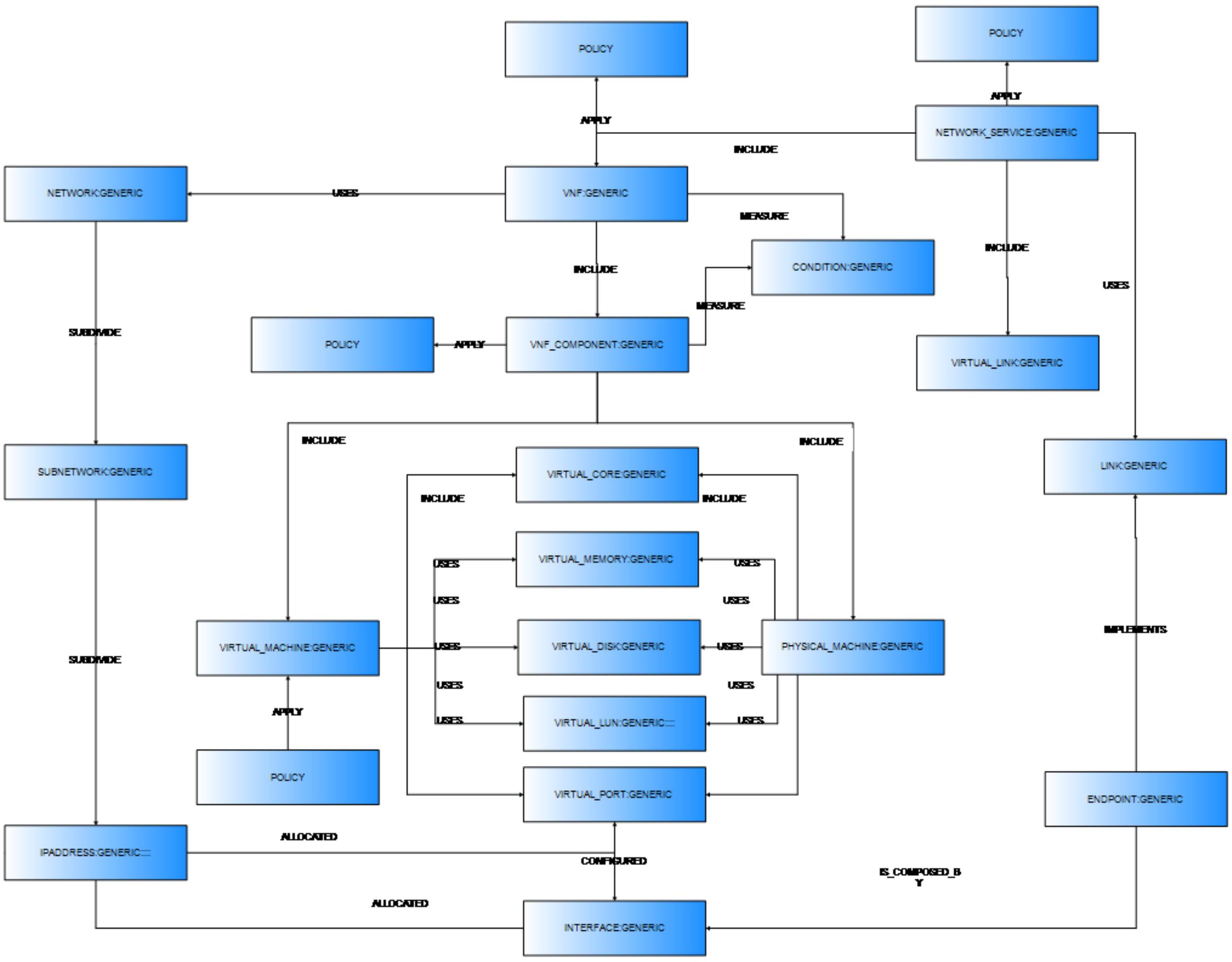


Figure 19 VNF Model

3.1.1.1 Template example with range policy

Templates

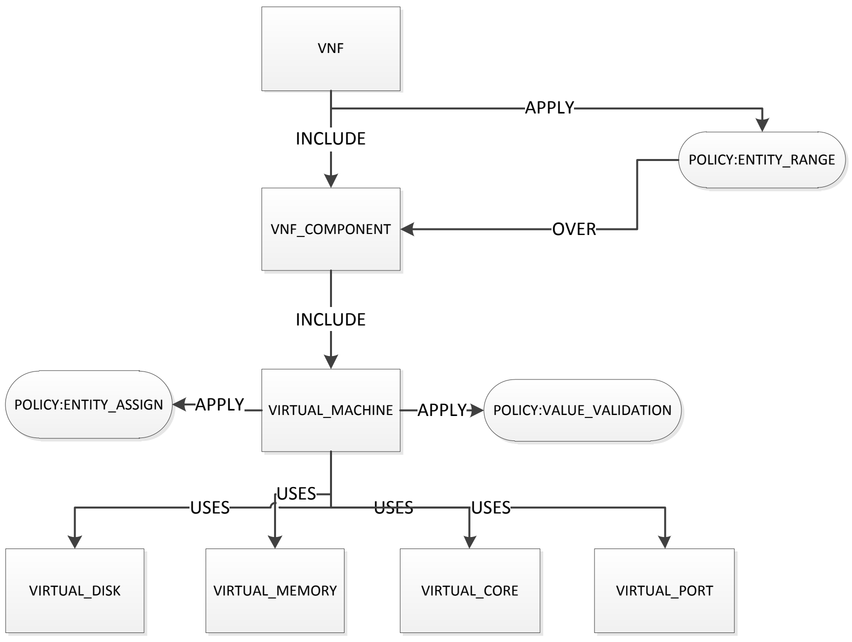


Figure 20 Template example with range policy

Instances

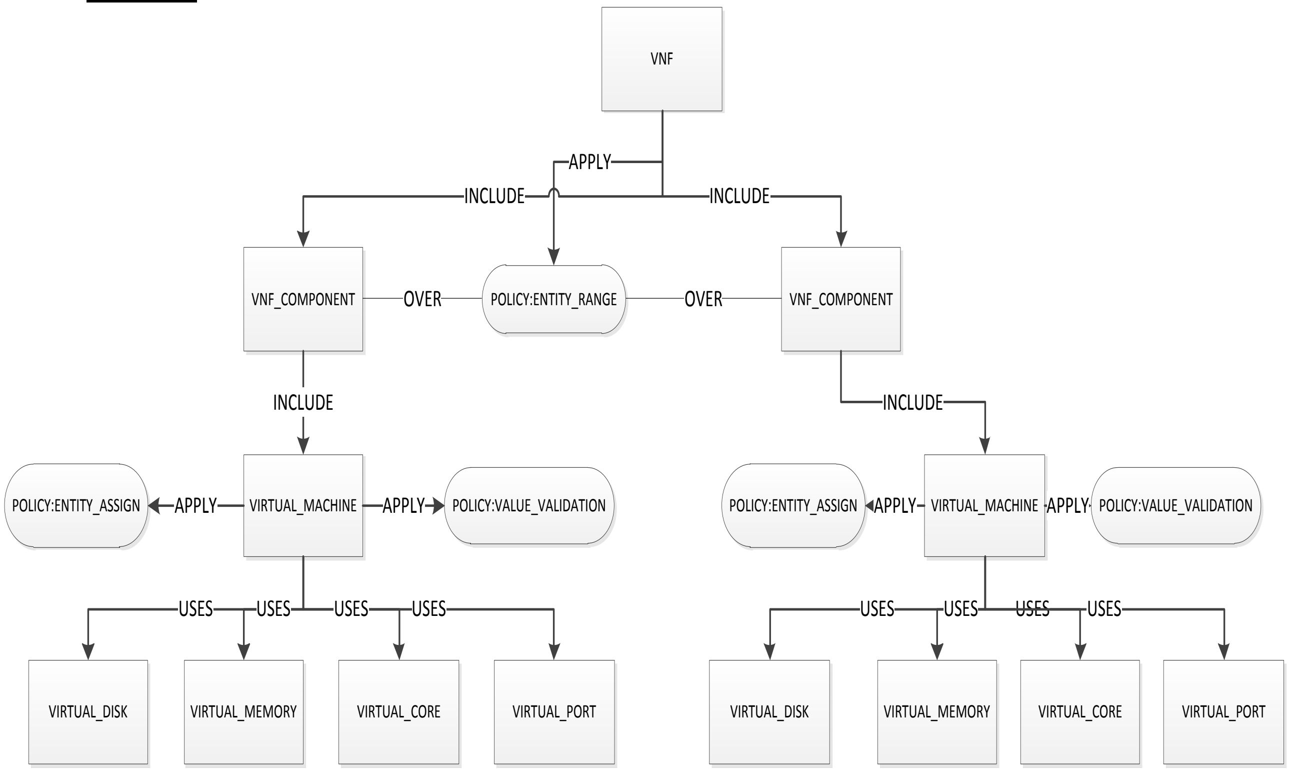


Figure 21 Instance result of template example with range policy

Performing the operation in the VNF with the parameters set on this way:

DEFAULT= 1, MAX= 5,

We obtain a VNF with 2 VNF_COMPONENT as children of VNF.

If the VNF_COMPONENT does not have the POLICY:ENTITY_RANGE, we obtain the same instances of the trees that are available in templates.

Note that, you can only create an instance, if the validation is correct.

3.1.1.2 Template example with assign policy

Templates

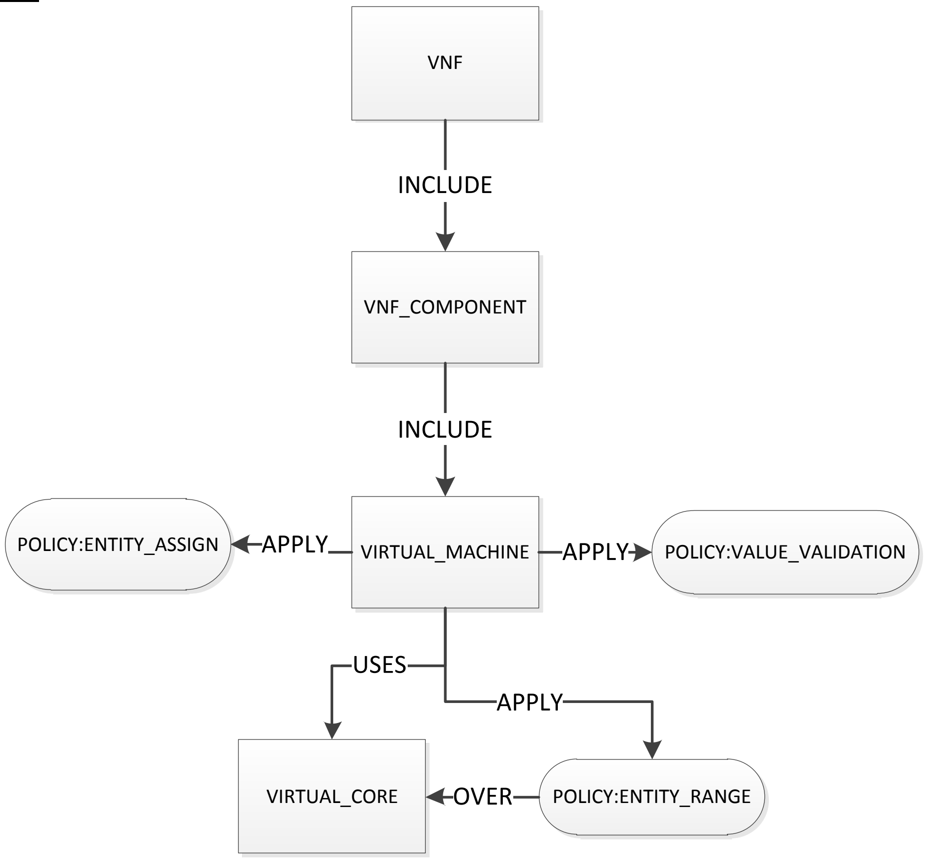


Figure 22 Template example with assign policy

Instances

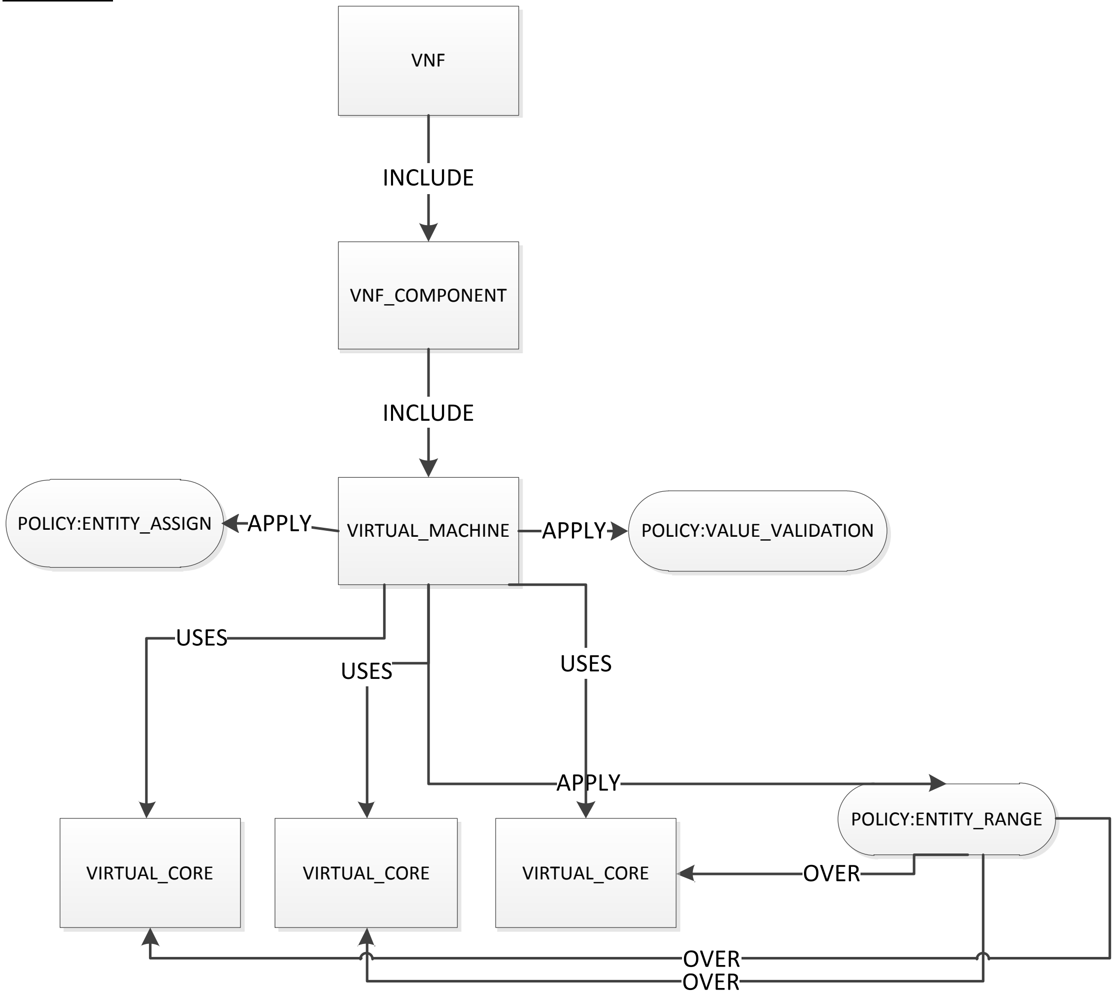


Figure 23 Instance result of Template example with assign policy

In this case, the policy creates two VIRTUAL_COREs instead of four, because the maximum is three.

3.1.2 Resource/HPSA template definition

The NFV Director comes with an out of the box resource model, that allows the modeling of the physical infrastructure.

3.1.2.1 Resources model

The resources model is composed of a set of definitions of artifacts and their relationships.

A high level resource model is described in the following illustration.

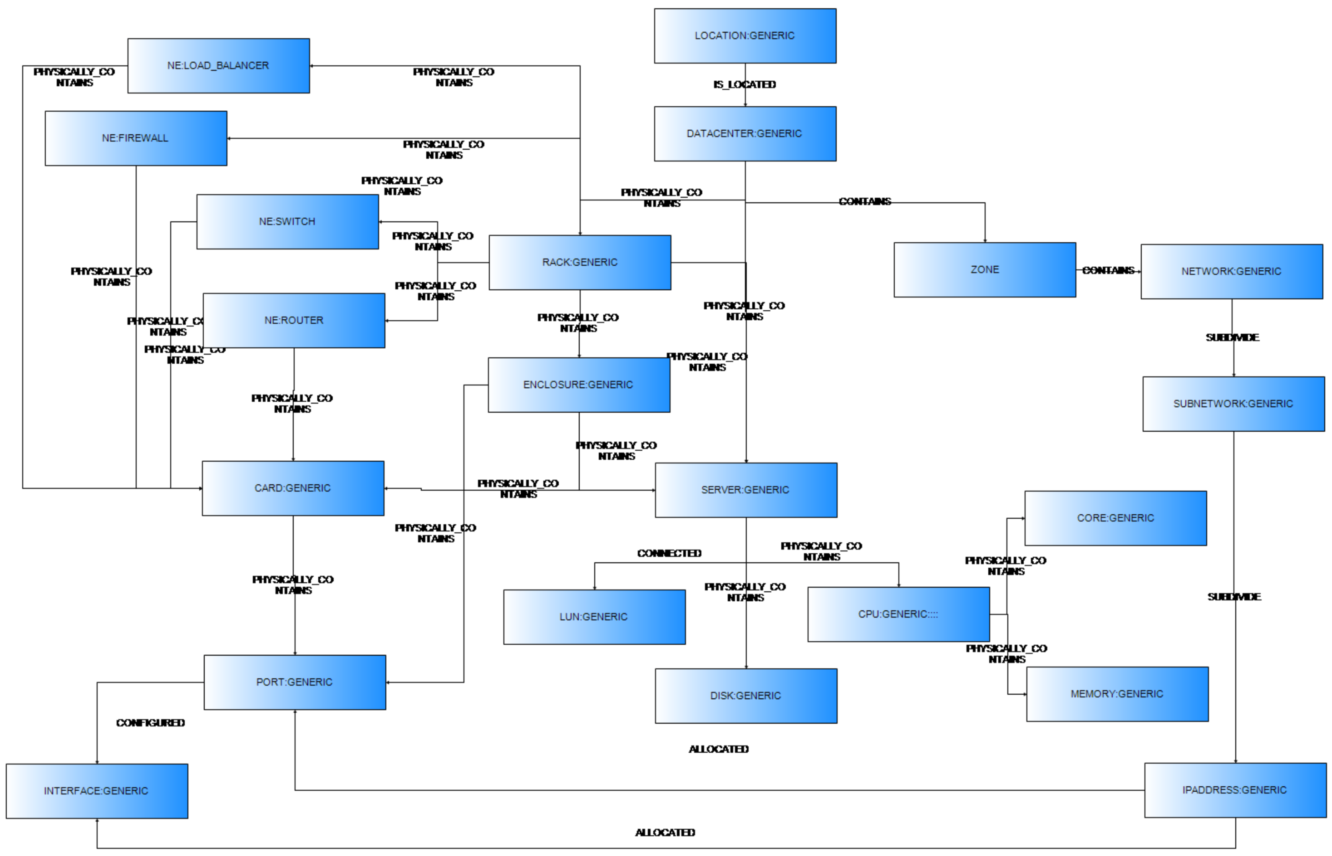


Figure 24 Resources Model

3.1.3 Monitoring definition

The NFV Director comes with and out of the box resource model that allows modeling the monitoring.

3.1.3.1 Monitoring model

The monitoring model is composed of a set of definitions of artifacts and their relationships.

A high level resource model is described in the following illustration.

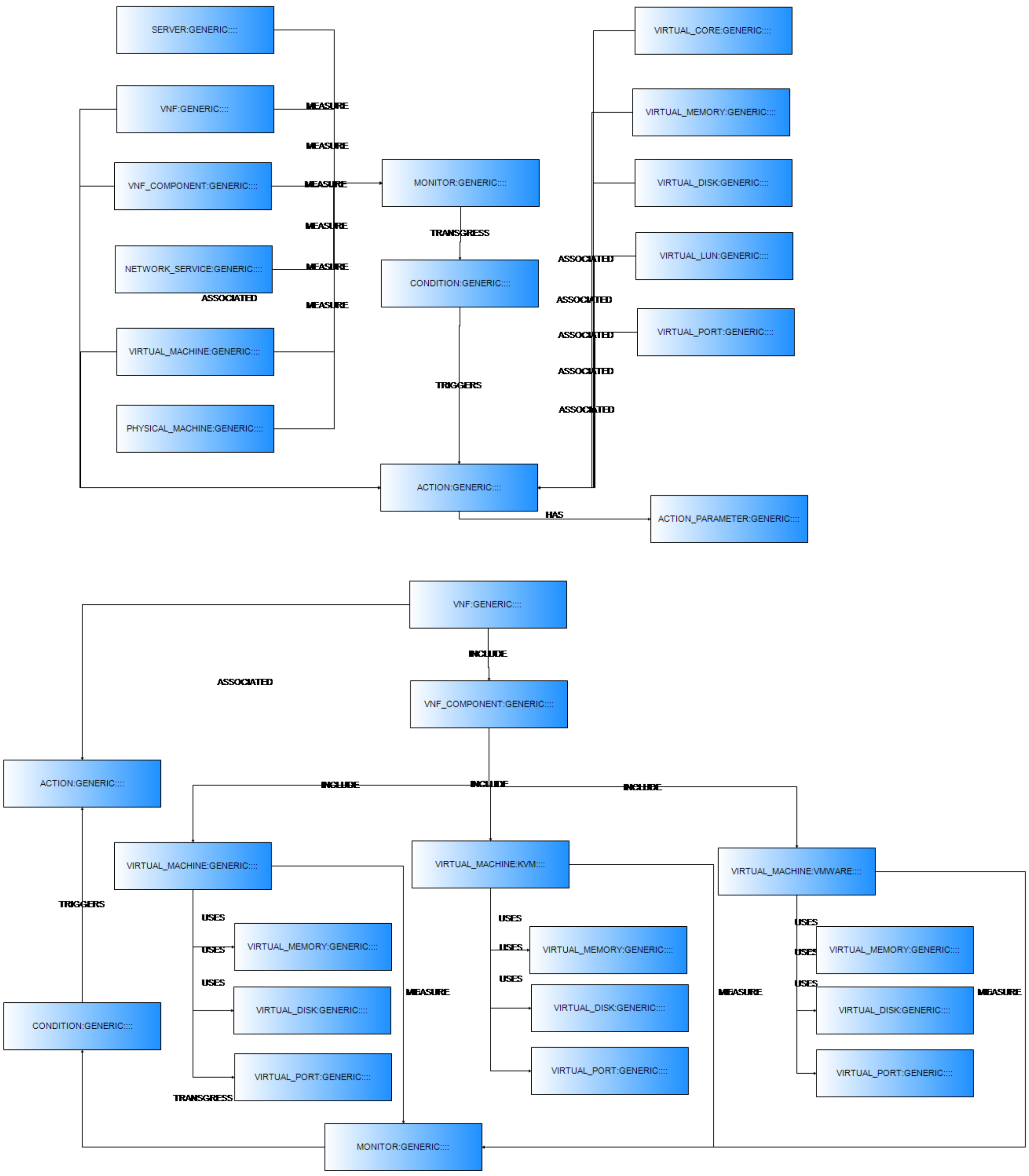


Figure 25 Monitoring Model

3.1.4 Monitoring modeling

The purpose of monitoring the model is to model a monitor for a VNF template that:

- Monitors the key performance indicators for given NFV resources.
- Generate threshold controlling alerts for KPI breaches.

These monitors represent the monitors that get deployed using SiteScope.

Once the Resource Tree, Assignment Tree, and VNF template is ready, the next thing is to add monitoring.

Artifacts: In general, the monitoring model is as follows (highlighted in blue).

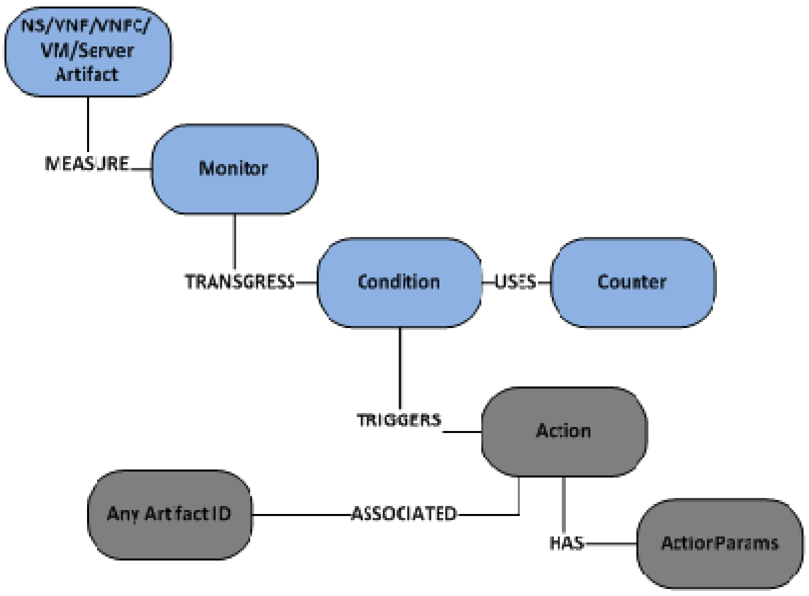


Figure 26 Monitoring Model

A monitor can be associated at various levels.

- A monitor can be associated at Virtual Machine or Physical host level.
- A monitor can be associated at VNFC level.
- A monitor can be associated at VNF level.
- A monitor can be associated at NS level.
-

The monitor artifact tells what to monitor, and a monitor handler artifact tells how, and from where we can get the required key performance indicators for that monitor.

3.2 Description of monitoring model artifact

3.2.1 Monitor artifact

A monitor artifact takes the following attributes.

Attribute	Description	Constraints
Name	Name of the monitor	The name should be same as that of SiteScope Monitor template Mandatory: Yes.
Description	Human readable description of the monitor	Mandatory : No
Frequency	Time value in seconds at which the monitor runs periodically.	Mandatory : Yes Should not be less than 20 seconds.
Deployment Path	The Path where the monitor should be deployed on SiteScope.	Mandatory: No. If this field is empty, a default path will be automatically built.
Template Path	The Path of SiteScope Monitor template from where it is to be deployed.	Generically used with Custom Monitors. Mandatory: Yes, only when custom monitors are used.

Table 3 Monitor artifact attributes

3.2.2 Condition artifact

Attribute	Description	Constraints
Name	The name of the condition.	Mandatory: Yes.
Type	Type of Condition	Available values are: • Good • Warning • Error
Expression	An expression that describes a condition. See Out of Box monitors provided with NFV Director to provide a valid and supported expression.	Mandatory: Yes, only when out of box NFV Director monitors are used. Optional when custom monitors are used.

Table 4 Condition artifact attributes

3.2.3 Counter artifact

Attribute	Description	Constraints
Name	Name of the counter	Mandatory : No
Description	Human readable description	Mandatory : No

Table 5 Counter artifact attributes

3.2.4 Credentials for Virtual Machine:

Credentials for Virtual machine is dynamically picked up from one of the below components.

a. Authentication

b. VIM

c. Hypervisor

DEPLOYMENT.Type in Monitor:Generic artifact can be set as:

a. AUTO – credentials will be picked up from Authentication if ceilometer is supported by VIM. If Authentication is not present VIM will be used. If VIM is not present then Hypervisor would be used to pick up credentials.

b. VIM – credentials will be picked up from VIM

c. HYPERVISOR – Credentials will be picked up from HYPERVISOR

MonitorArgument can be associated to Monitors. MonitorArgument are of two types:

a. Arguments – it contains name value pairs, name has to map to mandatory variables required by template in sitescope

b. CYPHER – in value field cypher query can be provided to pick up the mandatory variables from neo4j. Ex: "START n=node(*) match n<-[r*]-reg<-[r*]-auth where has(n.artifactId) and n.artifactId='MONITOR_ID' and has(reg.artifactFamily) and reg.artifactFamily='REGION' and has(auth.artifactFamily) and auth.artifactFamily='AUTHENTICATION' RETURN reg.`GENERAL.Name` as region,auth.`CREDENTIALS.Login` as user, auth.`CREDENTIALS.Password` as password, auth.`CREDENTIALS.Url` as keystoneURL, auth.`CREDENTIALS.TenantName` as tenantName" .

Here alias like tenantName, keystoneURL has to be given that matches to the mandatory varialbes in sitescope templates. There will be a place holder for MONITOR_ID which would be replaced by the actual monitor artifact Id to which the monitor argument is associated to. Make sure the cypher query provided as part of the value field escapes xml characters, i.e. below mentioned characters needs to be escaped.

"	&quot;
'	&apos;
<	&lt;
>	&gt;
&	&amp;

Only get queries are supported here, create and update query will not work and would be ignored.

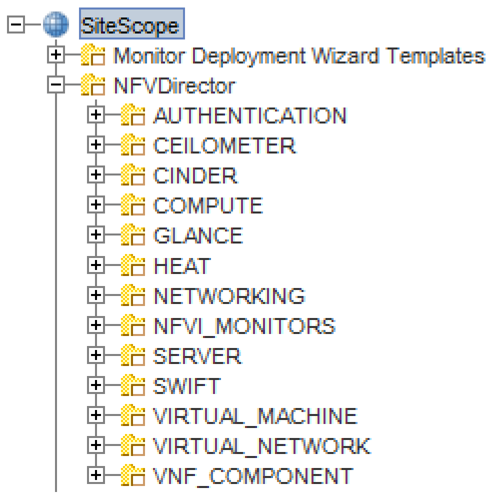
3.3 Northbound API for monitoring

The following operations are associated with a Monitor:

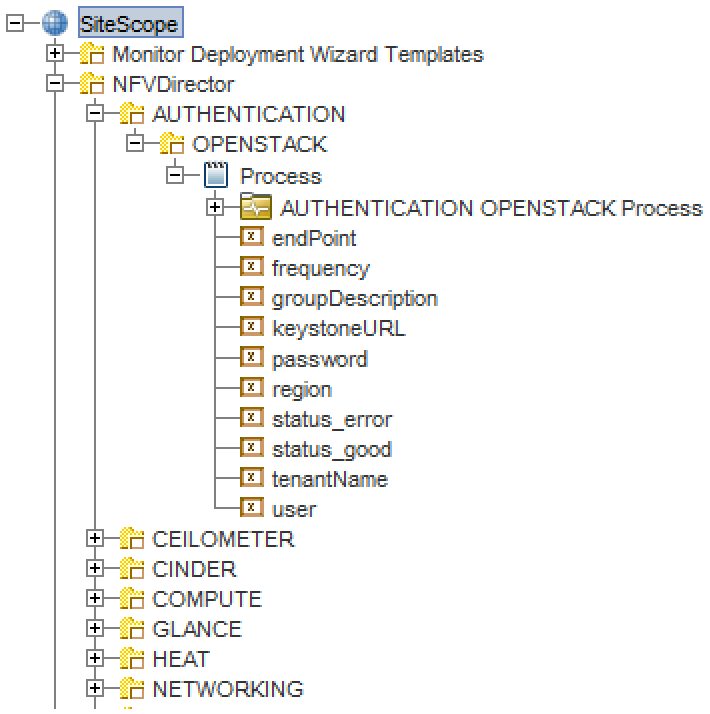
- Deploy a monitor
- Start a Monitor
- Stop a Monitor
- UnDeploy a Monitor

3.4 Out of box monitors provided by NFV Director

Below is the list of Monitors supported out of the box.


SiteScope
Monitor Deployment Wizard Templates
NFVDirector
AUTHENTICATION
CEILOMETER
CINDER
COMPUTE
GLANCE
HEAT
NETWORKING
NFVI_MONITORS
SERVER
SWIFT
VIRTUAL_MACHINE
VIRTUAL_NETWORK
VNF_COMPONENT

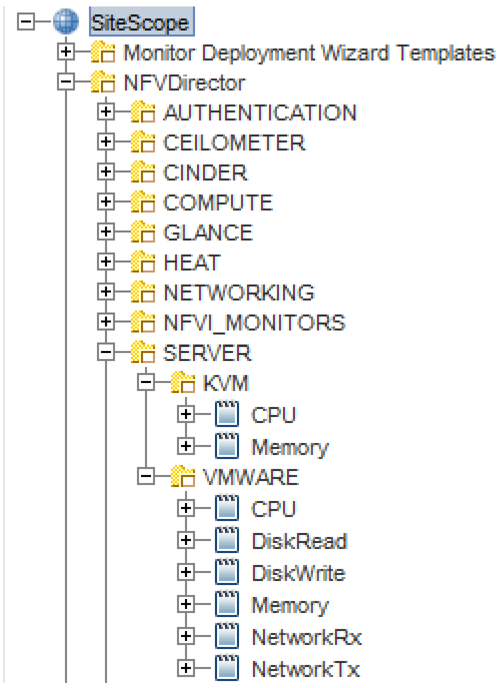
Process monitors for OpenStack services like authentication (i.e. Keystone), ceilometer, cinder, swift, heat, glance, networking (i.e. neutron),:
Counters supported here is status, i.e. only status can be used in condition expression. Only ERROR and GOOD condition is supported. Status for GOOD will be set to 0 and status for ERROR will be set to non zero



- SiteScope
  - Monitor Deployment Wizard Templates
  - NFVDirector
    - AUTHENTICATION
      - OPENSTACK
        - Process
          - AUTHENTICATION OPENSTACK Process
          - endPoint
          - frequency
          - groupDescription
          - keystoneURL
          - password
          - region
          - status_error
          - status_good
          - tenantName
          - user
    - CEILOMETER
    - CINDER
    - COMPUTE
    - GLANCE
    - HEAT
    - NETWORKING

Physical Sever monitors:
Counters:

CPU: cpu_usage_average
Memory: memory_usage_average
DiskRead:disk_read_requests
DiskWrite:disk_write_requests
NetworkRx:network_bytes_received
NetworkTx:network_bytes_transmitted

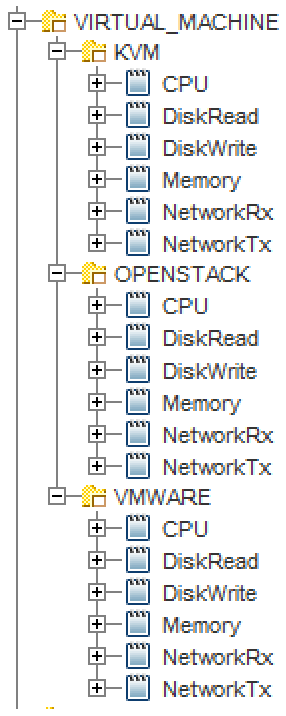


- SiteScope
  - Monitor Deployment Wizard Templates
  - NFVDirector
    - AUTHENTICATION
    - CEILOMETER
    - CINDER
    - COMPUTE
    - GLANCE
    - HEAT
    - NETWORKING
    - NFVI_MONITORS
    - SERVER
      - KVM
        - CPU
        - Memory
      - VMWARE
        - CPU
        - DiskRead
        - DiskWrite
        - Memory
        - NetworkRx
        - NetworkTx

VirtualMachine monitors: VMWARE monitor also supports VCENTER.
Counters:

CPU: cpu_usage_average
Memory: memory_usage_average

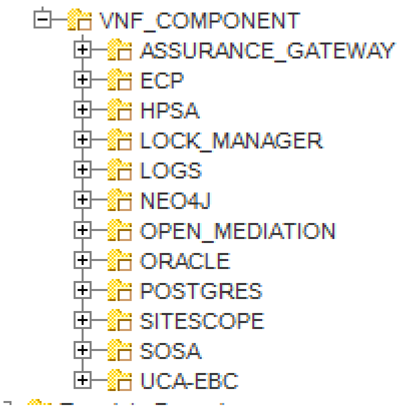
DiskRead:disk_read_requests
DiskWrite:disk_write_requests
NetworkRx:network_bytes_received
NetworkTx:network_bytes_transmitted


```
VIRTUAL_MACHINE
  KVM
    CPU
    DiskRead
    DiskWrite
    Memory
    NetworkRx
    NetworkTx
  OPENSTACK
    CPU
    DiskRead
    DiskWrite
    Memory
    NetworkRx
    NetworkTx
  VMWARE
    CPU
    DiskRead
    DiskWrite
    Memory
    NetworkRx
    NetworkTx
```

SelfMonitors: Process and Log monitors are supported.

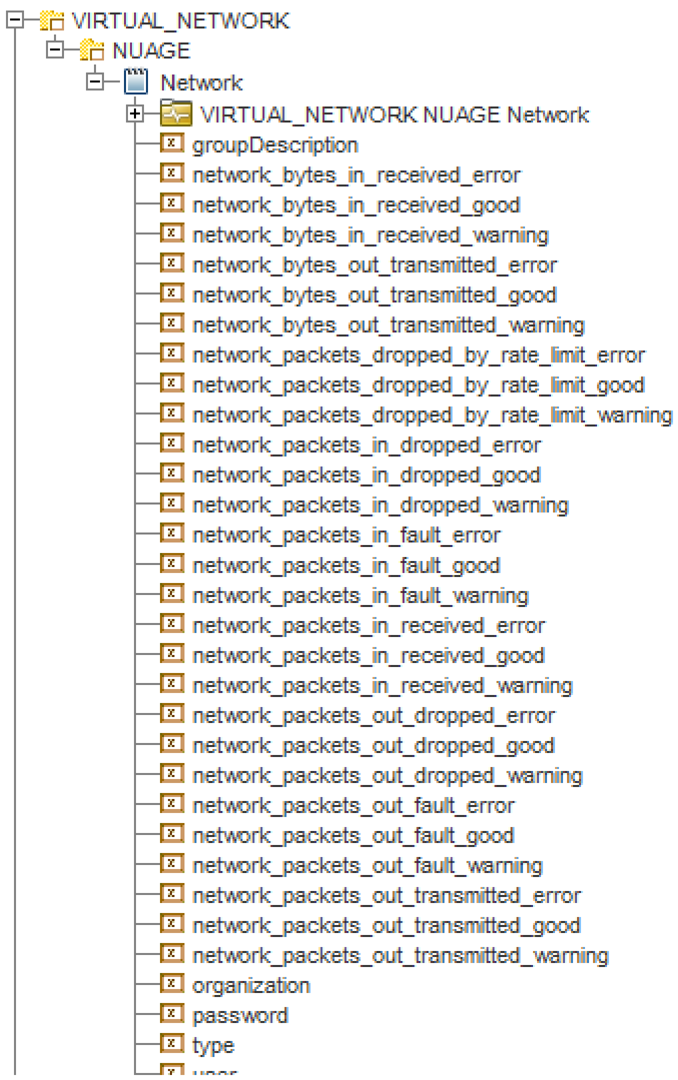
Counters: status: Value of status for error condition is <0.

Counters: pid : pid < 0 is treated as error, and pid >= 0 is good condition


```
VNF_COMPONENT
  ASSURANCE_GATEWAY
  ECP
  HPSA
  LOCK_MANAGER
  LOGS
  NEO4J
  OPEN_MEDIATION
  ORACLE
  POSTGRES
  SITESCOPE
  SOSA
  UCA-EBC
```

VirtualNetwork for nuage: Here we can see that multiple counters are supported.

Counters: network_bytes_in_received, network_bytes_out_transmitted, network_packets_dropped_by_rate_limit, network_packets_in_dropped, network_packets_in_fault, network_packets_in_received, network_packets_out_dropped, network_packets_out_fault, network_packets_out_transmitted



```
VIRTUAL_NETWORK
  NUAGE
    Network
      VIRTUAL_NETWORK NUAGE Network
      groupDescription
      network_bytes_in_received_error
      network_bytes_in_received_good
      network_bytes_in_received_warning
      network_bytes_out_transmitted_error
      network_bytes_out_transmitted_good
      network_bytes_out_transmitted_warning
      network_packets_dropped_by_rate_limit_error
      network_packets_dropped_by_rate_limit_good
      network_packets_dropped_by_rate_limit_warning
      network_packets_in_dropped_error
      network_packets_in_dropped_good
      network_packets_in_dropped_warning
      network_packets_in_fault_error
      network_packets_in_fault_good
      network_packets_in_fault_warning
      network_packets_in_received_error
      network_packets_in_received_good
      network_packets_in_received_warning
      network_packets_out_dropped_error
      network_packets_out_dropped_good
      network_packets_out_dropped_warning
      network_packets_out_fault_error
      network_packets_out_fault_good
      network_packets_out_fault_warning
      network_packets_out_transmitted_error
      network_packets_out_transmitted_good
      network_packets_out_transmitted_warning
      organization
      password
      type
```

3.5 Action modeling

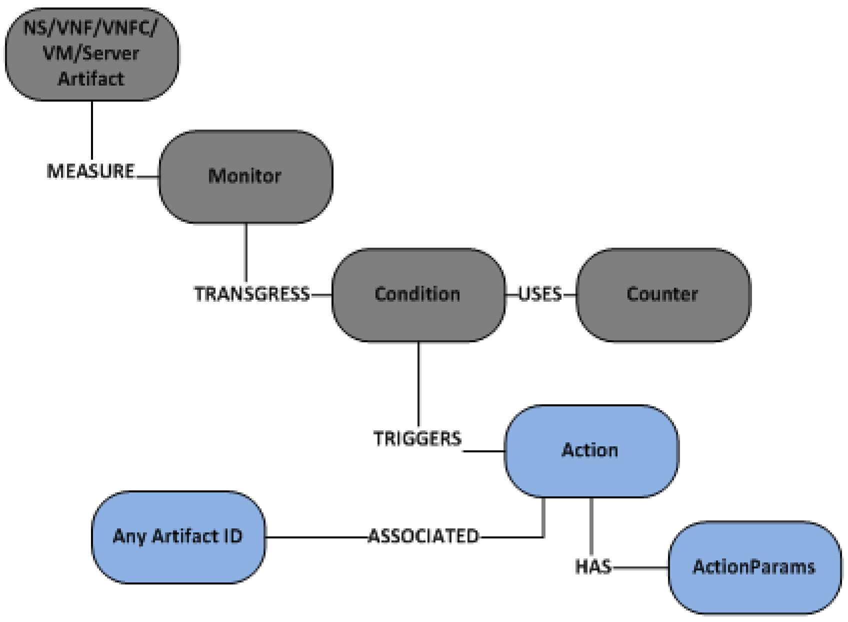


Figure 27 Action Modeling Artifacts

3.5.1 Action artifact

Attribute	Description	Mandatory
Name	A human readable name	Mandatory : Yes
Type	Type of Action	Five possible actions are supported: • Scale-In • Scale-Out • Scale-Up • Scale-Down • Script Mandatory: Yes
Description	A human readable test	Mandatory : No
Scale Value	The amount in integer to scale	Valid only when type is for Scale
Script Path	The script to execute	Valid only when Type is Script

Table 6 Action artifact attributes

3.5.2 Action parameters artifact

Attribute	Description	Mandatory
Name	A name	Mandatory, if Action Type is Script and the script takes arguments as input.
Type	Future Use	
Value	A value associated with the name	

Table 7 Action Parameters artifact attributes

3.5.3 Associated artifact ID for action

An action can be associated to any artifact in the NFV Model. Typically it is associated to:

- A Virtual Machine
- Resources of a Virtual Machine (CPU, DISK, Memory, and so on)
- VNFC Component
- VNF
- NS
- When a KPI breach is detected by the monitor, the action type and the associated artifact ID determines the action to be taken on the required artifact.

Example-1: Scale-Out

When there is KPI breach (say ERROR: high CPU usage) by the CPU monitor, and the action type is scale-out, and is associated to a VIRTUAL_MACHINE artifact, then NFV Director automatically triggers a scale-out action for the associated VIRTUAL_MACHINE.

Example-2: Script Action

When there is KPI breach (say WARNING: high DISK usage) by the DISK monitor, and the action type is ScriptAction, then NFV Director automatically executes the script specified in the action attribute ScriptPath by taking action parameters as input values to the script.

3.6 MultiSiteScope support:

With V3 now monitors can be deployed on more than one sitescope depending upon to which resource tree the monitor gets associated to.

a. Multiple VNFC:SITESCOPE can be created but only one of them can have IS_DEFAULT attribute set to true. The flag marks the VNFC:SITESCOPE as default sitescope which can be used for monitor deployment in case when no VNFC:SITESCOPE is attached to resource tree ex:DATACENTER etc. If the user has set IS_DEFAULT to true in multiple VNFC:SITESCOPE, then an error will be thrown back by assurance gateway as it will not be able to pick a VNFC:SITESCOPE uniquely.
b. If there are multiple VNFC:SITESCOPE created and associated to DATACENTER etc. Then VNFC:SITESCOPE having highest WEIGHTAGE (attribute in GENERAL category)

is used to decide which VNFC:SITESCOPE needs to be used for monitor deployment. If there are multiple VNFC:SITESCOPE artifacts having same value of weightage then one of them will be picked randomly

c. All Self monitors will always be deployed on default sitescope only. i.e all monitors configured under VNF:NFVD will be deployed in default sitescope only.

If you need to delete or update VNFC:SITESCOPE instance then make sure all the monitors on that sitescope are undeployed before delete/update operation. Please note that a delete/update operation on VNFC:SITESCOPE will only do the DB related operations it will not effect any changes on monitor.

3.7 Tagging of monitors in SiteScope

VNFC:SiteScope has an attribute SiteScopeTag under Category GENERAL. By default it has values DATACENTER, NETWORK_SERVICE. With this feature monitors associated for a particular DATACENTER/NETWORK_SERVICE can be grouped in sitescope . Ex: If there are two datacenters D1 and D2. If a monitor M1 gets created under D1 and monitor M2 gets created under D2. In Sitescope in multiview we can see a header D1 under that associated monitor M1.

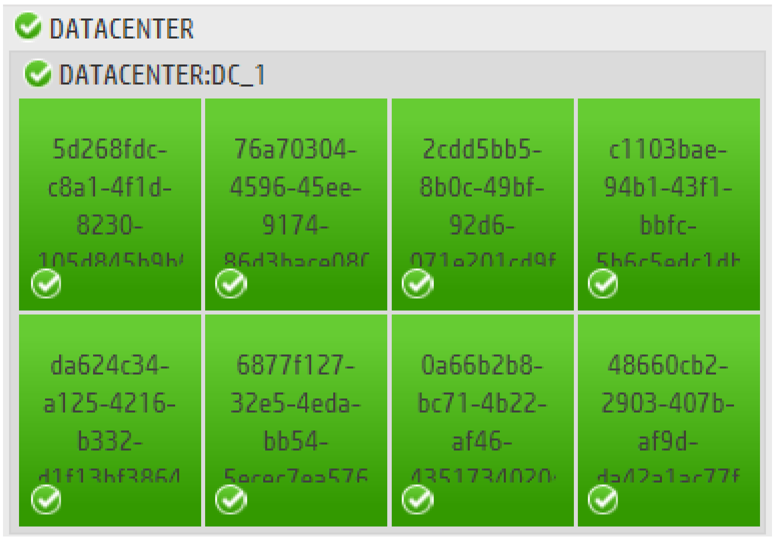


Figure 28 Monitor associated to Data Center

3.8 Integration between SiteScope alerts and UCA-EBC

The integration between SiteScope alerts and UCA-EBC is depicted as follows.

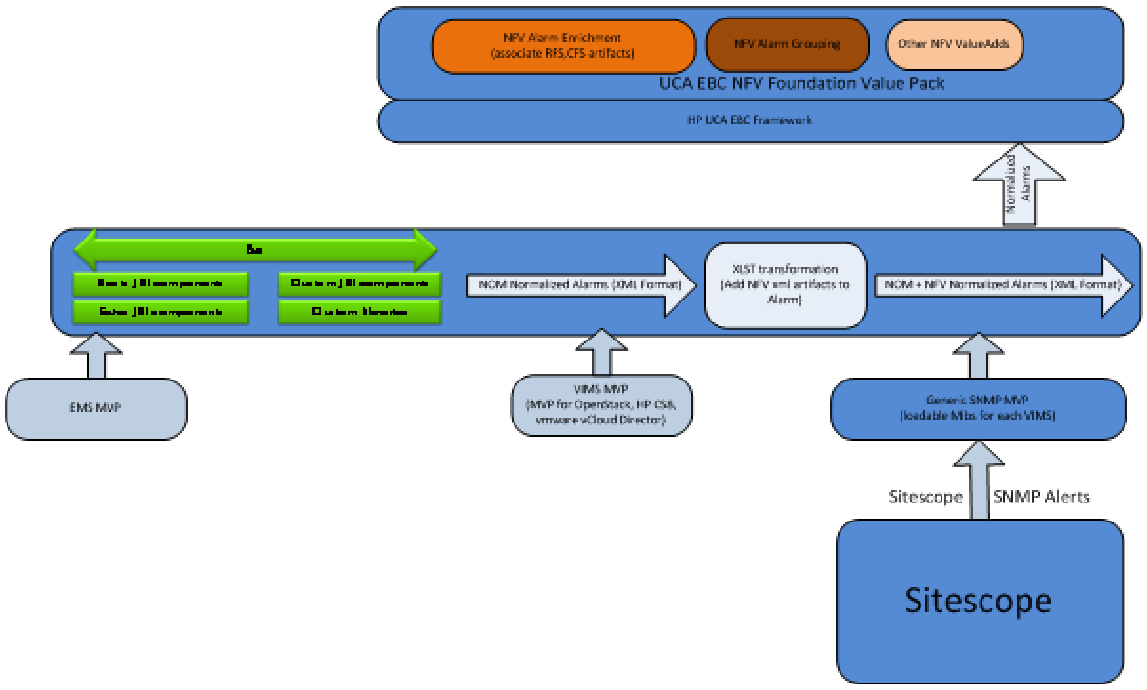

NFV Alarm Enrichment
(associate RFS,CFS artifacts)
NFV Alarm Grouping
Other NFV ValueAdds
UCA EBC NFV Foundation Value Pack
HP UCA EBC Framework
Normalized Alarms
NOM Normalized Alarms (XML Format)
XLST transformation
(Add NFV xml artifacts to Alarm)
NOM + NFV Normalized Alarms (XML Format)
EMS MVP
VIMS MVP
(MVP for OpenStack, HP CSB, vmware vCloud Director)
Generic SNMP MVP
(loadable Mibs for each VIMS)
Sitescope
SNMP Alerts
Sitescope

Figure 29 Integration of monitoring threshold crossing alerts for Correlation

Each monitor deployed in SiteScope is capable of generating an alert. Basically, SiteScope sends alerts for good, error and warning conditions as and when detected by the Monitor.

For this integration purpose, we use SNMPv2 where SiteScope sends SNMPv2 alerts to OM Bus; the OM bus in turn transforms the SNMP alert to the format that is recognizable and usable by UCA EBC.

In order to integrate SiteScope monitoring alerts with the UCA EBC platform, NFV Director provides a default SNMP template file NFVD_SNMP_TEMPLATE_XML.xml.

This file enables SiteScope monitors to generate and send SNMPv2 traps to OM Bus.

This file is usually available in:
<SiteScopeProduct>/templates.snmp/NFVD_SNMP_TEMPLATE_XML.xml

By default, all the out-of-box monitors provided with NFV Director are pre-integrated to use so that alerts are pushed to UCA-EBC via HP OM Bus.

See *HP NFV Director v3.0 Monitoring Guide* document for more detailed info.

3.9 Embedded VNF manager

3.9.1 Resource handling

NFV Director can behave as VNF manager allowing to:

- Define a catalog of VNF descriptors and its flavors.
- Deploy the VNF on the appropriate infrastructure (if the infrastructure and the VIMs are configured well and if the information is loaded into NFV Director).
- Monitor the VNF until it is deployed and managed in the lifecycle of the VNF.

- Detects events or manually triggers those events that scale in or out the VNF, or start and stop the VMs of the VNF.

VNF descriptors can be modeled as different templates according to the different flavors or the VNF configuration. For example of a Virtual CDN we can have:

- VNF_CDN_Compact_Template
    - This template may be modeling the configuration, where all CDN components except for the endpoint are deployed in the same virtual machine.
- VNF_CDN_Distributed_Template
    - This template may be modeling the configuration where all CDN components are deployed in different virtual machines.

Several workflows allow the performance of the above actions.

1. WF_NFVD_CREATE_INSTANCES_FROM_TEMPLATE

- Create a VNF instance (need a template id) in the database using a template following the policies defined in the template.
- This operation creates only the VNF structure in the database.

2. WF_NFVD_DELETE_INSTANCE

- The equivalent delete operation of the previous one (need an instance id).

3. ACTIVATE/DEACTIVATE

- These two workflows create or delete all the VMs in the corresponding VIM once the VNF has been properly assigned, and either deploys or undeploys the monitors for each one.

4. ASSIGMENT

- This workflow allows allocating resources from the servers of a datacenter, for the Virtual machines of the VNF.
- In fact, it uses ASSIGNMENT_RELATIONSHIP artifact to model the kind of relationships that must be created between a target instance, and a resource tree instance.
- It needs the instance ID of the VNF, instance ID of the datacenter, and the instance ID of the ASSIGNMENT_RELATIONSHIP artifact. The last one indicates the number of relations that needs to be created between the first tree and the second tree, and the level in which those must be created.

5. MONITORING

- SiteScope monitors will monitor the behavior of the VNF and will forward the information to the UCA correlation engine. The correlation engine will trigger any necessary actions like scale in or out operations.

6. WF_NFVD_SCALE_OUT/IN

- These two workflows scale in or out a VNF (they need the instance id).
- They look at the configuration of the instance policies and the configuration of the template policies.

For more info about VNF Manager see *HP NFV Director v3.0 VNF Manager Guide* document.

Chapter 4

NFV Director Customization

4.1 Custom resource handling

Extensions of the Artifact – Relationship model can include or modify current attributes, categories or even new artifact and relationships.

4.2 Custom resource monitoring

SiteScope provides the ability to develop monitoring templates in order to monitor KPIs for a given resource.

The procedure to develop customized SiteScope templates is described in SiteScope user documents.

Once the required SiteScope template is developed, the next step would be to integrate it with NFV Director.

The following procedure needs to be executed to integrate this new SiteScope template with NFV Director:

1. Import the SiteScope template in SiteScope.
2. Modify SiteScope template to send SNMPv2 traps to OM bus.
3. Adding monitor artifact to the VNF template.
4. Adding generic monitor handler artifact to VNF template.
5. Deploying the monitor.

   For more details, see Chapter 5Deploying or running customization.
6. Activating the monitor.

   For more details, see Chapter 5 or running customization.

See the out of the Box provided in SiteScope Monitor templates as reference to develop and integrate with UCA EBC.

4.3 Custom event collection

See the Open Mediation 6.2 guide, `OSS_Skeleton_CA_archetype_for_UCAEBC_Guide.pdf` to implement new channel adapter for NFVD.

In NFVD3.0, the SNMP traps from NFVD monitors (SiteScope, VCenter) are translated to x733 format by Generic SNMP Channel Adapter. By default, SiteScope and VMware customization are delivered for NFVD2.0 release. The SNMP UDP port can be configured in Generic SNMP Channel Adapter to receive SNMP traps from monitors. See the `Generic SNMP Channel Adapter` guide for configuration.

Sample alarm received at UCA EBC server.

```
<AlarmCreationInterface xmlns="http://hp.com/uca/expert/x733Alarm">
  <identifier>201019787:1654434a-731b-4547-974d-0bb9e3e0709f</identifier>
  <sourceIdentifier>NFVD_Source</sourceIdentifier>
  <alarmRaisedTime>2014-05-05T20:41:00.848+05:30</alarmRaisedTime>
  <originatingManagedEntity>KVM_TestVM</originatingManagedEntity>
  <originatingManagedEntityStructure>
    <classInstance instance="ossvm1.ind.hp.com" class="Host"/>
  </originatingManagedEntityStructure>
  <alarmType>QUALITY_OF_SERVICE_ALARM</alarmType>
  <probableCause>UtilizationPercentage</probableCause>
  <perceivedSeverity>CRITICAL</perceivedSeverity>
  <networkState>NOT_CLEARED</networkState>
  <operatorState>NOT_ACKNOWLEDGED</operatorState>
  <problemState>NOT_HANDLED</problemState>
  <problemInformation>Attribute not available</problemInformation>
  <specificProblem>ERROR</specificProblem>
  <additionalText>Sitescope alarm|MONITOR.cpuMonitor-001|CONDITION=ERROR|group=KVM VM CPU
Monitor|groupdescription=CPU|groupID=201070542|id=1</additionalText>
  <notificationIdentifier>0</notificationIdentifier>
  <correlationNotificationIdentifiers>Attribute not available</correlationNotificationIdentifiers>
</AlarmCreationInterface>
```

Alarm Attributed Name	**Description**
<identifier>	The value should be unique, so UUID is generated by Channel adapter and is appended with alert.
<sourceIdentifier>	The value should be configured at channel adapter to constant NFVD_Source configure channel adapter.
<perceivedSeverity>	CRITICAL for ERROR condition of monitor. WARNING for WARNING condition of monitor. CLEAR for GOOD condition of the monitor.
<alarmType>	The value should be configured at channel adapter to constant QUALITY_OF_SERVICE_ALARM.
<additionalText>	The value should contain groupdescription=<MonitorName> and <MonitorName> should be the value of property GENERAL.Name of Monitor artifact.

Table 8 NFVD Alarm attributes

4.4 Assurance Notification Interface support

NFVD Assurance provides VNF Lifecycle Changes Notification interface, Os-Nfvo, which is used to provide runtime notifications related to the state of the VNF instance, as a result of changes made to VNF instance including (not limited to) changes in the number of VDUs, changing VNF configuration and/or topology due to auto-scaling/update/upgrade/termination, switching to/from standby etc.

In such cases, NFVD publishes life cycle state change Alarm creation notification, where body is a string that contains XML document that is well formed and valid according to http://hp.com/openmediation/alarms/2011/08 schema, to OM JMS topic named com.hp.openmediation.alarms, which can be consumed by OSS, Analytics tools etc.

The consumer must use ActiveMQ connection factory and connect to broker running on localhost using vm:// transport. If such consumer is interested only in some messages then this Consumer may use JMS Message Selector to specify what messages it is interested in. All the custom field values are available as JMS properties.

For example to select only life cycle notifications: alarmName= LifeCycleStateChangeNotification has to be selected.

For Life cycle notifications: oldState and newState represents the old and new lifecycle states.

Alarms from assurance are configured to publish to specific URL and topic. These are available in the VNFC:OpenMediation Artifact Instance.

For Operational Status notification: oldState and newState represents the old and new Operational status

In case, VNFM provides an interface which can send traps or alarms, assurance correlation problem detection value pack will listen to alarm in x733 format, the mandatory fields required are, alarmName as “LifeCycleStateChangeNotification” and artifactId field and value of artifact id of respective originated managed object as custom field. The alarms will be enriched with topology information, which can be used by OSS. A channel adapter may be required to transform the VNFM traps/alarms to required x733 format.

For more information about alarm creation notification schema, refer to OSS Open Mediation Functional Specification Guide from Open Mediation.

For more information about VNF Lifecycle Changes Notification, Please refer: NFV-MAN001v062, Section 7.2.4 and7.2.5 document for more information.

Sample LifeCycle Alarm:

```
<Alarms xmlns="http://hp.com/openmediation/alarms/2011/08">
		<AlarmCreationInterface>
				<identifier>144e184e-387c-444e-a223-81a955bf8a9c</identifier>
				<sourceIdentifier>NFVD_Source</sourceIdentifier>
				<alarmRaisedTime>2014-12-12T08:34:48.747+05:30</alarmRaisedTime>
				<originatingManagedEntity>VIRTUAL_MACHINE KVMVM-2001</originatingManagedEntity>
				<originatingManagedEntityStructure>
						<classInstance clazz="VIRTUAL_MACHINE" instance="KVMVM-2001"/>
				</originatingManagedEntityStructure>
				<alarmType>UNKNOWN_ALARM_TYPE</alarmType>
				<probableCause>UNKNOWN</probableCause>
				<perceivedSeverity>INDETERMINATE</perceivedSeverity>
				<networkState>NOT_CLEARED</networkState>
				<operatorState>NOT_ACKNOWLEDGED</operatorState>
				<problemState>NOT_HANDLED</problemState>
				<additionalText>NS-512-updated/VNF-BLR1-updated/vnfc-BANGALORE-
updated/KVM_TestVM</additionalText>
				<customFields>
						<customField name="NFVTopology" value="&lt;START&gt;
&lt;VNF_COMPONENT&gt;
artifactId=VNFC-BLR1;
artifactFamily=VNF_COMPONENT;
artifactCategory=GENERIC;
GENERAL.Description=This is VNFC component name change update;
GENERAL.Name=vnfc-BANGALORE-updated;
&lt;/VNF_COMPONENT&gt;
&lt;NETWORK_SERVICE&gt;
artifactId=ns-9001;
artifactFamily=NETWORK_SERVICE;
artifactCategory=GENERIC;
GENERAL.Description=This Network service name update;
GENERAL.Name=NS-512-updated;
&lt;/NETWORK_SERVICE&gt;
&lt;VNF&gt;
artifactId=VNF-BLR1;
artifactFamily=VNF;
artifactCategory=GENERIC;
GENERAL.Description=This is update for VNF-Name changed;
GENERAL.Name=VNF-BLR1-updated;
&lt;/VNF&gt;
&lt;VIRTUAL_MACHINE&gt;
artifactId=KVMVM-2001;
artifactFamily=VIRTUAL_MACHINE;
artifactCategory=GENERIC;
GENERAL.Description=A Virtual machine;
GENERAL.Name=KVM_TestVM;
vimID=null;
hypervisorID=testing-hypervisor-4491-uuid-update1126;
&lt;/VIRTUAL_MACHINE&gt;
&lt;END&gt;
"/>
						<customField name="vmHostName" value="KVM_TestVM"/>
						<customField name="vmName" value="KVM_TestVM"/>
						<customField name="serverHostName" value="sheep.gre.hp.com"/>
						<customField name="vimID" value="89b88213-2fa3-4ec0-9f18-f8422b01890"/>
						<customField name="hypervisorID" value="testing-hypervisor-4491-uuid-
update1126"/>
						<customField name="NOMType"
value="http://hp.com/openmediation/alarms/2011/08"/>
						<customField name="SourceArtifactId" value="KVMVM-2001"/>
						<customField name="source" value="AGW"/>
						<customField name="userText" value="NFVD-PD"/>
						<customField name="oldState" value="INSTANTIATED"/>
						<customField name="newState" value="ACTIVE"/>
						<customField name="alarmName" value="LifeCycleStateChangeNotification"/>
						<customField name="alarmSubtype" value="ArtifactAlarm"/>
				</customFields>
		</AlarmCreationInterface>
</Alarms>
```

Sample Operational status notification Alarm:

```
<Alarms xmlns="http://hp.com/openmediation/alarms/2011/08">
		<AlarmCreationInterface>
				<identifier>3bc32f74-65d3-497e-88c9-68ecc5bbdb39</identifier>
				<sourceIdentifier>NFVD_Source</sourceIdentifier>
				<alarmRaisedTime>2014-12-12T08:44:20.799+05:30</alarmRaisedTime>
				<originatingManagedEntity>VIRTUAL_MACHINE KVMVM-
2001</originatingManagedEntity>
				<originatingManagedEntityStructure>
						<classInstance clazz="VIRTUAL_MACHINE" instance="KVMVM-2001"/>
				</originatingManagedEntityStructure>
				<alarmType>UNKNOWN_ALARM_TYPE</alarmType>
				<probableCause>UNKNOWN</probableCause>
				<perceivedSeverity>INDETERMINATE</perceivedSeverity>
				<networkState>NOT_CLEARED</networkState>
				<operatorState>NOT_ACKNOWLEDGED</operatorState>
				<problemState>NOT_HANDLED</problemState>
				<additionalText>NS-512-updated/VNF-BLR1-updated/vnfc-BANGALORE-
updated/KVM_TestVM|null</additionalText>
				<customFields>
						<customField name="NFVTopology" value="&lt;START&gt;
&lt;VNF_COMPONENT&gt;
artifactId=VNFC-BLR1;
artifactFamily=VNF_COMPONENT;
artifactCategory=GENERIC;
GENERAL.Description=This is VNFC component name change update;
GENERAL.Name=vnfc-BANGALORE-updated;
&lt;/VNF_COMPONENT&gt;
&lt;NETWORK_SERVICE&gt;
artifactId=ns-9001;
artifactFamily=NETWORK_SERVICE;
artifactCategory=GENERIC;
GENERAL.Description=This Network service name update;
GENERAL.Name=NS-512-updated;
&lt;/NETWORK_SERVICE&gt;
&lt;VNF&gt;
artifactId=VNF-BLR1;
artifactFamily=VNF;
artifactCategory=GENERIC;
GENERAL.Description=This is update for VNF-Name changed;
GENERAL.Name=VNF-BLR1-updated;
&lt;/VNF&gt;
&lt;VIRTUAL_MACHINE&gt;
artifactId=KVMVM-2001;
artifactFamily=VIRTUAL_MACHINE;
artifactCategory=GENERIC;
GENERAL.Description=A Virtual machine;
GENERAL.Name=KVM_TestVM;
vimID=null;
hypervisorID=testing-hypervisor-4491-uuid-update1126;
&lt;/VIRTUAL_MACHINE&gt;
&lt;END&gt;
"/>
						<customField name="vmHostName" value="KVM_TestVM"/>
						<customField name="vmName" value="KVM_TestVM"/>
						<customField name="serverHostName" value="sheep.gre.hp.com"/>
						<customField name="vimID" value="89b88213-2fa3-4ec0-9f18-
f8422b01890"/>
						<customField name="hypervisorID" value="testing-hypervisor-4491-uuid-
update1126"/>
						<customField name="NOMType"
value="http://hp.com/openmediation/alarms/2011/08"/>
						<customField name="SourceArtifactId" value="KVMVM-2001"/>
						<customField name="source" value="AGW"/>
						<customField name="oldState" value="faultCode-1012"/>
						<customField name="newState" value="faultCode-1032"/>
						<customField name="alarmName"
value="OperationalStatusChangeNotification"/>
						<customField name="alarmSubtype" value="ArtifactAlarm"/>
				</customFields>
		</AlarmCreationInterface>
</Alarms>
```

Field Description:

alarm Field	**Description**
identifier	Unique alarm Id
sourceIdentifier	Source filter, to be recognized by UCA-EBC
alarmRaisedTime	Alarm raised time
originatingManagedEntity	create/delete/Scale operations originating Entity <ARTIFACT_FAMILY> <ARTIFACT_ID> For E.g. VIRTUAL_MACHINE 123456789
originatingManagedEntityStructure	clazz: ARTIFACT_FAMILY, instance:ARTIFACT_ID
alarmType	Mandatory field as per OM schema. Set to UNKNOWN for Life Cycle Events
probableCause	Mandatory field as per OM schema, Set to UNKNOWN
perceivedSeverity	Mandatory field, added as indeterminate, as alarm will be a lifecycle alarm
networkState	Mandatory Field as per OM schema, Status of the alarm with respect to problem raised by the alarm.
operatorState	Mandatory Field as per OM schema, Status of the alarm with respect to the operator
problemState	Mandatory Field as per OM schema, Status of the alarm with respect to the associated trouble ticket
customField:alarmName	Name of the alarm Operational Status Alarm: OperationalStatusChangeNotification Life Cycle State Alarm:LifeCycleStateChangeNotification
customField:oldState	Old LifeCycle state in case of life cycle alarms Old status in case of operational status alarms
customField:newState	New Life Cycle state in case of life cycle alarms New Operational status in case of operational status change alarm
customField:serverHostname	Host Name of the server
customField:vmHostName	Virtual Machine Host Name
customField:vimID	VIM Identifier. Either uuid or any identifier to identify Vim for that Virtual machine.
customField:hypervisorID	Hypervisor Identifier. Either uuid or any identifier to identify Hypervisor for that Virtual machine.
customField:SourceArtifactId	Source artifact id of the originating alarm
customField:vmName	Virtual Machine Name

alarm Field	Description
customField:source	Alarm generation source Internal: In case of state propagation alarm AGW: In case of life cycle and operational status alarmgenerated within assurance gateway. External: In case of notification from VNFM
customField:NOMType	Variable which can be used as a selector for fetching alarms, presently UCA-EBC CA requires NOMType.
customField:alarmSubtype	Says about by which process alarm has been created, either during create/update/delete of artifacts or create/delete of relationships. ArtifactAlarm, In case of artifact Alarm RelationAlarm, In case of relationship changes alarm
customField:NFVTopology	Topology information
additionalText	Notification event details received from VNFM

Table 9 NFVD Alarms

4.5 Custom value pack

In NFVD3.0 we have "State propagation value' pack, which does the propagation of operational states for the child and all its affected parents based on "Propagation mode".

In case of propagation mode as "RULE_BASED" the alarm will be delegated to a custom value pack.

In order to delegate the alarm to custom value pack, below are the things to consider

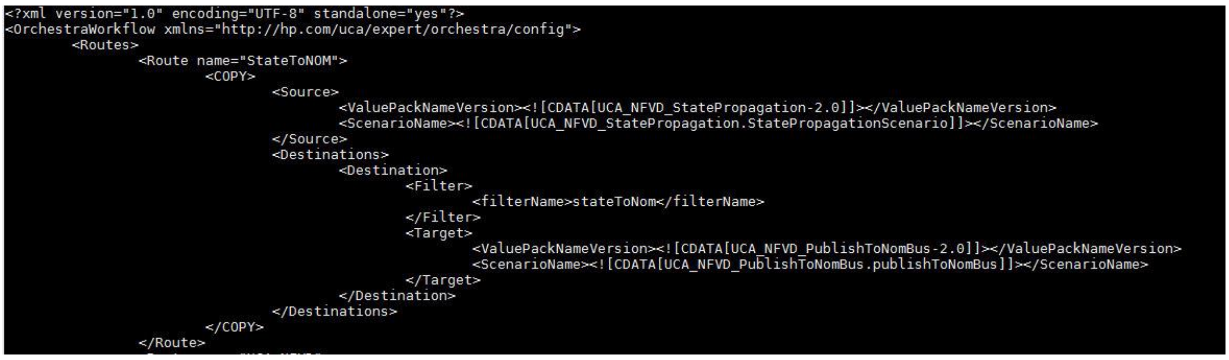
4.5.1 Add route in OrchestraConfiguration.xml

The file will be available at $UCA_EBC_DATA/instances/default/conf/

User has to define all the delegation paths in the OrchestraConfiguration.xml.

In order to delegate the alarm from "State Propagation valuepack" to another custom valuepack, a route has to be defined in this file as per syntax mentioned.

Example:



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OrchestraWorkflow xmlns="http://hp.com/uca/expert/orchestra/config">
        <Routes>
                <Route name="StateToNOM">
                        <COPY>
                                <Source>
                                        <ValuePackNameVersion><![CDATA[UCA_NFVD_StatePropagation-2.0]]></ValuePackNameVersion>
                                        <ScenarioName><![CDATA[UCA_NFVD_StatePropagation.StatePropagationScenario]]></ScenarioName>
                                </Source>
                                <Destinations>
                                        <Destination>
                                                <Filter>
                                                        <filterName>stateToNom</filterName>
                                                </Filter>
                                                <Target>
                                                        <ValuePackNameVersion><![CDATA[UCA_NFVD_PublishToNomBus-2.0]]></ValuePackNameVersion>
                                                        <ScenarioName><![CDATA[UCA_NFVD_PublishToNomBus.publishToNomBus]]></ScenarioName>
                                                </Target>
                                        </Destination>
                                </Destinations>
                        </COPY>
                </Route>
```

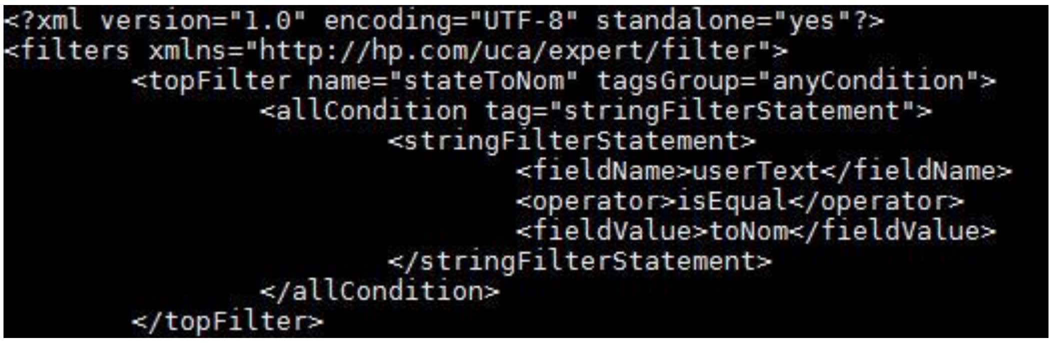
Similarly name and versions of custom value pack needs to be added in the routes so that alarms can be delegated from "UCA_NFVD_StatePropagation-2.0" to custom value pack.

4.5.2 Add filter in OrchestraFilter.xml

The "OrchestraConfiguration.xml" delegates the alarm to destination value pack if the condition mentioned in the filter is satisfied.

So a new filter can be added in "OrchestraFilter.xml" and same can be used in the "OrchestraConfiguratio.xml" to delegate the alarm to destination/custom value pack.

Example:



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<filters xmlns="http://hp.com/uca/expert/filter">
        <topFilter name="stateToNom" tagsGroup="anyCondition">
                <allCondition tag="stringFilterStatement">
                        <stringFilterStatement>
                                <fieldName>userText</fieldName>
                                <operator>isEqual</operator>
                                <fieldValue>toNom</fieldValue>
                        </stringFilterStatement>
                </allCondition>
        </topFilter>
```

4.5.3 Rule file changes

The rule file (like rule.drl) for custom value pack needs to have a condition that matches the value set by "State Propagation" value pack.

The alarms delegated to custom value pack have below property

Alarm custom field	Value
userText	toRule

Table 10 NFVD Alarm custom field

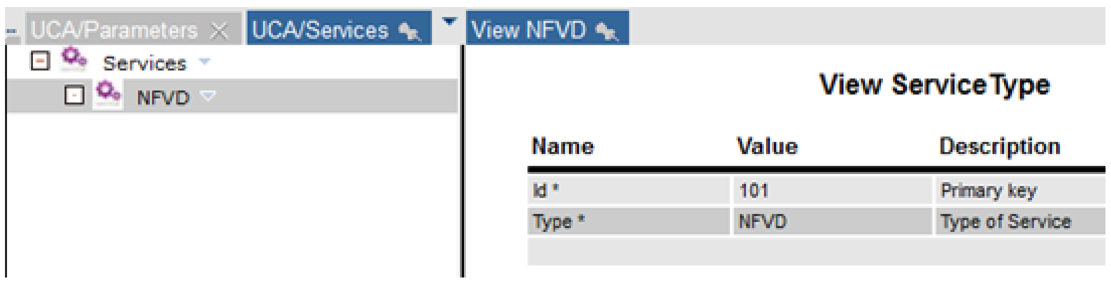
4.5.4 Defining NFVD problem-action framework in UCA automation

See section 7 in `HP UCA Automation – Administrator and User Interface Guide` for understanding the UCA Automation problem-action framework.

The HPSA NFVD solution value pack already has a set of predefined data for the NFVD solution.

4.5.5 Inventory data populated in UCA automation foundation value pack inventory

NFVD domain specific service definition


UCA/Parameters | UCA/Services | View NFVD

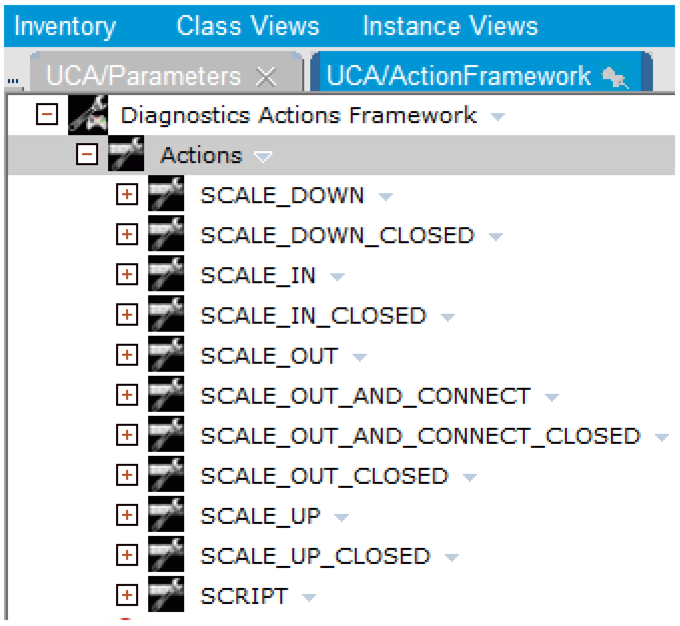
- Services
  - NFVD

View Service Type

Name	Value	Description
Id *	101	Primary key
Type *	NFVD	Type of Service

Figure 30 UCA Automation Foundation Inventory – UCA/Services

Action definitions for NFVD

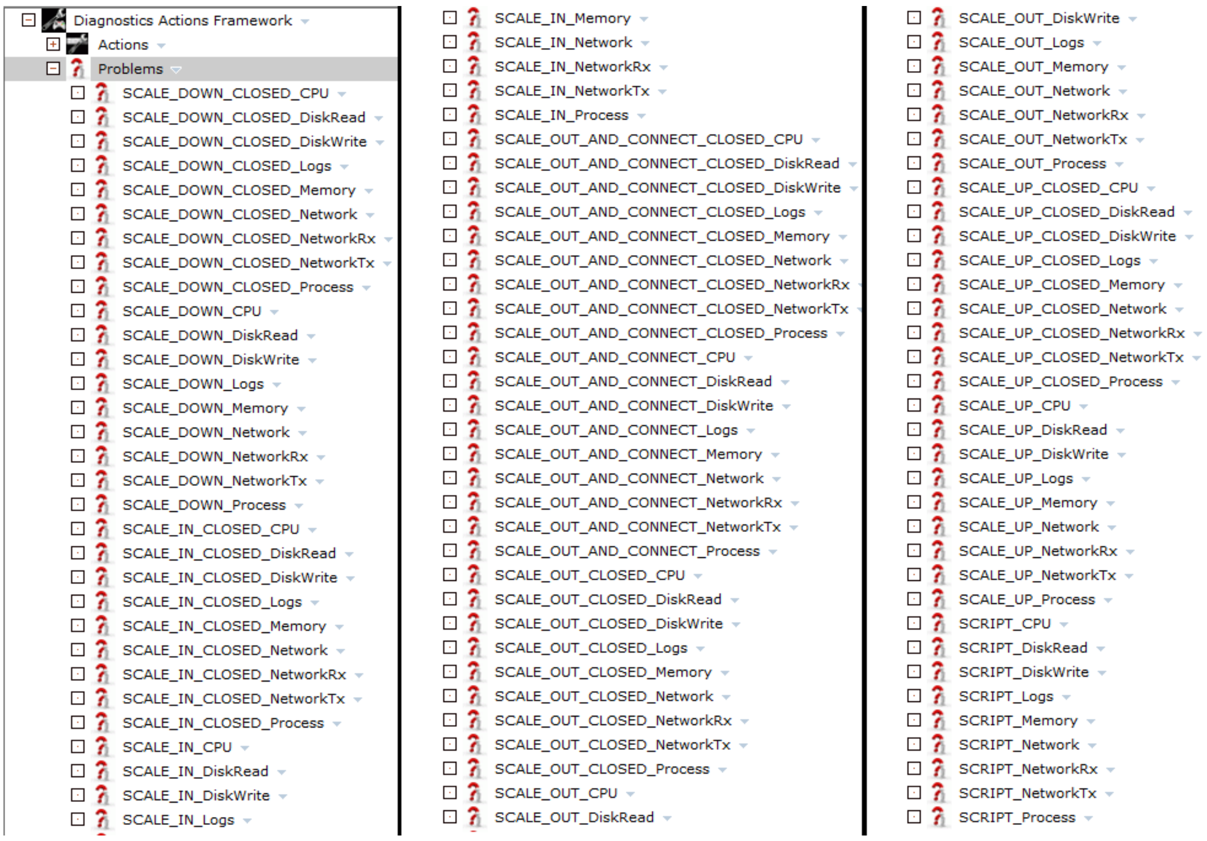

Inventory | Class Views | Instance Views

UCA/Parameters | UCA/ActionFramework

- Diagnostics Actions Framework
  - Actions
    - SCALE_DOWN
    - SCALE_DOWN_CLOSED
    - SCALE_IN
    - SCALE_IN_CLOSED
    - SCALE_OUT
    - SCALE_OUT_AND_CONNECT
    - SCALE_OUT_AND_CONNECT_CLOSED
    - SCALE_OUT_CLOSED
    - SCALE_UP
    - SCALE_UP_CLOSED
    - SCRIPT

Figure 31 UCA Automation Foundation Inventory – UCA/Services

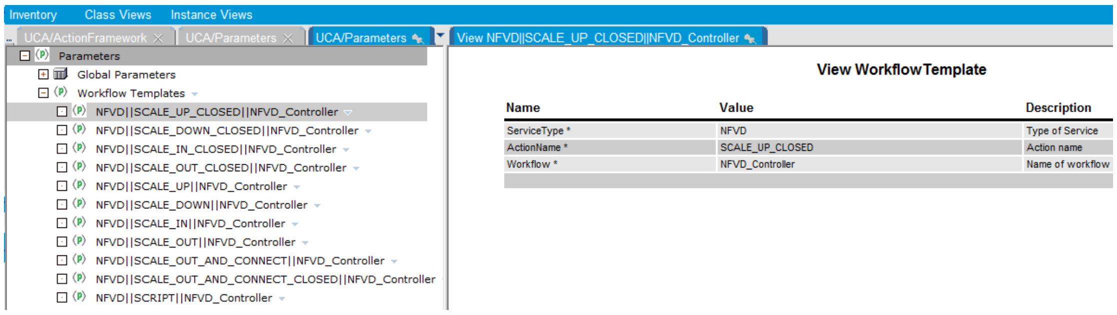
Problem definitions for NFVD:



- Diagnostics Actions Framework
  - Actions
  - Problems
    - SCALE_DOWN_CLOSED_CPU
    - SCALE_DOWN_CLOSED_DiskRead
    - SCALE_DOWN_CLOSED_DiskWrite
    - SCALE_DOWN_CLOSED_Logs
    - SCALE_DOWN_CLOSED_Memory
    - SCALE_DOWN_CLOSED_Network
    - SCALE_DOWN_CLOSED_NetworkRx
    - SCALE_DOWN_CLOSED_NetworkTx
    - SCALE_DOWN_CLOSED_Process
    - SCALE_DOWN_CPU
    - SCALE_DOWN_DiskRead
    - SCALE_DOWN_DiskWrite
    - SCALE_DOWN_Logs
    - SCALE_DOWN_Memory
    - SCALE_DOWN_Network
    - SCALE_DOWN_NetworkRx
    - SCALE_DOWN_NetworkTx
    - SCALE_DOWN_Process
    - SCALE_IN_CLOSED_CPU
    - SCALE_IN_CLOSED_DiskRead
    - SCALE_IN_CLOSED_DiskWrite
    - SCALE_IN_CLOSED_Logs
    - SCALE_IN_CLOSED_Memory
    - SCALE_IN_CLOSED_Network
    - SCALE_IN_CLOSED_NetworkRx
    - SCALE_IN_CLOSED_NetworkTx
    - SCALE_IN_CLOSED_Process
    - SCALE_IN_CPU
    - SCALE_IN_DiskRead
    - SCALE_IN_DiskWrite
    - SCALE_IN_Logs
    - SCALE_IN_Memory
    - SCALE_IN_Network
    - SCALE_IN_NetworkRx
    - SCALE_IN_NetworkTx
    - SCALE_IN_Process
    - SCALE_OUT_AND_CONNECT_CLOSED_CPU
    - SCALE_OUT_AND_CONNECT_CLOSED_DiskRead
    - SCALE_OUT_AND_CONNECT_CLOSED_DiskWrite
    - SCALE_OUT_AND_CONNECT_CLOSED_Logs
    - SCALE_OUT_AND_CONNECT_CLOSED_Memory
    - SCALE_OUT_AND_CONNECT_CLOSED_Network
    - SCALE_OUT_AND_CONNECT_CLOSED_NetworkRx
    - SCALE_OUT_AND_CONNECT_CLOSED_NetworkTx
    - SCALE_OUT_AND_CONNECT_CLOSED_Process
    - SCALE_OUT_AND_CONNECT_CPU
    - SCALE_OUT_AND_CONNECT_DiskRead
    - SCALE_OUT_AND_CONNECT_DiskWrite
    - SCALE_OUT_AND_CONNECT_Logs
    - SCALE_OUT_AND_CONNECT_Memory
    - SCALE_OUT_AND_CONNECT_Network
    - SCALE_OUT_AND_CONNECT_NetworkRx
    - SCALE_OUT_AND_CONNECT_NetworkTx
    - SCALE_OUT_AND_CONNECT_Process
    - SCALE_OUT_CLOSED_CPU
    - SCALE_OUT_CLOSED_DiskRead
    - SCALE_OUT_CLOSED_DiskWrite
    - SCALE_OUT_CLOSED_Logs
    - SCALE_OUT_CLOSED_Memory
    - SCALE_OUT_CLOSED_Network
    - SCALE_OUT_CLOSED_NetworkRx
    - SCALE_OUT_CLOSED_NetworkTx
    - SCALE_OUT_CLOSED_Process
    - SCALE_OUT_CPU
    - SCALE_OUT_DiskRead
    - SCALE_OUT_DiskWrite
    - SCALE_OUT_Logs
    - SCALE_OUT_Memory
    - SCALE_OUT_Network
    - SCALE_OUT_NetworkRx
    - SCALE_OUT_NetworkTx
    - SCALE_OUT_Process
    - SCALE_UP_CLOSED_CPU
    - SCALE_UP_CLOSED_DiskRead
    - SCALE_UP_CLOSED_DiskWrite
    - SCALE_UP_CLOSED_Logs
    - SCALE_UP_CLOSED_Memory
    - SCALE_UP_CLOSED_Network
    - SCALE_UP_CLOSED_NetworkRx
    - SCALE_UP_CLOSED_NetworkTx
    - SCALE_UP_CLOSED_Process
    - SCALE_UP_CPU
    - SCALE_UP_DiskRead
    - SCALE_UP_DiskWrite
    - SCALE_UP_Logs
    - SCALE_UP_Memory
    - SCALE_UP_Network
    - SCALE_UP_NetworkRx
    - SCALE_UP_NetworkTx
    - SCALE_UP_Process
    - SCRIPT_CPU
    - SCRIPT_DiskRead
    - SCRIPT_DiskWrite
    - SCRIPT_Logs
    - SCRIPT_Memory
    - SCRIPT_Network
    - SCRIPT_NetworkRx
    - SCRIPT_NetworkTx
    - SCRIPT_Process

Figure 32 UCA Automation Foundation Inventory – UCA/ActionFramework (Problems)

NFVD Controller workflow mapping definition:



- Parameters
  - Global Parameters
  - Workflow Templates
    - NFVD||SCALE_UP_CLOSED||NFVD_Controller
    - NFVD||SCALE_DOWN_CLOSED||NFVD_Controller
    - NFVD||SCALE_IN_CLOSED||NFVD_Controller
    - NFVD||SCALE_OUT_CLOSED||NFVD_Controller
    - NFVD||SCALE_UP||NFVD_Controller
    - NFVD||SCALE_DOWN||NFVD_Controller
    - NFVD||SCALE_IN||NFVD_Controller
    - NFVD||SCALE_OUT||NFVD_Controller
    - NFVD||SCALE_OUT_AND_CONNECT||NFVD_Controller
    - NFVD||SCALE_OUT_AND_CONNECT_CLOSED||NFVD_Controller
    - NFVD||SCRIPT||NFVD_Controller

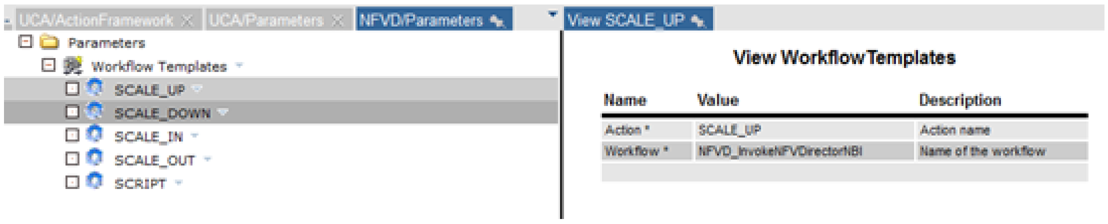
View Workflow Template

Name	Value	Description
ServiceType *	NFVD	Type of Service
ActionName *	SCALE_UP_CLOSED	Action name
Workflow *	NFVD_Controller	Name of workflow

Figure 33 UCA Automation Foundation Inventory – UCA/Parameters

4.5.6 Inventory data populated in NFVD value pack inventory

Action mapping to NFVD child workflows



- Parameters
  - Workflow Templates
    - SCALE_UP
    - SCALE_DOWN
    - SCALE_IN
    - SCALE_OUT
    - SCRIPT

View Workflow Templates

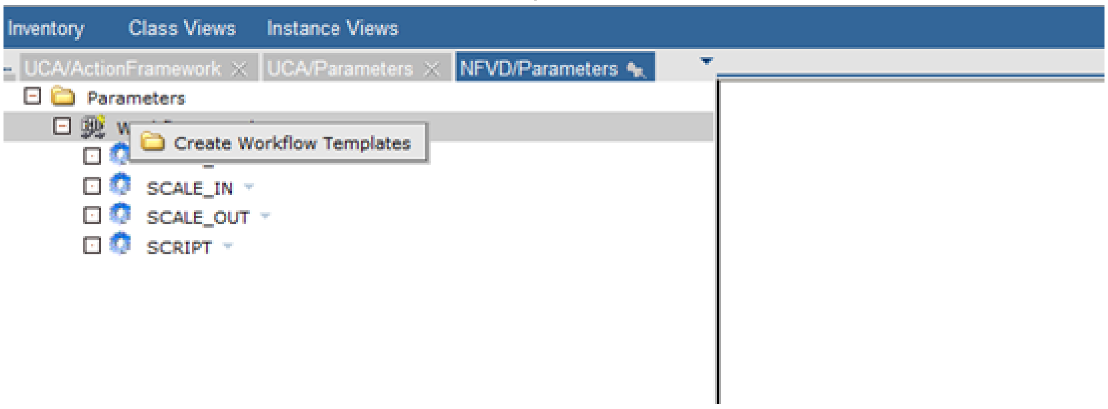
Name	Value	Description
Action *	SCALE_UP	Action name
Workflow *	NFVD_InvokeNFVDirectorNBI	Name of the workflow

Figure 34 NFVD Inventory – NFVD/Parameters

4.5.7 NFVD/Parameters

NFVD/Parameters provides mapping of the actions to the child workflows of NFVD.

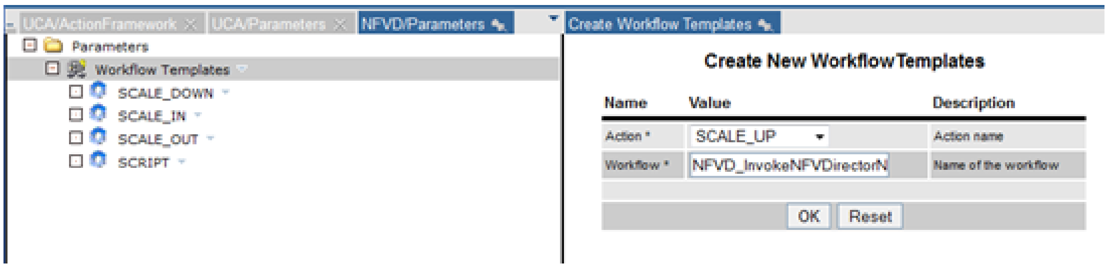
1. Create a new workflow template by performing a right click on NFVD/Parameters → Parameters → Create Workflow Template.



- Parameters
  - W
    - SCALE_IN
    - SCALE_OUT
    - SCRIPT

Figure 35 NFVD/Parameters

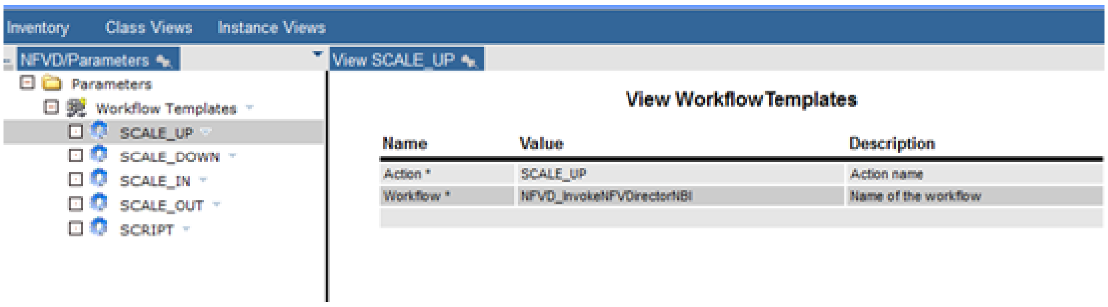
2. Select the required action and enter the name of the child workflow. Click `OK`.



Create New WorkflowTemplates

Name	Value	Description
Action *	SCALE_UP	Action name
Workflow *	NFVD_InvokeNFVDirectorN	Name of the workflow

Figure 36 Creating new Workflow Templates



View WorkflowTemplates

Name	Value	Description
Action *	SCALE_UP	Action name
Workflow *	NFVD_InvokeNFVDirectorNBI	Name of the workflow

Figure 37 View Workflow Templates

3. These mappings can be edited or deleted by performing a right click on them.

4.5.8 Resync topology between Fulfilment and Assurance Engine:

Fulfillment will retry sending notifications which assurance gateway missed when it was down, hence there would not be any resync operation at startup of assurance engine.

Full Topology between fulfilment and assurance can be triggered by invoking the URL: http://ASSURANCE_GATEWAY_HOST:ASSURANCE_GATEWAY_PORT/nfvd/instance/topologyResync. It will be an asynchronous call.

4.6 Discovery & Reconciliation

Discovery and Reconciliation is an independent module for NFV-Director, It discovers and manages resource tree information from underlying VIM, generates artifact instances and relationships, which will minimize manual intervention of resource tree creation, in current release discovery module supports Openstack Juno Rest APIs.

Discovery and Reconciliation features are implemented as channel adapters which will be deployed in Open Mediation. The channel adapters interact between each other using Jms topic internally.

4.6.1 Discovery

Note

VIM discovery can either be performed using a single NOM container or can be distributed among different NOM containers. If different NOM containers are used for VIM discovery, separate REST Endpoint port must be configured in the nfvd-recon-internal.properties. Default port configured is 18989.

When the VIM and AUTHENTICATION Instance data is prepared, the `VIM > CAPACITY_DISCOVERY_PARAMS > NOM_LoadBalance_Instance` must reflect the above NOM container.

Discovery channel adapter for example ,opentack-ca interacts with vim and query the resource details and converts them into NFV-D artifact relationship format model, and sends them to Reconciliation channel adapter (capacity-management-ca)

Openstack channel adapter provides the flexibility to parse the Openstack JSON response using java json path and populate NFV-Director artifact instance, category and attributes.
nfvd-grm-users.properties defines the topic to integrate with reconciliation channel adapter and other various variables and endpoints for programmatic purpose.
Discovery channel adapter uses artifact and relationship definitions to generate artifact and relationship instances

To generate artifact instance,Openstack json response were parsed by`Json paths, written externally in json-paths.properties file, and mapped to artifact elements, category and attributes. By default the artifactcategory will be OPENSTACK, if OPENSTACK category is applicable for a artifact then GENERIC category will be used. In case to generate artifact instance with different artifact category then artifact family and artifact catefory will be separated by hyphen '-' for, e.g., HYPERVISOR-KVM.

Example processing a response from Openstack:

Openstack json response for tenant:

```
{
  "tenant": {
    "description": null,
    "enabled": true,
    "id": "257c47239a6244049136f9d8a8c955bd",
    "name": "admin"
  }
}
```

`Pre-defined apache camel routes in channel adapter:`

```
<setHeader headerName="artifactFamilyCategory">
    <constant>TENANT</constant>
</setHeader>
<to uri="bean:arModelBuilder?method=setArtifactIdByJsonPath"/>
<to uri="bean:arModelBuilder?method=convertVimMessage"/>
```

`NFV-D model mapping in json-paths.properties`

```
#TENANTS
TENANT.id=tenant-$.tenant.id
TENANT.identifier=tenant-$.tenant.id
TENANT.GENERAL.NAME=$.tenant.name
TENANT.GENERAL.DESCRIPTION=$.tenant.description
```

Property format

```
<ARTIFACT FAMILY>.<ARTIFACT ELEMENTS> = [CONSTANTS] [JSON-PATH]
[DELIMITER:$][CONSTANTS]
<ARTIFACT FAMILY>.<CATEGORY>.<ATTRIBUTE> = [CONSTANTS] [JSON-PATH]
[DELIMITER:$][CONSTANTS]
```

```
<ARTIFACT FAMILY>.ARTIFACTDEFINITION,<ATTRIBUTE> = [CONSTANTS] [JSON-PATH]
[DELIMITER:$][CONSTANTS]
```

Relationship between artifacts are defined in child-parent-mapping.properties, relationhip type will be picked up from NFV-D relationship definition.

Format:

Child=Parent1,Parent2 (Comma separated parents)

For example;

```
TENANT=VIM
HYPERVISOR=VIM
HYPERVISOR-KVM=VIM
SERVER=HYPERVISOR,AVAILABILITY_ZONE,ENCLOSURE
```

4.6.2 Reconciliation(Capacity-management-ca)

Capacity management channel adapter, reconciles the data from discovery module and fulfillment database.

Reconciliation uses the below configuration in VIM artifact instance definition

```
  <category>
              <label>CAPACITY_DISCOVERY_PARAMS</label>
              <order>4</order>
              <attributes>
                  <attribute>
                      <label>NOM_LoadBalance_Instance</label>
                      <order>1</order>
                      <mandatory>true</mandatory>
                      <value>0</ value>
                      <type>TEXT</type>
                      <unit>TEXT</unit>
                  </attribute>
                  <attribute>
                      <label>Enable_Rest_Call</label>
                      <order>2</order>
                      <mandatory>false</mandatory>
                      <value>true</value>
                      <type>TEXT</type>
                      <unit>TEXT</unit>
                  </attribute>
                <attribute>
                      <label>Enable_Log_Entry</label>
                      <order>3</order>
                      <mandatory>false</mandatory>
                      <value>false</value>
                      <type>TEXT</type>
                      <unit>TEXT</unit>
                  </attribute>
              </attributes>
          </category>
```

Enable_Rest_Call – To upload instances directly to fulfillment database

Enable_Log_Entry – To Generate a report of artifact relationship instances to be created/updated/deleted

```
NOM_LoadBalance_Instance - Which NOM instance has to be used for this
vim discovery.
```

In case, `Enable_Log_Entry` is set to true, the reconciled data will be logged into different files as given below.

```
CreateArtifactInstances_<CAPACITY_TREE_ID>_<DATE>.xml -contains  new artifacts
instances data to be created in fulfillment
CreateRelationInstances_<CAPACITY_TREE_ID>_<DATE>.xml - contains new
relationship data to be created
CreateTreeInstanceArtifacts_<CAPACITY_TREE_ID>_<DATE>.xml - contains artifacts
data to be added to capacity tree.
CreateTreeInstanceRelationships_<CAPACITY_TREE_ID>_<DATE>.xml - contains
relationship data to be added to capacity tree.
DeleteArtifactInstances_<CAPACITY_TREE_ID>_<DATE>.xml - Contains existing
artifacts to be deleted.
DeleteRelationInstances_<CAPACITY_TREE_ID>_<DATE>.xml - contains existing
relationships to be deleted.
UpdateArtifactInstances_<CAPACITY_TREE_ID>_<DATE>.xml - contains artifact which
has changes in attribute values and require update in fulfillment
computeAttributes.properties gives the artifact, categories and attributes
eligible for reconciliation.
```

Example attributes configuration are given below.

```
HYPERVISOR.KVM=true
HYPERVISOR.VMWARE=true
HYPERVISOR.VCENTER=true
HYPERVISOR.KVM.GENERAL.host=true
HYPERVISOR.KVM.GENERAL.Type=true
HYPERVISOR.VMWARE.GENERAL.host=true
HYPERVISOR.VMWARE.GENERAL.Type=true
HYPERVISOR.VCENTER.GENERAL.host=true
HYPERVISOR.VCENTER.GENERAL.Type=true
```

Format:

<ARTIFACT-FAMILY>.<ARTIFACT-CATEGORY>.<CATEGORY>.<ATTRIBUTE>=true/false

4.7 Export KPI metrics and Topology Information

4.7.1 Export Topology

Topology information i.e notifications received from Fulfillment are loaded into a file if analytics is enabled .

Analytics can be enabled by setting the property STARTUP_ANALYTICS to true in nfvd.properties file in /var/opt/HP/nfvd/conf. A couple of minutes are needed for this to take effect. If this property is set to true then a thread will be started which will be waiting for notifications to be received. The wait time for the thread can be configured in nfvd.properties file and the property to be set is ANALYTICS_TIME_TOWAIT_FORARTIFACTSTOBERECEIVED. By default the value is 300000 milli seconds. The Analytics thread will be waiting for any notifications to be received within this wait time. If no more notifications are received within this time, Analytics thread will calculate the capacity at Virtual_Machine, VNF, VNF_Component, Netowrk_Service, Server, Region, Availabilty_zone, Rack, Data Center, Location levels.

The topology files are created in CSV format and the files are exported in /var/opt/HP/nfvd/DataDirectory/Topology. The topology files will be dumped into this directory. The format of the files that are dumped will be TOPOLOGY_<ARTIFACTFAMILY><ARTIFACTCATEGORY>_<HOST>_<TIMESTAMP>.csv. For details about the file contents, please refer the USER and Administration Guide.

Assurance receives notifications from fulfillment for many artifacts, But the csv files aren't generated for all the artifacts. The csv files are generated only for those artifact family and artifact category which are configured in the nfvd-internal.properties file. The format which has to be specified for csv files to be generated in the nfvd-internal.properties file is export.VIRTUAL_MACHINE.GENERIC=true.

A default set of artifact family and artifact category is configured in the nfvd-internal.properties file . If the artifact data shouldn't be exported to a csv file then export.VIRTUAL_MACHINE.GENERIC should be set to false .

Eg: export.VIRTUAL_MACHINE.GENERIC=false.

Any valid artifact_ can be exported. The artifact family and artifact category for that artifact should be configured as

export.<ARTIFACTFAMILY>.<ARTIFACTCATEGORY>=true

Eg: export.MONITOR.GENERIC=true

4.7.2 Dump All Topology

Dump All Topology is a feature in NFVD which dumps the data from neo4j Database to CSV Files. The CSV files that are produced based on the configuration in the nfvd-internal.properties file i.e export.<ARTIFACTFAMILY>.<ARTIFACTCATEGORY>=true . The Dump All topology should be invoked through a REST API.

The API to access is http://<HOST>:<PORT>/instance/artifact/dumpalltopology

HOST : host where Assurance gateway is running

Port : Port where Assurance is configured.

4.7.3 Export KPI METRICS

The metrics information generated from Sitescope are exported into a CSV file and exported into metrics folder. The folder where the metrics information is stored is /var/opt/HP/nfvd/DataDirectory/Metrics. The metrics information is triggered from Sitescope for a predefined interval. Please refer installation guide for more details

The columns of the files are

FIELD	FIELD NAME	DESCRIPTION/COMMENT
1	COLLECTOR	Application collecting the data, which is always Site Scope
2	COLLECTOR HOST	SiteScope Host
3	GROUP NAME	Name of the Group
4	GROUP DESCRIPTION(OPTIONAL)	Group description if entered for the group
6	TYPE	Type of the Monitor eg:Memory,CPU,Costom Monitor etc.
7	TARGET	Remote server being monitored
8	TARGET IP	IP address of the remote server being

		monitored
9	TIMESTAMP	Time of the measurement
10	MONITOR QUALITY	Status as determined by the monitor's thresholds **Possible values**: 0 - no data (no thresholds defined) 1 - informational (good) 2 –warning 3 - critical
11	SOURCE TEMPLATE NAME	Name of the source template
12	MONITOR NAME	Monitor name as defined by the user
13	MONITOR DESC(OPTIONAL)	Monitor description if entered for the monitor
14	VM ARTIFACT ID	ARTIFACT ID of the VM
15	HYPERVISOR ID	ARTIFACT ID of the Hypervisor
16	VIM ID	ARTIFACT ID of the VIM
17	COUNTER 1 QUALITY	Status of the counter as determined by the counter's threshold **Possible values**: 0 - no data (no thresholds defined) 1 - informational (good) 2 – warning 3 - critical
18	COUNTER 1 STATUS (OPTIONAL)	
19	COUNTER1 DESCRIPTION(OPTIONAL)	Monitor description if entered for the monitor
20	COUNTER 1 NAME	Counter name
21	COUNTER 1 VALUE	Counter value
22	COUNTER 2 QUALITY	
23	COUNTER 2 STATUS	
24	COUNTER2DESCRIPTION	
25	COUNTER2 NAME	

Sample contents of the files are

```
COLLECTOR,COLLECTOR HOST,General.Name,GROUP DESCRIPTION,TYPE,TARGET,TARGET
IP,TIMESTAMP,MONITOR QUALITY,SOURCE TEMPLATE NAME,MONITOR NAME,MONITOR
DESC,MONITOR_TYPE,MONITOR_ARTIFACT_ID,MONITORED_ENTITY_ID,MONITORED_ENTITY_FAMI
LY,MONITORED_ENTITY_CATEGOORY,VM_HYPERVISOR_ID,VM_VIM_ID,COUNTER NAME1,COUNTER
VALUE1,COUNTER STATUS 1,COUNTER DESCRIPTION1,COUNTER QUALITY1,COUNTER
NAME2,COUNTER VALUE2,COUNTER STATUS 2,COUNTER DESCRIPTION2,COUNTER
QUALITY2,COUNTER NAME3,COUNTER VALUE3,COUNTER STATUS 3,COUNTER
DESCRIPTION3,COUNTER QUALITY3,COUNTER NAME4,COUNTER VALUE4,COUNTER STATUS
4,COUNTER DESCRIPTION4,COUNTER QUALITY4,COUNTER NAME5,COUNTER VALUE5,COUNTER
STATUS 5,COUNTER DESCRIPTION5,COUNTER QUALITY5,COUNTER NAME6,COUNTER
VALUE6,COUNTER STATUS 6,COUNTER DESCRIPTION6,COUNTER QUALITY6,COUNTER
NAME7,COUNTER VALUE7,COUNTER STATUS 7,COUNTER DESCRIPTION7,COUNTER
QUALITY7,COUNTER NAME8,COUNTER VALUE8,COUNTER STATUS 8,COUNTER
DESCRIPTION8,COUNTER QUALITY8,COUNTER NAME9,COUNTER VALUE9,COUNTER STATUS
9,COUNTER DESCRIPTION9,COUNTER QUALITY9,COUNTER NAME10,COUNTER VALUE10,COUNTER
STATUS 10,COUNTER DESCRIPTION10,COUNTER QUALITY10

SiteScope,nfvdvm31,VIRTUAL_MACHINE OPENSTACK CPU,(MONITOR_ARTIFACT_ID=0aad1b44-
8301-49c5-ad17-a771d58f208e)(MONITOR_TYPE=CPU)(MONITORED_ENTITY_ID=4e93e066-
e084-4949-bdf6-
a017dd236b00)(MONITORED_ENTITY_FAMILY=VIRTUAL_MACHINE)(MONITORED_ENTITY_CATEGOR
Y=GENERIC)(VM_HYPERVISOR_ID=7d941343-0170-4317-909a-71a413ca57b6)(
VM_VIM_ID=7d941343-0170-4317-909a-
71a413ca57b6)(counterName=cpu_usage_average),Custom Monitor,nfvdvm31,unknown
```

```
host,2015-06-
18T11:21:48.887+0000,3,NFVDirector/VIRTUAL_MACHINE/OPENSTACK/CPU,7d941343-0170-
4317-909a-71a413ca57b6,null,Custom Monitor,0aad1b44-8301-49c5-ad17-
a771d58f208e,4e93e066-e084-4949-bdf6-
a017dd236b00,VIRTUAL_MACHINE,null,7d941343-0170-4317-909a-
71a413ca57b6,7d941343-0170-4317-909a-71a413ca57b6,cpu_usage_average,No
data,0,null,3
```

Chapter 5

Deploying or running customization

5.1 Custom resource monitoring

Once the custom monitors imported into SiteScope and the VNF template is updated to use these new monitors, then the following operations can be done on the new custom VNF monitors.

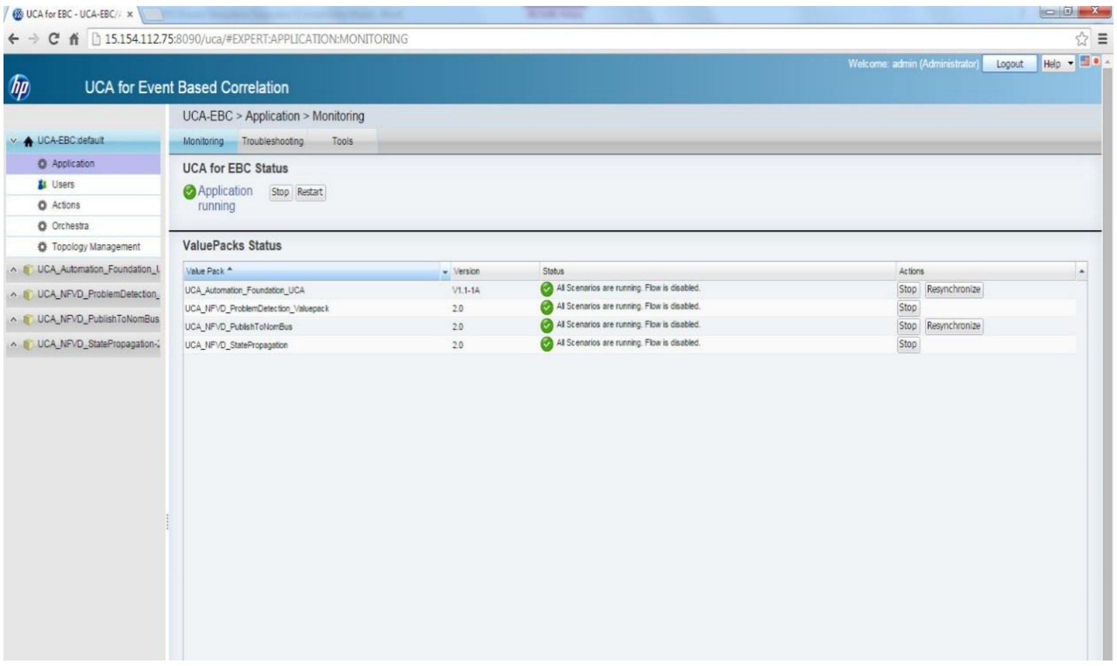
When you invoke the VNF CREATE WebService API, the new custom monitors will get deployed and activated as part of VNF CREATE process.

5.2 Custom event collection

See the `Generic SNMP Channel Adapter Installation and Configuration guide` for deployment of Generic SNMP Channel Adapter for customization of event collection.

5.3 Custom automated action

Launch UCA EBC UI, login as admin to deploy and start UCA NFVD Problem Detection Valuepack, UCA Automation Foundation Valuepack, and UCA NFVD PublishToNomBus Valuepack.


UCA for EBC - UCA-EBC/
15.154.112.75:8090/uca/#EXPERT:APPLICATION:MONITORING
Welcome: admin (Administrator)
Logout
Help
UCA for Event Based Correlation
UCA-EBC > Application > Monitoring
Monitoring
Troubleshooting
Tools
UCA-EBC:default
Application
Users
Actions
Orchestra
Topology Management
UCA_Automation_Foundation_
UCA_NFVD_ProblemDetection_
UCA_NFVD_PublishToNomBus
UCA_NFVD_StatePropagation-
Stop
Restart

UCA for EBC Status

Application running

ValuePacks Status

Value Pack	Version	Status	Actions
UCA_Automation_Foundation_UCA	V1.1-1A	✓ All Scenarios are running. Flow is disabled.	Stop Resynchronize
UCA_NFVD_ProblemDetection_Valuepack	2.0	✓ All Scenarios are running. Flow is disabled.	Stop
UCA_NFVD_PublishToNomBus	2.0	✓ All Scenarios are running. Flow is disabled.	Stop Resynchronize
UCA_NFVD_StatePropagation	2.0	✓ All Scenarios are running. Flow is disabled.	Stop

Figure 38 UCA for Event Based Correlation

Chapter 6

Examples of templates

6.1 Data center template

Please see the embedded Data center template.

DataCenterTemplate_6.1.xml

6.2 VNF template example

Please see the embedded VNF Template example.

VNFTemplate_6.2.xml

6.3 VNF template example

Please see the embedded VNF Create_and_Connect (VNF with NET and Scales and Monitor) example.

VNFwithNet_and_Sc
ales_new.xml

Activation_TASK_DE
FINITION_Tree_new.

IcehouseResources_
new.xml

OpenstackResources
_new.xml

VNFwithNet_and_Sc
ales_and_Monitor_ne

Chapter 7

Integrating with a VNF Manager

7.1 Register VNFManager

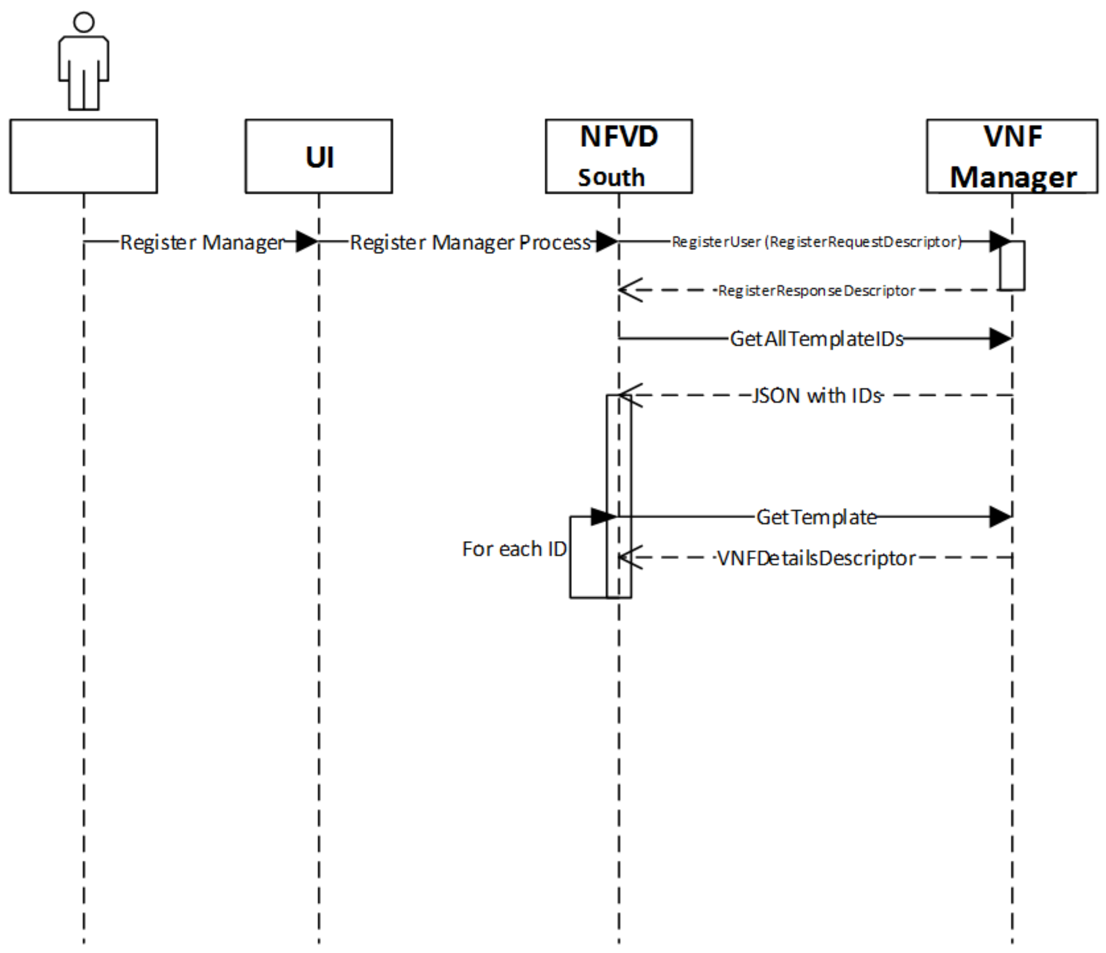


Figure 39 Register VNFManager

7.2 Instantiate a VNF

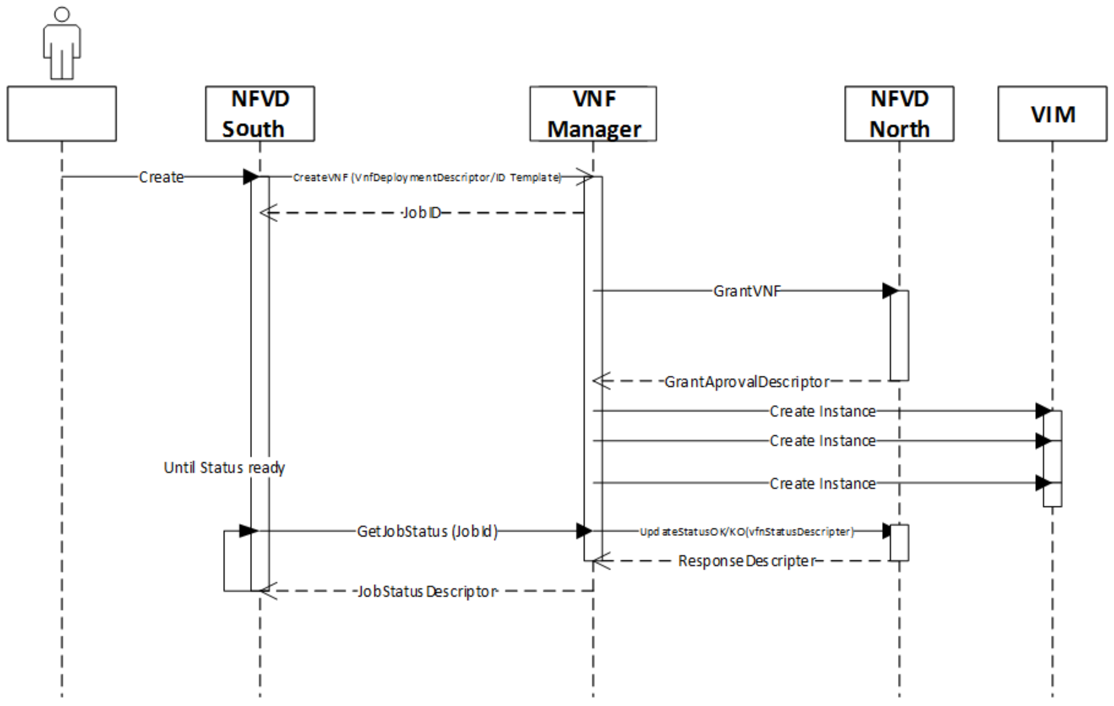


Figure 40 Instantiate a VNF

7.2.1 Behavior

Next steps and defined as complete process to create a VNF managed by VNFManager. Previously NFVD must contain the descriptor (for VNFManager) as a VNF Template.

To relate this behavior, we consider better to understanding if all steps describes supported by same example, to do that with consider next scenario of Descriptor or Template:

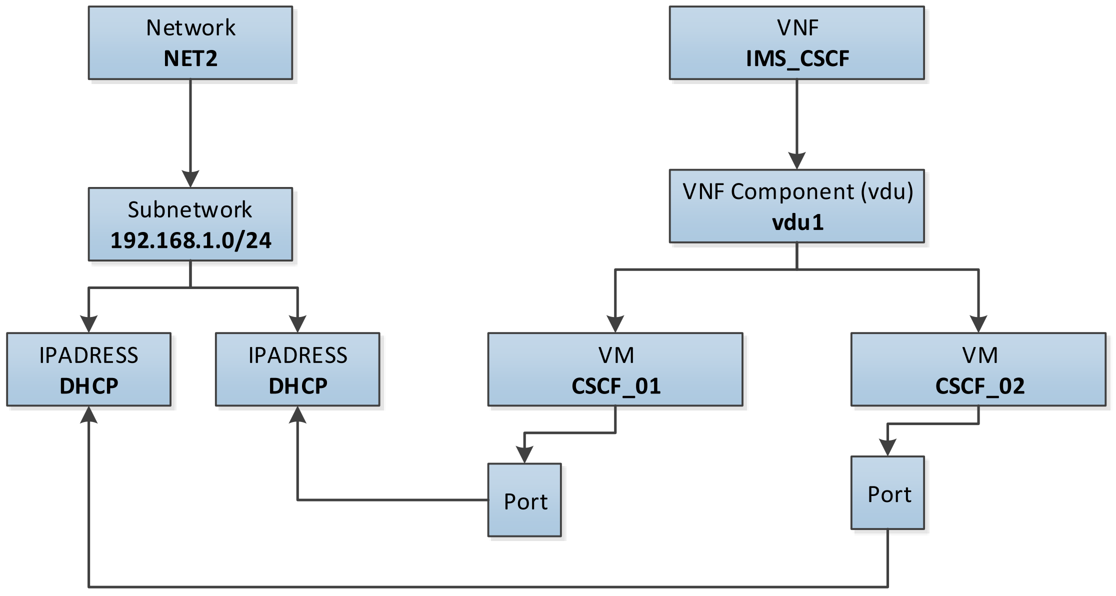


Figure 41 Behavior

7.2.1.1 Creation from NVFD UI

User selects a template from NFVD UI and launches creation of a VNF Manager. This request must contain:

VNF Template (defined previously by VNF Manager Descriptor)

Flavor of descriptor that he wants create.

Descriptor version

7.2.1.2 Invocation of Creation to a VNFManager

Since out template, the process must be query what manager manage this template and its credential:

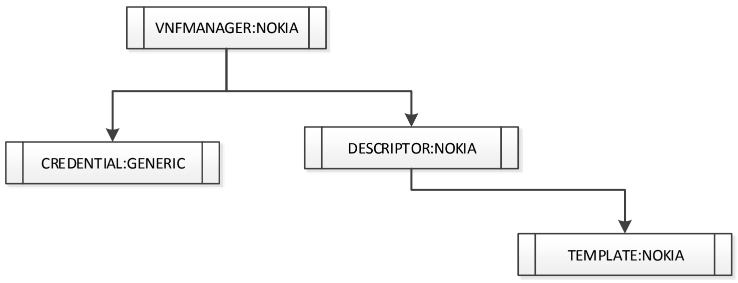


Figure 42 VNFManager model

Then, instantiate process needs to create invocation to a VNFmanager with:

URL: Composed by attributes ENDPONT.host and ENDPOINT.port from Credential Artifact

Http basic authentication with GENERAL.user and GENERAL.password from Credential Artifact

Http operation: POST

Path: v1/vnfdescriptor/<vnfdescriptorid>. Vnfdescriptorid queried from GENERAL.id (Descriptor artifact)

Json, it includes next tags:

- vnfdescriptorid: mandatory, same that URL path
- flavor: selected by user.
- Descriptor version
- vnfDetailsDescriptor: Optionally, this request may include the original descriptor stored on CONTENT.json attribute of DESCRIPTOR Artifact
- grantApprovalDescriptor: Optionally NFVD may include final resources used to allocate VNF.

Example:

```
{
"vnfdescriptorid" : "ID_Descriptor",
"flavor" : "Gold",
"vnfdescriptorversion" : "1.0",
"vnfDetailsDescriptor": {
        "vnfdescriptorid" : "ID_Descriptor",
        "vnf":[ {
                "version": "1.5",
                "templateId":"IMS_CSCF",
                "lowlevel_assigment_supported" : "vim"
                "vdus":[{
                        "id" : "vdu1",
                        "vms":[ {
                                "id" : "CSCF_01",
                                "status" : "poweredon",
                                "image" : "image name used to instantiate VM on
VIM",
                                "flavor" : "flavor of VM used",

                                "Resources": {
                                        "noOfCPUs": {
                                                "Amount" : "3"
                                        },
                                        "memoryinMB": {
                                                "Amount" : "10240"
                                        },
                                        "diskinGb": {
                                                "Amount" : "100"
                                        },
                                        "luns": [{
                                                "id" : "lun1",
                                                "Amount" : "3"
                                        }]
                                },
                                "Networks":[{
                                        "id": "NET2"
                                }]
                        },
                        {
                                "id" : "CSCF_02",
                                "status" : "poweredon",
                                "Resources": {
                                        "noOfCPUs": {
                                                "Amount" : "3"
                                        },
                                        "memoryinMB": {
                                                "Amount" : "10240"
                                        },
                                        "diskinGb": {
                                                "Amount" : "100"
                                        },
                                        "luns": [{
                                                "id" : "lun1",
                                                "Amount" : "3"
                                        }]
                                },
                                "Networks":[
                                        {"id": "NET2"}
                                ]
                        }]
                }]
                "flavors" : [{
                        "id" : "Gold",
                        "includedVDUIds" : "vdu1"
                        "includedVMIds" : "CSCF_01, CSCF_02"
                }],
                "networks" : [
                        {
                        "id" : "NET2",
                        "shared" : "false",
                        "admin_state_up" : "true ",
                        "external" : "false",
                        "provider-physical_network" : "",
                        "provider-network_type" : "",
                        "provider-segmentation_id" : "",
                        "subnetworks" : [{
                                "type" : "ipv4 ",
                                "id" : "subnet1",
                                "ip" : "192.168.1.0",
                                "mask" : "24",
                                "enable_dhcp" : "true",
                                "gateway_ip" : "192.168.1.1",
                                "allocation_pools" : ""
                                }]
                }]
        }]
} ,
"grantApprovalDescriptor": {
        "vnf": {
                "vnfInstantiationPossible": "true",
                "id": "IMS",
                "flavorid" : "Gold",
                "vnfInstantiate": {
                        "vim": {
                                "id": "IMSVIM",
                                "tenant id": "TENANT_1",
                                "Credentials":{
                                        "username": "nfvoadmin",
                                        "password": "Passw0rd432"
                                }
                        }
                }
        }
}
}
```

The response of that request must include:

jobId: to query for result of this operation

link: optionally to identify endpoint for a REST request to get job status

entityID: optionally the identifier of new VNF instance in creation

entitylink: optionally to identify endpoint for a REST request to get VNF details

Example

```
{
"description": "......",
"jobId": "1001001",
"link":http: //<machine>: <port>/v1/jobs/1001001"
"entityId" : "IMS 01"
"entitylink" : "http: //<machine>: <port>/v1/vnf/IMS_01"
}
```

7.2.1.3 Pooling for Jobs

Parallel NFVD works on two ways, the first one is query periodically for jobs status and retrieve VNF information; and on the other hand, VFN Manager must query NFVD to get permission for deploy VNF (grant operation) if this grant was not included on first create invocation.

Pooling until Job finish

NFVD will query periodically for job status to a VNF Manager with this parameters:

URL: detailed on previous step response.

Http basic authentication with GENERAL.user and GENERAL.password from Credential Artifact

Http operation: GET

Path: v1/jobs/<job_id> (/v1/jobs/1001001)

The JSON response example may be:

```
{
"jboId": "1001001",
"jboStatus": "Finished",
"jobDescription": "service creation job",
"statusDescription": "Request is posted into VNFM"
}
```

Note: Final status must be “Error” or “Finished”

Getting VNF creation result

When creation jobs is finish NFVD may query for details of VNF created, to do that NFVD will invoke this operation:

URL: detailed on previous step response (entitylink) http: //<machine>:<port>/v1/vnf/<vnf_id>”

Http basic authentication with GENERAL.user and GENERAL.password from Credential Artifact

Http operation: GET

Path: v1/vnf/<vnf_id> (/v1/vnf/IMS_01)

The JSON response example may be:

```
{
"vnfDeploymentDescriptor": {
        "flavor" : "Gold",
        "version": "1.5",
        "vnfdescriptorid" : "ID_Descriptor",
        "vnf":{
                "vnfid" : "IMS_01",
                "templateId":"IMS_CSCF",
                "lowlevel assigment supported" : "vim"
                "vdus":[{
                        "id" : "vdu1",
                        "vms":[ {
                                "id" : "CSCF_01",
                                "status" : "poweredon",
                                "hostname" : "vm.hp.com",
                                "image" : "image name used to instantiate
VM on VIM",
                                "flavor" : "flavor of VM used",
                                "VIM" : {
                                    "intanceid" : "Artifact ID inside
VIM",
                                    "instancename" : "Artifact NAME
inside VIM"
                             },
                             "HYPERVISOR" : {
                                    "intanceid" : "Artifact ID inside
Hypervisor",
                                    "instancename" : "Artifact NAME
inside Hypervisor"
                             }
                             "Resources": {
                                    "noOfCPUs": {
                                           "Amount" : "3"
                                    },
                                    "memoryinMB": {
                                           "Amount" : "10240"
                                    },
                                    "diskinGb": {
                                           "Amount" : "100"
                                    },
                                    "luns": [{
                                           "id" : "lun1",
                                           "Amount" : "3"
                                    }]
                             },
                             "Networks":[{
                                    "id": "NET2"
                             }]
                      },
                      {
                             "id" : "CSCF_02",
                             "status" : "poweredon",
                             "hostname" : "vm.hp.com",
                             "image" : "image name used to instantiate
VM on VIM",
                             "flavor" : "flavor of VM used",
                             "VIM" : {
                                    "intanceid" : "VM ID inside VIM",
                                    "instancename" : "VM NAME inside
VIM"
                             },
                             "HYPERVISOR" : {
                                    "intanceid" : "VM ID inside
Hypervisor",
                                    "instancename" : "VM NAME inside
Hypervisor"
                             }
                             "Resources": {
                                    "noOfCPUs": {
                                           "Amount" : "3"
                                    },
                                    "memoryinMB": {
                                           "Amount" : "10240"
                                    },
                                    "diskinGb": {
                                           "Amount" : "100"
                                    },
                                    "luns": [{
                                           "id" : "lun1",
                                           "Amount" : "3"
                                    }]
                             },
                            "Networks":[
                                   {"id": "NET2"}
                            ]
                     }]
              }]
              "flavors" : [{
                     "id" : "Gold",
                     "includedVDUIds" : "vdu1"
                     "includedVMIds" : "CSCF_01, CSCF_02"
              }],
              "networks" : [
                     {
                     "id" : "NET2",
                     "shared" : "false",
                     "admin_state_up" : "true ",
                     "external" : "false",
                     "provider-physical_network" : "",
                     "provider-network_type" : "",
                     "provider-segmentation_id" : "",
                     "VIM" : {
                                   "intanceid" : "Network ID inside 
VIM",
                                   "instancename" : "Network NAME 
inside VIM"
                     },
                     "subnetworks" : [{
                            "type" : "ipv4 ",
                            "id" : "subnet1",
                            "ip" : "192.168.1.0",
                            "mask" : "24",
                            "enable_dhcp" : "true",
                            "gateway_ip" : "192.168.1.1",
                            "allocation pools" : "",
                            "VIM" : {
                                   "intanceid" : "Network ID inside 
VIM",
                                   "instancename" : "Network NAME 
inside VIM"
                            },
                     }]
              }]
       }
}
"grantApprovalDescriptor": {
       "vnf": {
              "vnfInstantiationPossible": "true",
              "id": "IMS_01",
              "flavorid" : "Gold",
              "vnfInstantiate": {
                     "vim": {
                            "id": "IMSVIM",
                            "tenant id": "TENANT 1",
                            "Credentials":{
                                   "username": "nfvoadmin",
                                   "password": "Passw0rd432"
                            }
                     }
              }
       }
}
}
```

7.2.1.4 Grant request

Grant Request from VNFManager

VNF Manager need to query NFVD for details of resource that can be used to instantiate the VNF, to do it Manager invoke this operation:

URL: NFV Director northbound Interface (informed on register process)

Http basic authentication with user and password provided on register process

Http operation: PUT

Path: v1/vnfdescriptor/<vnfdescriptorid>/grant (v1/vnfdescriptor/ID_Descriptor/grant)

Request Json must include next tags:

- vnfdescriptorid
- flavor:
- Descriptor version
- vnfid:
- grantApprovalDescriptor: Optionally, resources proposed by Manager. For this release NFVD will ignore it and recalculate optimum resources to allocate the VNF

Example:

```
{
"vnfdescriptorid" : "ID_Descriptor",
"vnfid" : "IMS 01",
"flavor" : "Gold",
"vnfdescriptorversion": "1.0",
"grantApprovalDescriptor": {
        "vnf": {
                "vnfInstantiationPossible": "true",
                "id": "IMS_01",
                "flavorid" : "Gold",
                "vnfInstantiate": {
                        "vim": {
                                "id": "IMSVIM",
                                "tenant id": "TENANT_1",
                                "Credentials":{
                                        "username": "nfvoadmin",
                                        "password": "Passw0rd432"
                                }
                        }
                }
        }
}
}
```

The JSON response example may be:

```
{
"description": "......",
"jobId": "1001",
"link":http: //<machine>: <port>/v1/jobs/<job_id> "
“entityId” : “IMS_01”
“entitylink” : “http: //<machine>: <port>/v1/vnf/IMS_01/grant”
}
```

Pooling for Grant

VNF manager need to query periodically for job status to NFVD with these parameters:

URL: NFV Director northbound Interface (informed on register process)

Http basic authentication with user and password provided on register process

Http operation: GET

Path: v1/jobs/<job_id> (/v1/jobs/1001)

The JSON response example may be:

```
{
"jboId": "1001",
"jboStatus": "Finished",
"jobDescription": "service grant job",
"statusDescription": "Grant is posted into NFVD"
}
```

Note: Final status must be “Error” or “Finished”

Retrieve Grant details

When grant job is finish VNFManager may query for details of VNF resources granted, to do that VNFManager will invoke this operation:

URL: NFV Director northbound Interface (informed on register process)

Http basic authentication with user and password provided on register process

Http operation: GET

Path: v1/vnf/<vnf_id>/grant (/v1/vnf/IMS_01/grant)

The JSON response example will be:

```
{
"vnfdescriptorid" : "ID_Descriptor",
"vnfid" : "IMS_01",
"flavor" : "Gold",
"vnfdescriptorversion": "1.0",
"grantApprovalDescriptor": {
       "vnf": {
              "vnfInstantiationPossible": "true",
              "id": "IMS_01",
              "flavorid" : "Gold",
              "vnfInstantiate": {
                     "vim": {
                            "id": "IMSVIM",
                            "tenant id": "TENANT_1",
                            "Credentials":{
                                   "username": "nfvoadmin",
                                   "password": "Passw0rd432"
                            }
                     }
              }
       }
}
}
```

7.2.1.5 Update Status

VNFManager could be update the entire VNF status in any time or invoke fail update to notify that something was wrong.

URL: NFV Director northbound Interface (informed on register process)

Http basic authentication with user and password provided on register process

Http operation: PUT

Path: v1/vnf/<vnfid>/vnfstatus (v1/vnf/IMS_01/vnfstatus)

The JSON response example will be:

```
{
"id": "IMS 01",
"version": "1.0",
"noOfVDUElements": 1,
"status": "deployed",
"VDUStatusDescripter": [{
       "unitId": "vdu1",
       "status": "active"
       "instances": [
              {
                     "id": "CSCF 01", "status": "poweredOn"
              },
              {
                     "id": "CSCF 01", "status": "poweredOff"
              }
       ]
}]
}
```

7.2.2 Implementation

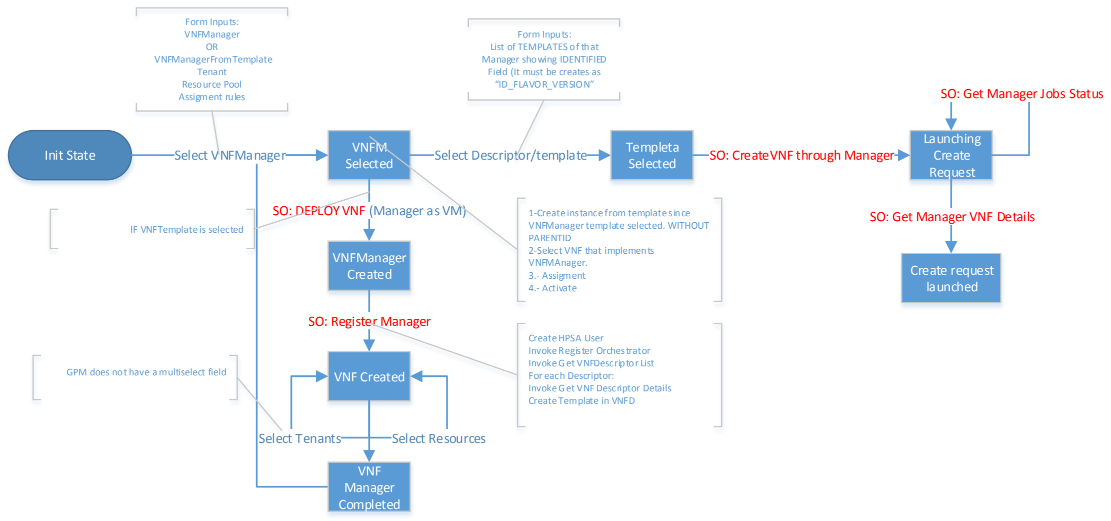

Form Inputs:
VNFManager
OR
VNFManagerFromTemplate
Tenant
Resource Pool
Assigment rules
Form Inputs:
List of TEMPLATES of that Manager showing IDENTIFIED Field (It must be creates as "ID_FLAVOR_VERSION"
SO: Get Manager Jobs Status
Init State
Select VNFManager
VNFM Selected
Select Descriptor/template
Templeta Selected
SO: Create VNF through Manager
Launching Create Request
SO: Get Manager VNF Details
Create request launched
IF VNFTemplate is selected
SO: DEPLOY VNF (Manager as VM)
VNFManager Created
1-Create instance from template since VNFManager template selected. WITHOUT PARENTID
2-Select VNF that implements VNFMAnager.
3.- Assigment
4.- Activate
SO: Register Manager
Create HPSA User
Invoke Register Orchestrator
Invoke Get VNFDescriptor List
For each Descriptor:
Invoke Get VNF Descriptor Details
Create Template in VNFD
GPM does not have a multiselect field
VNF Created
Select Tenants
Select Resources
VNF Manager Completed

Figure 43 Implementation

This process starts throwing new operation over NVF Manager PA. When a manager will need to deploy and instantiate a vnf managed by self, VNF Manager will request to NFVD permission to create it and asking where VMs could be instantiated, depends of VNF Manager it could support host level details or VIM level to instantiate the VMs.

The request must contain information about VNF Manager (VNFManagerID) and descriptorID on URL path and a JSON related before. This JSON could contain a grant proposal, <u>but since V2 it will be ignored.</u>

When this request arrives to NFVD it will run a GPM process and follow this steps:

1. Select VNFManager: First action is a request to select a valid VNF Manager or for create it. The fields showed will be:
   - VNF Manager Instance ID (Text): ID of already existent VNF Manager on NFVD
   - VNF Template (Select): If not exist an instance of NVF, NFVD could create it, to do it the user must provide some data to create Instance and Activate a VNFManager, and this template is needed. By Default: VNFMANAGER_DEFAULT_EMPLATE
   - Tenant (Select): Name of tenant parent of new VNF instance
   - Resource Pool (Text): ID of Resource Pool or Data Center where VNF Manager will be deployed.
   - Assignment Rule (Select): Assignment Policy used in creation process. By Default: VNFMANAGER_DEFAULT_ASSIGMENT

Note: If VNF Manager Instance ID is provided, next fields will be ignores, and the form must be explain it.
If a VNFManager Instance is selected, next step will be 5 (skipping 2, 3 and 4).

VNFMANAGER_DEFAULT_TEMPLATE:

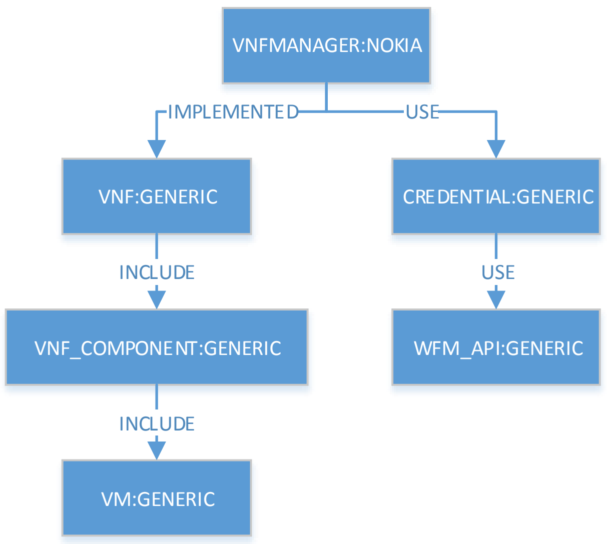


Figure 44 VNFMANAGER_DEFAULT_TEMPLATE

VNFMANAGER_DEFAULT_ASSIGMENT:

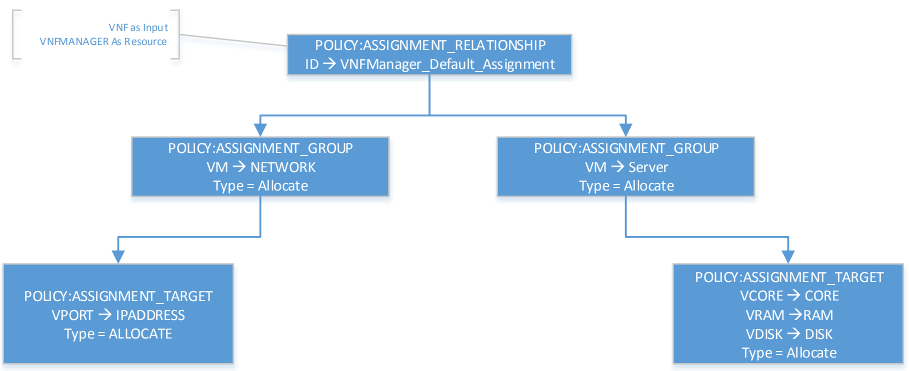


Figure 45 VNFMANAGER_DEFAULT_ASSIGMENT

2. Deploy VNF: This step will be an invocation of already New SO "NFVD-VNF-CREATE_AND_DEPLOY_MANAGER" with parameters selected on previous step:
    - VNF Template. By Default: VNFMANAGER_DEFAULT_EMPLATE
    - Tenant.
    - Resource Pool.
    - Assignment Rule. By Default: VNFMANAGER_DEFAULT_ASSIGMENT

Basically is similar execution of existing SO "NFVD-VNF-CREATE", but include a new SA between Instance creation and Assignment:

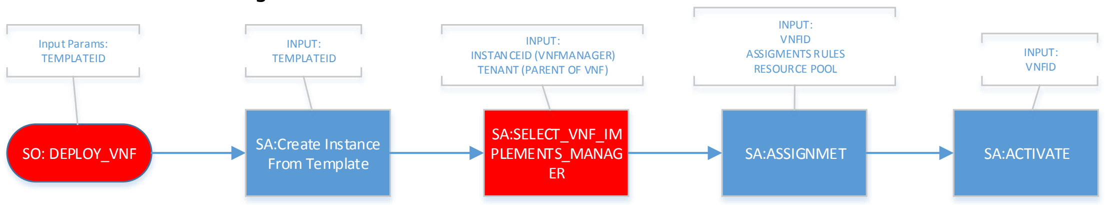


Figure 46 Deploy VNF

Workflow details on
<u>**VIEW V2.0-WORKFLOWS_VNFMANAGER.vsd → WF_NFVD_SELECT_VNF_IMPL_MANAGER**</u>

3. Register Manager: This new Service Order will be responsible of register process:
    - Create a user in HPSA for VNFManager
    - Create tree structure of VNF Manager Instance
    - Invoke Register Orchestrator on VNF (IP selected of VM instantiated on previous step)
    - Invoke Get VNF descriptor list
    - Invoke Get VNF Descriptor Details for each one.
    - Create a template for each Descriptor/version/flavor

Workflow details on **VIEW V2.0-WORKFLOWS_VNFMANAGER.vsd → WF_NFVD_GET_JSON_GRANT**

4. Assignment Tenants and Resources to a VNFManager, iterate over itself until no select or click to another button showed (review possibilities of GPM forms)

5. Select Descriptor. This action will be a form showed to the user to request this parameters:
    - Descriptor (Select): Mandatory. This field show all descriptors with its possibilities, to select uniquely a Template
    - VERSION
    - FLAVOR
    - Tenant (Select): Optional. A list of Tenant that the Manager can use. If no select one, means that could be used anyone of these.
    - Resource Pool (Select): Optional. A list of Resources that the Manager can use. If no select one, means that could be used anyone of these.

6. Create VNF Through Manager: This new Service Order will be responsible for starting the Instantiate VNF process:
    - Invoke Create VNF over a Manager with vnfdescriptorid, flavor, vnfdescriptorversion, vnfDetailsDescriptor in JSON request.

Workflow details on
VIEW V2.0-WORKFLOWS_VNFMANAGER.vsd → WF_NFVD_CREATE_VNFDESC_MANAGER

7. Get Manager Job Status: This will be a SO that query periodically if creation Job has finished.

Workflow details on
VIEW V2.0-WORKFLOWS_VNFMANAGER.vsd → WF_NFVD_JOBS_STAUS_MANAGER

8. Get Manager VNF Details: This will be a new SO that query for details of VNF deployment to VNFManager.

Workflow details on
VIEW V2.0-WORKFLOWS_VNFMANAGER.vsd → WF_NFVD_GET_MANAGER_VNF_DETAILS

7.3 Delete a VNF

7.3.1 Implementation

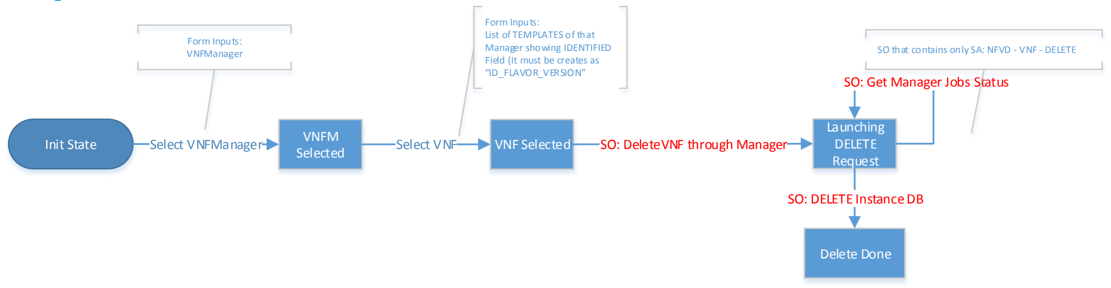

Form Inputs:
VNFManager
Form Inputs:
List of TEMPLATES of that Manager showing IDENTIFIED Field (It must be creates as "ID_FLAVOR_VERSION"
SO that contains only SA: NFVD - VNF - DELETE
SO: Get Manager Jobs Status
Init State
Select VNFManager
VNFM Selected
Select VNF
VNF Selected
SO: DeleteVNF through Manager
Launching DELETE Request
SO: DELETE Instance DB
Delete Done

Figure 47 Delete a VNF

7.4 Scale VNF

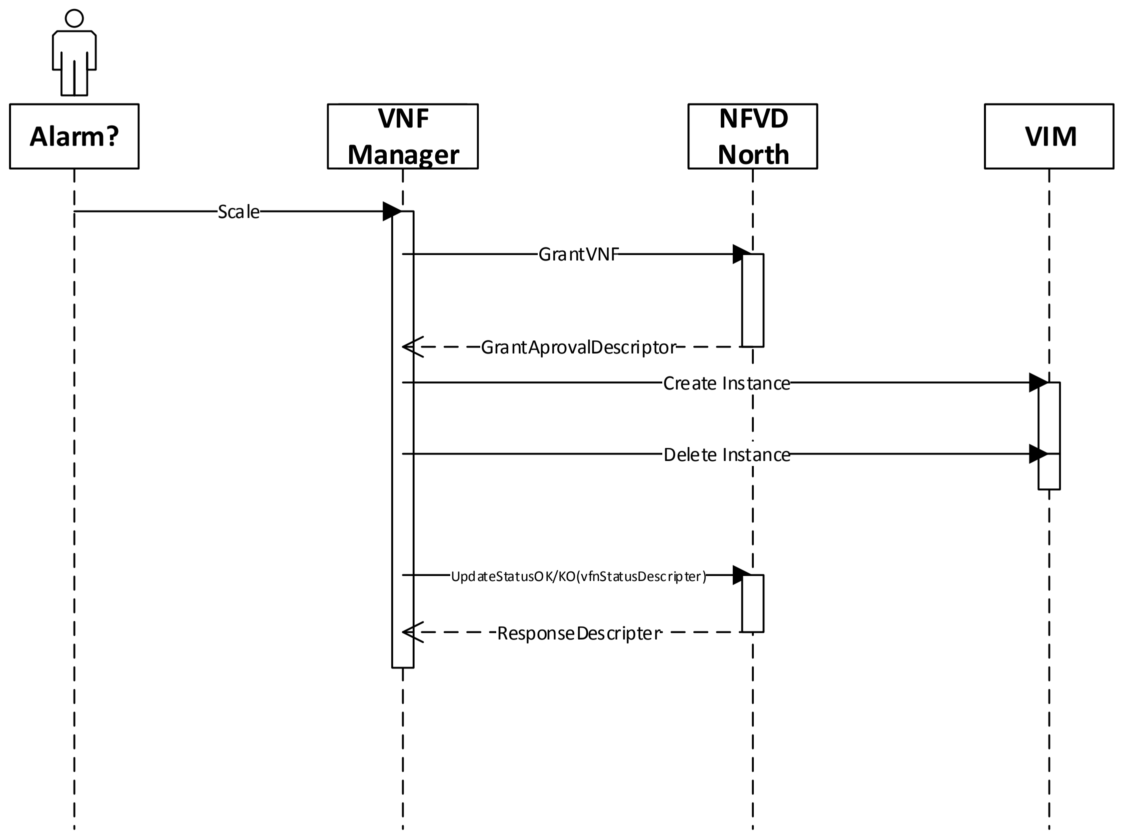


Figure 48 Scale VNF

7.4.1 Behavior

7.4.1.1 Grant Request from VNFManager

VNF Manager needs to query NFVD for details of resource that can be used to instantiate the VNF, to do it Manager invoke this operation:

URL: NFV Director northbound Interface (informed on register process)

Http basic authentication with user and password provided on register process

Http operation: PUT

Path: v1/vnf/<vnfid>/scale-out/grant (v1/IMS_01/scale_out/grant)

Request Json must include next tags:

- vnfdescriptorid
- flavor:
- Descriptor version
- vnfid:
- grantApprovalDescriptor: Optionally, resources proposed by Manager. For this release NFVD will ignore it and recalculate optimum resources to allocate the VNF

Example:

```
{
```

```
"vnfdescriptorid" : "ID_Descriptor",
"vnfid" : "IMS_01",
"flavor" : "Gold",
"vnfdescriptorversion": "1.0",
"grantApprovalDescriptor": {
        "vnf": {
                "vnfInstantiationPossible": "true",
                "id": "IMS_01",
                "flavorid" : "Gold",
                "vnfInstantiate": {
                        "vim": {
                                "id": "IMSVIM",
                                "tenant id": "TENANT_1",
                                "Credentials":{
                                        "username": "nfvoadmin",
                                        "password": "Passw0rd432"
                                }
                        }
                }
        }
}
}
```

The JSON response example may be:

```
{
"description": "......",
"jobId": "1002",
"link":http: //<machine>: <port>/v1/jobs/<job id> "
"entityId" : "IMS"
"entitylink" : "http: //<machine>: <port>/v1/vnf/IMS_01/grant"
}
```

7.4.1.2 Pooling for Grant

VNF manager need to query periodically for job status to NFVD with these parameters:

URL: NFV Director northbound Interface (informed on register process)

Http basic authentication with user and password provided on register process

Http operation: GET

Path: v1/jobs/<job_id> (/v1/jobs/1002)

The JSON response example may be:

```
{
"jboId": "1002",
"jboStatus": "Finished",
"jobDescription": "service grant job",
"statusDescription": "Grant is posted into NFVD"
}
```

Note: Final status must be “Error” or “Finished”

7.4.1.3 Retrieve Grant details

When grant job is finish VNFManager may query for details of VNF resources granted, to do that VNFManager will invoke this operation:

URL: NFV Director northbound Interface (informed on register process)

Http basic authentication with user and password provided on register process

Http operation: GET

Path: v1/vnf/<vnf_id>/grant (/v1/vnf/IMS_01/grant)

The JSON response example will be:

```
{
"vnfdescriptorid" : "ID Descriptor",
"vnfid" : "IMS_01",
"flavor" : "Gold",
"vnfdescriptorversion": "1.0",
"grantApprovalDescriptor": {
       "vnf": {
              "vnfInstantiationPossible": "true",
              "id": "IMS_01",
              "flavorid" : "Gold",
              "vnfInstantiate": {
                     "vim": {
                            "id": "IMSVIM",
                            "tenant id": "TENANT 1",
                            "Credentials":{
                                   "username": "nfvoadmin",
                                   "password": "Passw0rd432"
                            }
                     }
              }
       }
}
}
```

7.5 NORTHBOUND Interface

7.5.1 Operation details

7.5.1.1 VNFStatus OK notification

- Path: v1/vnfm/<vnfmanagerid>/vnf/<vnfid>/vnfstatus
- HTTP Operation: POST
- Normal Response Codes:
- Error Response Codes:
- Request JSON (vnfStatusDescriptor)

```
{
"id": "cscf_10.1VI",
"version": "x", "noOfVDUElements": 4, "status": "deployed",
"VDUStatusDescriptor": [
{
        "unitId": "cscf",
        "status": "active"
        "instances": [
                {
                        "id": "cscf-1", "status": "poweredOn"
                },
                {
                        "id": "cscf-2", "status": "poweredOff"
                }
        ]
},
{
        "unitId": "lb", "instances": [
                {
                        "id": "lb-1", "status": "poweredOn"
                },
                {
                        "id": "lb-2", "status": "poweredOff"
                }
        ]
}
]
}
```

- Response JSON (ResponseDescriptor)

```
{
"description": "......",
"jobId": "<job_id>",
"link": "http: //<machine>: <port>/v1/jobs/<job_id>"
"entityId" : "<vnfid>"
"entitylink" : "http: //<machine>: <port>/v1/vnf/<vnf_id>"
}
```

7.5.1.2 Behavior

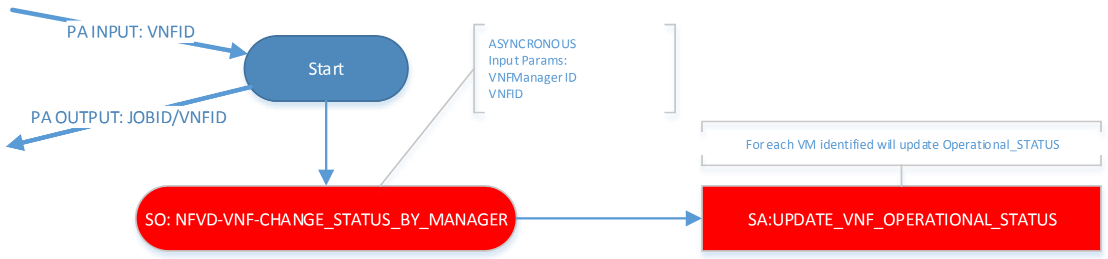

PA INPUT: VNFID
PA OUTPUT: JOBID/VNFID
Start
ASYNCRONOUS
Input Params:
VNFManager ID
VNFID
For each VM identified will update Operational_STATUS
SO: NFVD-VNF-CHANGE_STATUS_BY_MANAGER
SA:UPDATE_VNF_OPERATIONAL_STATUS

Figure 49 Behavior

This operation will launch a new SO named "NFVD-VNF-CHANGE_STATUS_BY_MANAGER" in Asynchronous mode.

For each VNF, VDU, VM detailed on request we will update attribute "STATUS. Operational_Status" with given value, and "STATUS. Operational_Status_Date" with current timestamp.

Workflow details on **VIEW V2.0-WORKFLOWS_VNFMANAGER.vsd → WF_NFVD_VNFM_UPDATE_STATUS**

7.5.1.3 VNFStatus Fail Notification

- Path: v1/vnfm/<vnfmanagerid>/vnf/<vnfid>/fault
- HTTP Operation: POST
- Normal Response Codes:
- Error Response Codes: (400, 500….)

computeFault (400, 500, …), UnprocessableEntity (422), serviceUnavailable (503), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415)

- Request JSON (vnfStatusDescriptor)

```
{
"vnfId": "cscf_10.1VI", "faultCode": "10001", "faultDescription":
"....", "VDUFaultDescripter": [
{
       "unitId": "cscf", "faultCode": "10001",
       "faultDescription": "Failed due to ..."
},
{
       "unitId": "lb", "faultCode": "10001", "faultDescription":
"...."
}
]
}
```

- Response JSON (ResponseDescriptor)

```
{
"description": "......",
"jobId": "<job_id>",
"link": "http: //<machine>: <port>/v1/jobs/<job_id>"
"entityId" : "<vnfid>"
"entitylink" : "http: //<machine>: <port>/v1/vnf/<vnf_id>"
}
```

7.5.1.4 Behavior

Same behavior as change status.

7.5.1.5 Grant VNF Creation

- Path: v1/vnfm/<vnfmanagerid>/vnfdescriptor/<vnfdescriptorid>/grant
- HTTP Operation: PUT
- Normal Response Codes:
- Error Response Codes:
- Request JSON :
    - vnfdescriptorid
    - flavor:
    - Descriptor version
    - vnfid:

- grantApprovalDescriptor: Optionally, resources proposed by Manager. For this release NFVD will ignore it and recalculate optimum resources to allocate the VNF

```
{
"vnfdescriptorid" : "ID Descriptor",
"vnfid" : "IMS 01",
"flavor" : "Gold",
"vnfdescriptorversion": "1.0",
"grantApprovalDescriptor": {
        "vnf": {
                "vnfInstantiationPossible": "true",
                "id": "IMS_01",
                "flavorid" : "Gold",
                "vnfInstantiate": {
                        "vim": {
                                "id": "IMSVIM",
                                "tenant id": "TENANT_1",
                                "Credentials":{
                                        "username": "nfvoadmin",
                                        "password": "Passw0rd432"
                                }
                        }
                }
        }
}
}
```

- Response JSON (ResponseDescriptor)

```
{
"description": "......",
"jobId": "<job_id>",
"link": “http: //<machine>: <port>/v1/jobs/<job_id>"
“entityId” : “<vnfid>”
“entitylink” : “http: //<machine>: <port>/v1/vnf/<vnf_id>”
}
```

7.5.1.6 Behavior

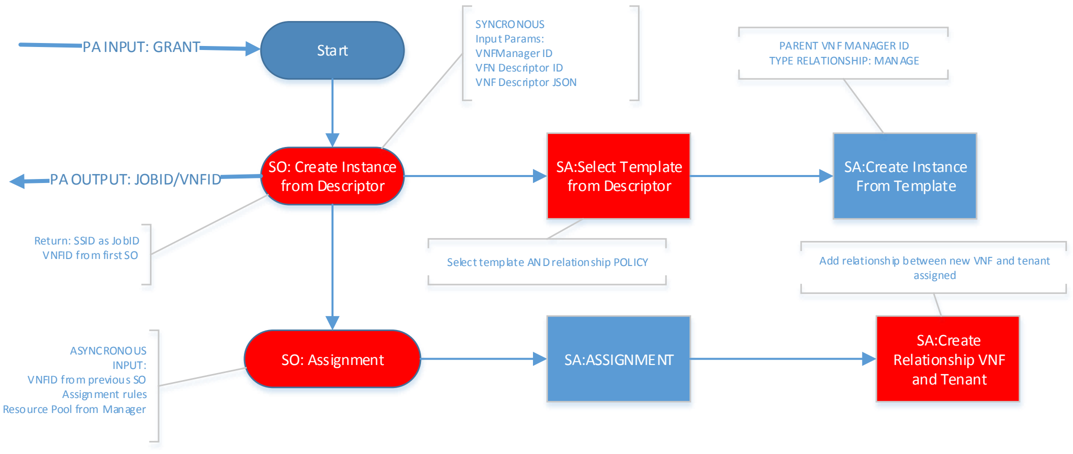

PA INPUT: GRANT
Start
SYNCRONOUS
Input Params:
VNFManager ID
VFN Descriptor ID
VNF Descriptor JSON
PARENT VNF MANAGER ID
TYPE RELATIONSHIP: MANAGE
PA OUTPUT: JOBID/VNFID
SO: Create Instance from Descriptor
SA:Select Template from Descriptor
SA:Create Instance From Template
Return: SSID as JobID
VNFID from first SO
Select template AND relationship POLICY
Add relationship between new VNF and tenant assigned
ASYNCRONOUS
INPUT:
VNFID from previous SO
Assignment rules
Resource Pool from Manager
SO: Assignment
SA:ASSIGNMENT
SA:Create Relationship VNF and Tenant

Figure 50 Creating new Workflow Templates

When the NFVD get a Grant request, The PA first will launch a Synchronous SO to create an instance tree of request descriptor, and when this SO finished could get VNFID inside the response.

Then a new SO will launched in Asynchronous mode, and at this point the PA can responds with SSID of this new SO as “jobId” and VNFID as “entityId”.

Workflow Select Template from Descriptor
VIEW V2.0-WORKFLOWS_VNFMANAGER.vsd → WF_NFVD_SELECT_TEMPLATE_FROM_DESCRIPTOR

VNFMANAGER_DEFAULT_ASSIGMENT:

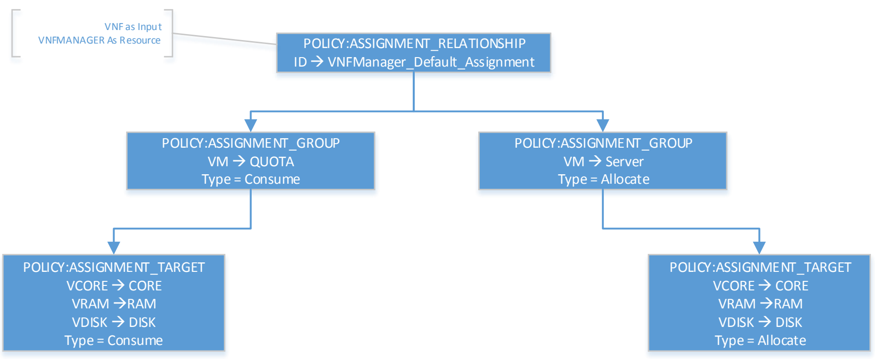


Figure 51 VNFMANAGER_DEFAULT_ASSIGMENT:

Workflow Select Template from Descriptor
VIEW V2.0-WORKFLOWS_VNFMANAGER.vsd → WF_NFVD_RELATE_VNF_TO_TENANT

7.5.1.7 Grant Scale

7.5.1.8 Scale_out

- Path for VNF: v1/vnfm/<vnfmanagerid>/vnf/<vnfid>/scale-out/grant
- Path for VDU: v1/vnfm/<vnfmanagerid>/vnf/<vnfid>/vdu/<vduid>/scale-out/grant

7.5.1.9 Scale_in

- Path for VNF: v1/vnfm/<vnfmanagerid>/vnf/<vnfid>/scale-in/grant
- Path for VDU: v1/vnfm/<vnfmanagerid>/vnf/<vnfid>/vdu/<vduid>/scale-in/grant

7.5.1.10 Scale_up

- Path for VNF: v1/vnfm/<vnfmanagerid>/vnf/<vnfid>/scale-up/grant
- Path for VDU: v1/vnfm/<vnfmanagerid>/vnf/<vnfid>/vdu/<vduid>/scale-up/grant

7.5.1.11 Scale_down

- Path for VNF: v1/vnfm/<vnfmanagerid>/vnf/<vnfid>/scale-down/grant
- Path for VDU: v1/vnfm/<vnfmanagerid>/vnf/<vnfid>/vdu/<vduid>/scale-down/grant

7.5.1.12 Common parameters

- HTTP Operation: PUT
- Normal Response Codes:
- Error Response Codes:

- Request JSON must include next tags:
    - vnfdescriptorid
    - flavor:
    - Descriptor version
    - vnfid:
    - grantApprovalDescriptor: Optionally, resources proposed by Manager. For this release NFVD will ignore it and recalculate optimum resources to allocate the VNF
    - Example:

```
{
"vnfdescriptorid" : "ID_Descriptor",
"vnfid" : "IMS 01",
"flavor" : "Gold",
"vnfdescriptorversion": "1.0",
"grantApprovalDescriptor": {
        "vnf": {
                "vnfInstantiationPossible": "true",
                "id": "IMS_01",
                "flavorid" : "Gold",
                "vnfInstantiate": {
                        "vim": {
                                "id": "IMSVIM",
                                "tenant id": "TENANT_1",
                                "Credentials":{
                                        "username": "nfvoadmin",
                                        "password": "Passw0rd432"
                                }
                        }
                }
        }
}
}
```

- The JSON response example may be:

```
{
"description": "......",
"jobId": "1002",
"link":http: //<machine>: <port>/v1/jobs/<job_id> "
"entityId" : "IMS"
"entitylink" : "http: //<machine>: <port>/v1/vnf/IMS_01/grant"
}
```

7.5.1.13 Behavior

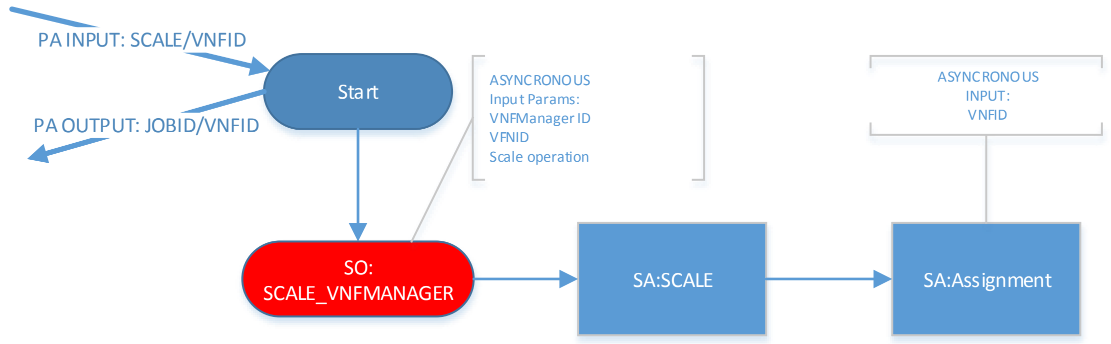


Figure 52 Behavior

On any kind of Scale, NFVD do always the same, starting a new SO that includes current SA to scale and assignment.

7.5.1.14 Get Grant details

Path: v1/vnfm/<vnfmanagerid>/vnf/<vnf_id>/grant

- HTTP Operation: GET
- Normal Response Codes:
- Error Response Codes:
- Request JSON :
- Response JSON (grantApprovalDescriptor)

ALL assigned to VIM:

```
{
"vnfdescriptorid" : "ID_Descriptor",
"vnfid" : "IMS 01",
"flavor" : "Gold",
"vnfdescriptorversion": "1.0",
"grantApprovalDescriptor": {
       "vnfd": {
              "vnfInstantiationPossible": "true",
              "id": "IMS",
              "flavorid" : "Gold",
              "vnfInstantiate": {
                     "vim": {
                            "id": "IMSVIM",
                            "tenant id": "<ID of tenant>",
                            "Credentials":{
                                   "username": "nfvoadmin",
                                   "password": "Passw0rd432"
                            }
                     }
              }
       }
}
}
```

Assigned VM-Availability Zone (Hostname optional):

```
{
"vnfdescriptorid" : "ID_Descriptor",
"vnfid" : "IMS_01",
"flavor" : "Gold",
"vnfdescriptorversion": "1.0",
"grantApprovalDescriptor": {
       "vnfd": {
              "vnfInstantiationPossible": "true",
              "id": "IMS",
              "flavorid" : "Gold",
              "vdus":[{
                     "id" : "vdu1"
                     "vms":[ {
                            "id" : "CSCF_01",
                            "vmInstantiate": {
                                   "vim": {
                                          "id": "IMSVIM",
                                          "tenant id": "<ID of tenant>",
                                          "Credentials":{
                                                 "username":
"nfvoadmin",
                                                 "password":
"Passw0rd432"
                                          }
                                          "availability_zone" : {
                                                 "name" : "AZ1",
                                                 "host" :
"Hypervisor_hostname"
                                          }
                                   }
                            }
                     }]
              }]
       }
}
}
```

7.5.1.15 Behavior

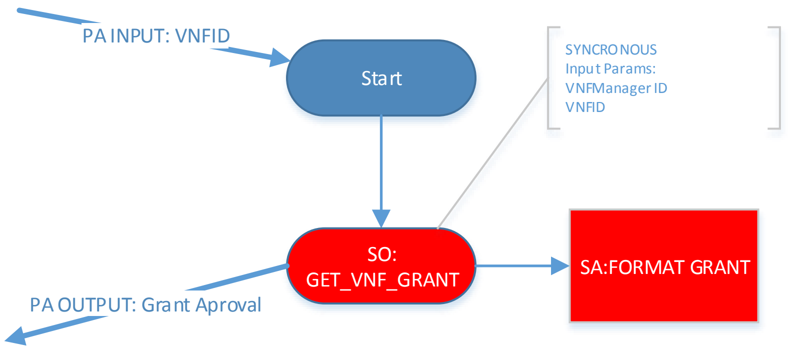


Figure 53 Creating new Workflow Templates

When a request to get Grant Approval is received by NFVD, the PA will execute a new SO "GET_VNF_GRANT", this SO/SA will launch a workflow that translate the information of allocation on grantApprovalDescriptor format.

VIEW V2.0-WORKFLOWS_VNFMANAGER.vsd → WF_NFVD_GET_JSON_GRANT

7.5.1.16 Get Job Status

- Path: v1/vnfm/<vnfmanagerid>/jobs/<jobid>
- HTTP Operation: GET
- Normal Response Codes:
- Error Response Codes:
- Request JSON () N/A
- Response JSON (ResponseDescriptor)

```
{
"jobId": "1001001",
"jobStatus": "Started",
"jobDescription": "Status change job",
"statusDescription": "Request is posted into NFVD"
}
```

7.5.1.17 Behavior

For this request NFVD only need to query SOSA with SO status using Jobid as SSID

7.6 SOUTHBOUND Interface

7.6.1 Operation details

7.6.1.1 Create VNF

- Path: v1/vnfdescriptor/<vnfdescriptorid>
- HTTP Operation: POST
- Normal Response Codes:
- Error Response Codes:
- Request JSON
    - Vnfdescriptorid Mandatory
    - Vnf descriptor Version
    - Flavor Mandatory if Descriptor contains flavors
    - vnfDetailsDescriptor Optional
    - grantApprovalDescriptor Optional

```
{
"vnfdescriptorid" : "ID Descriptor",
"flavor" : "Gold",
"vnfdescriptorversion" : "1.0",
"vnfDetailsDescriptor": {
        "vnfdescriptorid" : "ID Descriptor",
        "vnf":[ {
                "version": "1.5",
                "templateId":"IMS CSCF",
                "lowlevel_assigment_supported" :
"vim|region|availability_zone|host"
                "vdus":[{
                        "id" : "vdu1",
                        "vms":[ {
                                "id" : "CSCF_01",
                                "status" : "poweredon",
                                "Resources": {
                                        "noOfCPUs": {
                                                "Amount" : "3"
                                        },
                                        "memoryinMB": {
                                                "Amount" : "10240"
                                        },
                                        "diskinGb": {
                                                "Amount" : "100"
                                        },
                                        "luns": [{
                                                "id" : "lun1",
                                                "Amount" : "3"
                                        }]
                                },
                                "Networks":[{
                                        "id": "NET1",
                                        "IP": "1.1.1.1"
                                        },
                                        {"id": "NET2"},
                                        {"id": "NET3"}
                                ]
                        },
                        {
                                "id" : "CSCF_02",
                                "status" : "poweredon",
                                "Resources": {
                                        "noOfCPUs": {
                                                "Amount" : "3"
                                        },
                                        "memoryinMB": {
                                                "Amount" : "10240"
                                        },
                                        "diskinGb": {
                                                "Amount" : "100"
                                        },
                                        "luns": [{
                                                "id" : "lun1",
                                                "Amount" : "3"
                                        }]
                                },
                                "Networks":[{
                                        "id": "NET1",
                                        "IP": "1.1.1.2"
                                        },
                                        {"id": "NET2"},
                                        {
                                        "id": "NET3"
                                        }
                                ]
                        }]
                }]
                "flavors" : [{
                        "id" : "Gold",
                        "includedVDUIds" : "vdu1"
                        "includedVMIds" : "CSCF 01, CSCF 02"
                },
                {
                        "id" : "Silver",
                        "includedVDUIds" : "vdu1"
                        "includedVMIds" : "CSCF_01"
```

```
            }],
            "networks" : [
                  {
                  "id" : "NET1",
                  "shared" : "true|false",
                  "admin_state_up" : "true|false",
                  "external" : "true|false",
                  "provider-physical_network" : "",
                  "provider-network_type" : "flat|vlan|vxlan|gre",
                  "provider-segmentation_id" : "",
                  "subnetworks" : [{
                        "type" : "ipv4|ipv6",
                        "id" : "subnet1",
                        "ip" : "1.1.1.0",
                        "mask" : "24",
                        "enable_dhcp" : "true|false",
                        "gateway_ip" : "1.1.1.1",
                        "allocation_pools" : ""
                        }]
                  },
                  {
                  "id" : "NET2",
                  "shared" : "true|false",
                  "admin_state_up" : "true|false",
                  "external" : "true|false",
                  "provider-physical_network" : "",
                  "provider-network_type" : "flat|vlan|vxlan|gre",
                  "provider-segmentation id" : "",
                  "subnetworks" : [{
                        "type" : "ipv4|ipv6",
                        "id" : "subnet1",
                        "ip" : "192.168.1.0",
                        "mask" : "24",
                        "enable_dhcp" : "true|false",
                        "gateway_ip" : "192.168.1.1",
                        "allocation pools" : ""
                        }]
            }]
      }]
} ,
"grantApprovalDescriptor": {
      "vnf": {
            "vnfInstantiationPossible": "true",
            "id": "IMS",
            "flavorid" : "Gold",
            "vnfInstantiate": {
                  "vim": {
                        "id": "IMSVIM",
                        "tenant id": "<ID of tenant>",
                        "Credentials":{
                              "username": "nfvoadmin",
                              "password": "Passw0rd432"
                        }
                  }
            }
      }
}
}
```

- Response JSON (ResponseDescripter)

```
{
"description": "......",
"jobId": "<job_id>",
"link":http: //<machine>: <port>/v1/jobs/<job_id> "
"entityId" : "<vnfid>"
"entitylink" : "http: //<machine>: <port>/v1/vnf/<vnf_id>"
}
```

7.6.1.2 Delete VNF

- Path: v1/vnf/<vnfid>
- HTTP Operation: DELETE
- Normal Response Codes:
- Error Response Codes:
- Request JSON () N/A
- Response JSON (ResponseDescripter)

```
{
"description": "......",
"jobId": "<job_id>",
"link":http: //<machine>: <port>/v1/jobs/<job_id> "
"entityId" : "<vnfid>"
"entitylink" : "http: //<machine>: <port>/v1/vnf/<vnf_id>"

}
```

7.6.1.3 Create VNF Descriptor

- Path: v1/vnfdescriptor
- HTTP Operation: POST
- Normal Response Codes:
- Error Response Codes:
- Request JSON
    - Vnfdescriptorid Mandatory
    - Vnf descriptor Version
    - Flavor Mandatory if Descriptor contains flavors
    - vnfDetailsDescriptor Optional
    - grantApprovalDescriptor Optional

```
{
"vnfdescriptorid" : "ID_Descriptor",
"flavor" : "Gold",
"vnfdescriptorversion" : "1.0",
"vnfDetailsDescriptor": {
       "vnfdescriptorid" : "ID_Descriptor",
       "vnf":[ {
              "version": "1.5",
              "templateId":"IMS_CSCF",
              "lowlevel assigment supported" : "vim"
              "vdus":[{
                     "id" : "vdu1",
                     "vims":[ {
                            "id" : "CSCF_01",
                            "status" : "poweredon",
                          "image" : "image name used to instantiate
VM on VIM",
                          "flavor" : "flavor of VM used",

                          "Resources": {
                                "noOfCPUs": {
                                      "Amount" : "3"
                                },
                                "memoryinMB": {
                                      "Amount" : "10240"
                                },
                                "diskinGb": {
                                      "Amount" : "100"
                                },
                                "luns": [{
                                      "id" : "lun1",
                                      "Amount" : "3"
                                }]
                          },
                          "Networks":[{
                                "id": "NET2"
                          }]
                    },
                    {
                          "id" : "CSCF_02",
                          "status" : "poweredon",
                          "Resources": {
                                "noOfCPUs": {
                                      "Amount" : "3"
                                },
                                "memoryinMB": {
                                      "Amount" : "10240"
                                },
                                "diskinGb": {
                                      "Amount" : "100"
                                },
                                "luns": [{
                                      "id" : "lun1",
                                      "Amount" : "3"
                                }]
                          },
                          "Networks":[
                                {"id": "NET2"}
                          ]
                    }]
              }]
              "flavors" : [{
                    "id" : "Gold",
                    "includedVDUIds" : "vdu1"
                    "includedVMIds" : "CSCF_01, CSCF_02"
              }],
              "networks" : [
                    {
                    "id" : "NET2",
                    "shared" : "false",
                    "admin_state_up" : "true ",
                    "external" : "false",
                    "provider-physical network" : "",
                    "provider-network_type" : "",
                    "provider-segmentation_id" : "",
                    "subnetworks" : [{
                          "type" : "ipv4 ",
                          "id" : "subnet1",
                           "ip" : "192.168.1.0",
                           "mask" : "24",
                           "enable_dhcp" : "true",
                           "gateway_ip" : "192.168.1.1",
                           "allocation_pools" : ""
                           }]
             }]
       }]
}
}
```

- Response JSON (ResponseDescripter)

```
{
"description": "......",
"jobId": "<job_id>",
"link":http: //<machine>: <port>/v1/jobs/<job_id> "
"entityId" : "<vnfid>"
"entitylink" : "http: //<machine>:
<port>/v1/vnfdescriptor/<ID_Descriptor>"
}
```

7.6.1.4 Get Job Status

- Path: v1/jobs/<jobid>
- HTTP Operation: GET
- Normal Response Codes:
- Error Response Codes:

- Request JSON () N/A

- Response JSON (ResponseDescripter)

```
{
"jobId": "1001001",
"jobStatus": "Started",
"jobDescription": "service creation job",
"statusDescription": "Request is posted into VNFM"
}
```

7.6.1.5 Get VNFDescriptor list

- Path: v1/vnfdescriptor/
- Parameters:
  - flavor
  - vnfDescriptorVersion
- HTTP Operation: GET
- Normal Response Codes:
- Error Response Codes:

- Request JSON () N/A
- Response JSON (vnfdescriptors)

```
{
"vnfdescriptors": [
{
       "Id": "vnf Descriptor ID 1",
},
{
       "Id": " vnf Descriptor ID2",
}
]
}
```

7.6.1.6 Get VNFDescriptor Details

- Path: v1/vnfdescriptor/<vnfdescriptorID>
- HTTP Operation: GET
- Normal Response Codes:
- Error Response Codes:
- Request JSON () N/A
- Response JSON (vnfDetailsDescriptor)

```
{
"vnfDetailsDescriptor": {
       "vnfdescriptorid" : "ID Descriptor",
       "vnf":[ {
              "version": "1.5",
              "templateId":"IMS_CSCF",
              "lowlevel assigment supported" :
"vim|region|availability_zone|host"
              "vdus":[{
                     "id" : "vdu1",
                     "scale" : {
                            "default" : "1",
                            "scale out" : "2",
                            "scale_in" : "1"
                            "min"
                            "max",
                            "mandatory" : "must|should"
                     }
                     "vms":[ {
                            "id" : "CSCF_01",
                            "status" : "poweredon",
                                   "scale" : {
                                   "default" : "1",
                                   "scale_out" : "2",
                                   "scale_in" : "1"
                                   "min"
                                   "max",
                                   "mandatory" : "must|should"
                            },
                            "Resources": {
                                   "noOfCPUs": {
                                          "Amount" : "3",
                                          "scale" : {
                                                 "default" : "1",
                                                 "scale_out" : "2",
                                                 "scale_in" : "1"
                                                 "min"
                                                 "max",
                                                 "mandatory" :
"must|should"

                                          }
                            }
                                   "memoryinMB": {
                                          "Amount" : "10240",
                                          "scale" : {
                                                 "default" : "1",
                                                 "scale_out" : "2",
                                                 "scale_in" : "1"
                                                 "min"
                                                 "max",
                                                 "mandatory" :
"must|should"

                                          }
                            }
                                   "diskinGb": {
                                          "Amount" : "100",
                                          "scale" : {
                                                 "default" : "1",
                                                 "scale out" : "2",
                                                 "scale_in" : "1"
                                                 "min"
                                                 "max",
                                                 "mandatory" :
"must|should"

                                          }
                            }
                                   "luns": [{
                                          "id" : "lun1",
                                          "Amount" : "3",
                                          "scale" : {
                                                 "default" : "1",
                                                 "scale out" : "2",
                                                 "scale_in" : "1"
                                                 "min"
                                                 "max",
                                                 "mandatory" :
"must|should"

                                          }
                            }]

                            },
                            "Networks":[{
                                   "id": "NET1",
                                   "IP": "1.1.1.1"
                                   },
                                   {"id": "NET2"},
                                   {
                                   "id": "NET3"
                                   }
                            ]
                     }]
              }]
```

```
                "flavors" : [{
                        "id" : "Gold",
                        "includedVDUIds" : "vdu1"
                        "includedVMIds" : "CSCF_01, CSCF_02"
                },
                {
                        "id" : "Silver",
                        "includedVDUIds" : "vdu1"
                        "includedVMIds" : "CSCF_01"
                }],
                "networks" : [
                        {
                        "id" : "NET1",
                        "shared" : "true|false",
                        "admin_state_up" : "true|false",
                        "external" : "true|false",
                        "provider-physical_network" : "",
                        "provider-network_type" : "flat|vlan|vxlan|gre",
                        "provider-segmentation_id" : "",
                        "subnetworks" : [{
                                "type" : "ipv4|ipv6",
                                "id" : "subnet1",
                                "ip" : "1.1.1.0",
                                "mask" : "24",
                                "enable_dhcp" : "true|false",
                                "gateway_ip" : "1.1.1.1",
                                "allocation_pools" : ""
                                }]
                        },
                        {
                        "id" : "NET2",
                        "shared" : "true|false",
                        "admin state up" : "true|false",
                        "external" : "true|false",
                        "provider-physical_network" : "",
                        "provider-network type" : "flat|vlan|vxlan|gre",
                        "provider-segmentation_id" : "",
                        "subnetworks" : [{
                                "type" : "ipv4|ipv6",
                                "id" : "subnet1",
                                "ip" : "192.168.1.0",
                                "mask" : "24",
                                "enable dhcp" : "true|false",
                                "gateway_ip" : "192.168.1.1",
                                "allocation_pools" : ""
                                }]
                }]
        }]
}
}
```

7.6.1.7 Get VNF list

- Path: v1/vnf
- Parameters:
    - flavor
    - vnfDescriptorId
    - vnfDescriptorVersion

- HTTP Operation: GET
- Normal Response Codes:
- Error Response Codes:

- Request JSON () N/A

- Response JSON (vnfs)

```
{
"vnfs": [
{
       "Id": "vnf ID 1",
},
{
       "Id": "vnfId 2",
}
]
}
```

7.6.1.8 Get VNF Details

- Path: : v1/vnf/<vnfId>
- HTTP Operation: GET
- Normal Response Codes:
- Error Response Codes:
- Request JSON () N/A
- Response JSON (vnfDeploymentDescriptor and GrantAprovalDescriptor)

```
{
"vnfDeploymentDescriptor": {
       "flavor" : "Gold",
       "version": "1.5",
       "vnfdescriptorid" : "ID_Descriptor",
       "vnf":{
              "vnfid" : "IMS_01",
              "templateId":"IMS_CSCF",
              "lowlevel_assigment_supported" : "vim"
              "vdus":[{
                     "id" : "vdu1",
                     "vms":[ {
                            "id" : "CSCF_01",
                            "status" : "poweredon",
                            "hostname" : "vm.hp.com",
                            "image" : "image name used to instantiate 
VM on VIM",
                            "flavor" : "flavor of VM used",
                            "VIM" : {
                                   "intanceid" : "Artifact ID inside 
VIM",
                                   "instancename" : "Artifact NAME 
inside VIM"
                            },
                            "HYPERVISOR" : {
                                   "intanceid" : "Artifact ID inside
Hypervisor",
                                   "instancename" : "Artifact NAME
inside Hypervisor"
                            }
                            "Resources": {
                                   "noOfCPUs": {
                                          "Amount" : "3"
                                   },
                                   "memoryinMB": {
                                          "Amount" : "10240"
                                   },
                                   "diskinGb": {
                                          "Amount" : "100"
                                   },
                                   "luns": [{
                                          "id" : "lun1",
                                          "Amount" : "3"
                                   }]
                            },
                            "Networks":[{
                                   "id": "NET2"
                            }]
                     },
                     {
                            "id" : "CSCF_02",
                            "status" : "poweredon",
                            "hostname" : "vm.hp.com",
                            "image" : "image name used to instantiate
VM on VIM",
                            "flavor" : "flavor of VM used",
                            "VIM" : {
                                   "intanceid" : "VM ID inside VIM",
                                   "instancename" : "VM NAME inside
VIM"
                            },
                            "HYPERVISOR" : {
                                   "intanceid" : "VM ID inside
Hypervisor",
                                   "instancename" : "VM NAME inside
Hypervisor"
                            }
                            "Resources": {
                                   "noOfCPUs": {
                                          "Amount" : "3"
                                   },
                                   "memoryinMB": {
                                          "Amount" : "10240"
                                   },
                                   "diskinGb": {
                                          "Amount" : "100"
                                   },
                                   "luns": [{
                                          "id" : "lun1",
                                          "Amount" : "3"
                                   }]
                            },
                            "Networks":[
                                   {"id": "NET2"}
                            ]
                     }]
              }]
              "flavors" : [{
                     "id" : "Gold",
                     "includedVDUIds" : "vdu1"
                     "includedVMIds" : "CSCF_01, CSCF_02"
              }],
              "networks" : [
                     {
                     "id" : "NET2",
                     "shared" : "false",
                     "admin_state_up" : "true ",
                     "external" : "false",
                     "provider-physical_network" : "",
                     "provider-network_type" : "",
                     "provider-segmentation_id" : "",
                     "VIM" : {
                                   "intanceid" : "Network ID inside
VIM",
                                   "instancename" : "Network NAME
inside VIM"
                     },
                     "subnetworks" : [{
                            "type" : "ipv4 ",
                            "id" : "subnet1",
                            "ip" : "192.168.1.0",
                            "mask" : "24",
                            "enable_dhcp" : "true",
                            "gateway_ip" : "192.168.1.1",
                            "allocation_pools" : "",
                            "VIM" : {
                                   "intanceid" : "Network ID inside
VIM",
                                   "instancename" : "Network NAME
inside VIM"
                            },
                     }]
              }]
       }
}
"grantApprovalDescriptor": {
       "vnf": {
              "vnfInstantiationPossible": "true",
              "id": "IMS_01",
              "flavorid" : "Gold",
              "vnfInstantiate": {
                     "vim": {
                            "id": "IMSVIM",
                            "tenant id": "TENANT 1",
                            "Credentials":{
                                   "username": "nfvoadmin",
                                   "password": "Passw0rd432"
                            }
                     }
              }
       }
}
}
```

7.6.1.9 Scale

7.6.1.10 Scale_out

- Path for VNF: v1/vnf/<vnfid>/scale-out

- Path for VDU: v1/vnf/<vnfid>/vdu/<vduid>/scale-out
- Path for VM: v1/vnf/<vnfid>/vdu/<vduid>/vm/<vmid>/scale-out

7.6.1.11 Scale_in

- Path for VNF: v1/vnf/<vnfid>/scale-in
- Path for VDU: v1/vnf/<vnfid>/vdu/<vduid>/scale-in
- Path for VM: v1/vnf/<vnfid>/vdu/<vduid>/vm/<vmid>/scale-in

7.6.1.12 Scale_up

- Path for VNF: v1/vnf/<vnfid>/scale-up
- Path for VDU: v1/vnf/<vnfid>/vdu/<vduid>/scale-up
- Path for VM: v1/vnf/<vnfid>/vdu/<vduid>/vm/<vmid>/scale-up

7.6.1.13 Scale_down

- Path for VNF: v1/vnf/<vnfid>/scale-down
- Path for VDU: v1/vnf/<vnfid>/vdu/<vduid>/scale-down
- Path for VM: v1/vnf/<vnfid>/vdu/<vduid>/vm/<vmid>/scale-down

7.6.1.14 Common parameters

- HTTP Operation: PUT
- Normal Response Codes:
- Error Response Codes:

- Request JSON
  - Vnfdescriptorid Mandatory
  - Vnf descriptor Version
  - Flavor Mandatory if Descriptor contains flavors
  - vnfDetailsDescriptor Optional (see details on chapter 3.2.1)
  - grantApprovalDescriptor Optional (see details on chapter 3.2.1)

```
{
"vnfdescriptorid" : "ID Descriptor",
"flavor" : "Gold",
"vnfdescriptorversion" : "1.0"
}
```

- Response JSON (ResponseDescripter)

```
{
"description": "......",
"jobId": "<job_id>",
"link":http: //<machine>: <port>/v1/jobs/<job id>.json"
“entityId” : “<vnfid>”
“entitylink” : “http: //<machine>: <port>/v1/vnf/<vnf_id>”
}
```

7.6.1.15 Change Flavor

- Path: v1/vnf/<vnfid>
- HTTP Operation: PUT
- Normal Response Codes:
- Error Response Codes:

- Request JSON
    - Vnfdescriptorid Mandatory
    - Vnf descriptor Version
    - Flavor Mandatory if Descriptor contains flavors
    - vnfDetailsDescriptor Optional (see details on chapter 3.2.1)
    - grantApprovalDescriptor Optional (see details on chapter 3.2.1)

```
{
"vnfdescriptorid" : "ID Descriptor",
"flavor" : "Gold",
"vnfdescriptorversion" : "1.0"
}
```

- Response JSON (ResponseDescripter)

```
{
"jboId": "1001001",
"jboStatus": "Started",
"jobDescription": "service creation job",
"statusDescription": "Request is posted into VNFM"
}
```

7.6.1.16 Register Orchestrator

- Path: v1/vnfm/register
- HTTP Operation: POST
- Normal Response Codes:
- Error Response Codes:

- Request JSON (RegisterRequestDescriptor)

```
{
"nfvodomain": "cust.domainname.net",
"nfvoip": "10.10.10.2",
"Protocol": "https",
"Port": "8080",
"Credentials":{
       "username": "nfvoadmin",
       "password": "Passw0rd432"
}
}
```

- Response JSON (RegisterResponseDescriptor)

```
{
"status": "Succes",
"statusDescription": "NFVO Details Successfully added"
}
```

Example of a RegisterResponseDescriptor for a failed case:

```
{
"status": "Failed",
"errorCode": "500",
"statusDescription": "NFVO Details not added due to communication
problem"
}
```

7.6.1.17 Unregister Orchestrator

- Path: v1/vnfm/register/<vnfid>
- HTTP Operation: DELETE
- Normal Response Codes:
- Error Response Codes:
- Request JSON () N/A
- Response JSON (RegisterResponseDescriptor)

```
{
"status": "Succes",
"statusDescription": "NFVO Details Successfully added"
}
```

Example of a RegisterResponseDescriptor for a failed case

```
{
"status": "Failed",
"errorCode": "500",
"statusDescription": "NFVO Details not added due to communication
problem"
}
```

7.6.1.18 Get list Orchestrator registered

- Path: v1/vnfm/register
- Parameters:
  - nfvodomain
  - nfvoip
- HTTP Operation: GET
- Normal Response Codes:
- Error Response Codes:
- Request JSON () N/A
- Response JSON (RegisterResponseDescriptor)

```
{
"nfvdos": [
{
       "Id": "vnfd Orchestrator ID 1",
       “url” : “”
},
{
       "Id": "vnfd Orchestrator ID 2",
       “url” : “”
}
]
}
```

Chapter 8

Secure Connections

8.1 Enabling SSL communication between OpenStack and SiteScope

SSL communication can be enabled in OpenStack components by performing the following changes:

- Obtain X.509 Certificate
- Configure the above certificate in OpenStack as follows:
    - Edit the SSL section in /etc/keystone.conf

```
[ssl]
enable = True
certfile = <path to keystone.pem>
keyfile = <path to keystonekey.pem>
ca_certs = <path to ca.pem>
cert_required = True
```

Here
certfile: Path to the Identity Service public certificate file.
keyfile: Path to the Identity Service private certificate file. If you include the private key in the certfile, you can omit the keyfile.
ca_certs: Path to the CA trust chain
cert_required: Requires client certificate

8.2 Enabling SSH between OpenStack and SiteScope

The SSH communication is required to be enable to execute NFVI script monitors.

1. Login to Sitescope machine as root user and execute the below command to generate the key.

```
# ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
87:bc:83:12:6e:e4:db:18:ed:85:43:8c:31:47:53:fc root@osskemp1
The key's randomart image is:
+--[ RSA 2048]----+
|     oo.        |
|   . ..       |
|  o . .      |
|   * . .E     |
|  + o S .     |
| + + o o      |
|  * = +       |
| . B o .      |
|  o o         |
+-----------------+
#
```

2. The generated public key is stored in /root/.ssh/id_rsa.pub file.Check the contents of id_rsa.pub file by executing the below command

```
# cat /root/.ssh/id_rsa.pub
```

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA1YWKnVXx3QPHTU8YGLpZFk8CblLJVHLLC+Uez6gZmoT+f1
Vlz8nuKGbLFdTpFFHSv5g3iiEODAzSPRohAfmtdYfcyHkyN5PKdU8aRhv4dgTkeKS19okGcuXC8RAlaGeC
9pOxTFtx8+mwbnzZEoVLWYiNJ+3lBeE5J3D9ePn7Y4Cxqs7f6oHOZiGn93wptnnkTjQBWjqEeWLbQTYdy
PA07MvIzy8Qur1BQrTb8MyyzyTOz3nEjrd9YFmj+khyqz1W6y0CIshCDJELQ8YmONcpoE9pHDba1sXVD
VqUCWlZ67fx5vYxKa1QehlEl6t/2GE55CJtk34mBxc/o/PlkucCSQ== root@host
#
```

3. Transfer the *id_rsa.pub* file to OpenStack Machine */root/.ssh/ folder*
4. Check is there a *"authorized_keys"* file exists in that folder.If the file is not existing create a file by using the below command.

```
touch authorized_keys
```

5. Append the contents of *id_rsa.pub to "*authorized_keys" file using the command

```
# cat id_rsa.pub >> authorized_keys
```

6. Exit and try login to OpenStack Machine from Sitescope Machine using SSH.It should be able to login without any password.
7. Repeat Steps 3 to 6 for all the OpenStack Nodes.

8.3 Enabling SSL in NFVD GUI

8.3.1 Configuring the UOC for SSL

For instructions on configuring the NFVD-UI server for secured communication between the client browser and the server, see section 11.7 of the HP UOC MR - Installation Guide V1.0

8.3.2 Configuring the NFVD GUI for SSL

This section will only document how to configure NFVD-UI to access the fulfillment or assurance servers over SSL.
To configure SSL access from the UOC server to the NFVD RestAPI server,

Edit the protocol, host and port as needed in this file:
<install_dir>/server/public/addons/plugins/nfvd/config.json

8.3.2.1 Signed certificate case:

Add the name of the certification authority certificate file, if needed. This file is obtained from your signing provider and must be copied in:
<install_dir>/server/public/ssl

```
...
"server_fulfill": {
"protocol": "https",
        "host": " nfvd_server_system_full_hostname_or_ip_address",
        "port": "8443",
"ssl": {
"caCertFile": "nfvd_ca.crt"
}
    },
    "server_assurance": {
"protocol": "https",
        "host": "
nfvd_assurance_server_system_full_hostname_or_ip_address",
        "port": "8443",
"ssl": {
"caCertFile": "nfvd_ca.crt"
}
    },
...
```

8.3.2.2 Self signed certificate case:

Add the strictSSL parameter to disable strict certificate checking.

```
...
"server_fulfill": {
"protocol": "https",
        "host": " nfvd server system full hostname or ip address",
        "port": "8443",
"ssl": {
"strictSSL": false
}
    },
    "server assurance": {
"protocol": "https",
        "host": "
nfvd assurance server system full hostname or ip address",
        "port": "8443",
"ssl": {
"strictSSL": false
}
    },
...
```

8.4 Enabling SSL in Assurance Gateway

To enable SSL in the assurance gateway, we need to create a private key and a public certificate, which can be bundled inside a “Keystore” file ,which will be used to decrypt the data over HTTPS secured port e.g. 18443.

8.4.1 Create a Key store file

1. use java keytool for the same
2. Go to $JAVA_HOME directory and run the keytool, e.g.

   keytool -genkey -keyalg RSA -alias assuranceselfsigned –keystore assuranceKeyStore.jks -storepass admin123 -validity 360 -keysize 2048

3. Fill the details of certificates

   For the details of each field refer here.
4. assuranceKeyStore.jks will be generated in the current directory, so you can move to any other folder conveniently.

8.4.2 Configure the JBOSS

1. If JBOSS is running in standalone mode.
2. Configure the web subsystem in /opt/HP/nfvd/tpp/jboss/standalone/configuration standalone.xml file to set up HTTPS connector and mention the location of "Keystore" file and its details

   e.g.

```
    <subsystem xmlns="urn:jboss:domain:web:1.1" default-virtual-server="default-host" native="false">
        <connector name="http" protocol="HTTP/1.1" scheme="http" socket-binding="http"  redirect-
port="18443" />
       <connector name="https" protocol="HTTP/1.1" scheme="https" socket-binding="https" enable-
lookups="false" secure="true">
                  <ssl name="ssl" key-alias="assuranceselfsigned" password="admin123" certificate-key-
file="/var/opt/HP/nfvd/conf/assuranceKeyStore.jks" protocol="TLSv1" verify-client="false"/>
        </connector>
        <virtual-server name="default-host" enable-welcome-root="true">
          <alias name="localhost"/>
          <alias name="example.com"/>
        </virtual-server>
    </subsystem>
```

3. Enable the HTTPS port in the same standalone.xml

   e.g. <socket-binding name="https" port="${jboss.https.port:18443}"/>

4. Configure nfvd_agw_env.sh

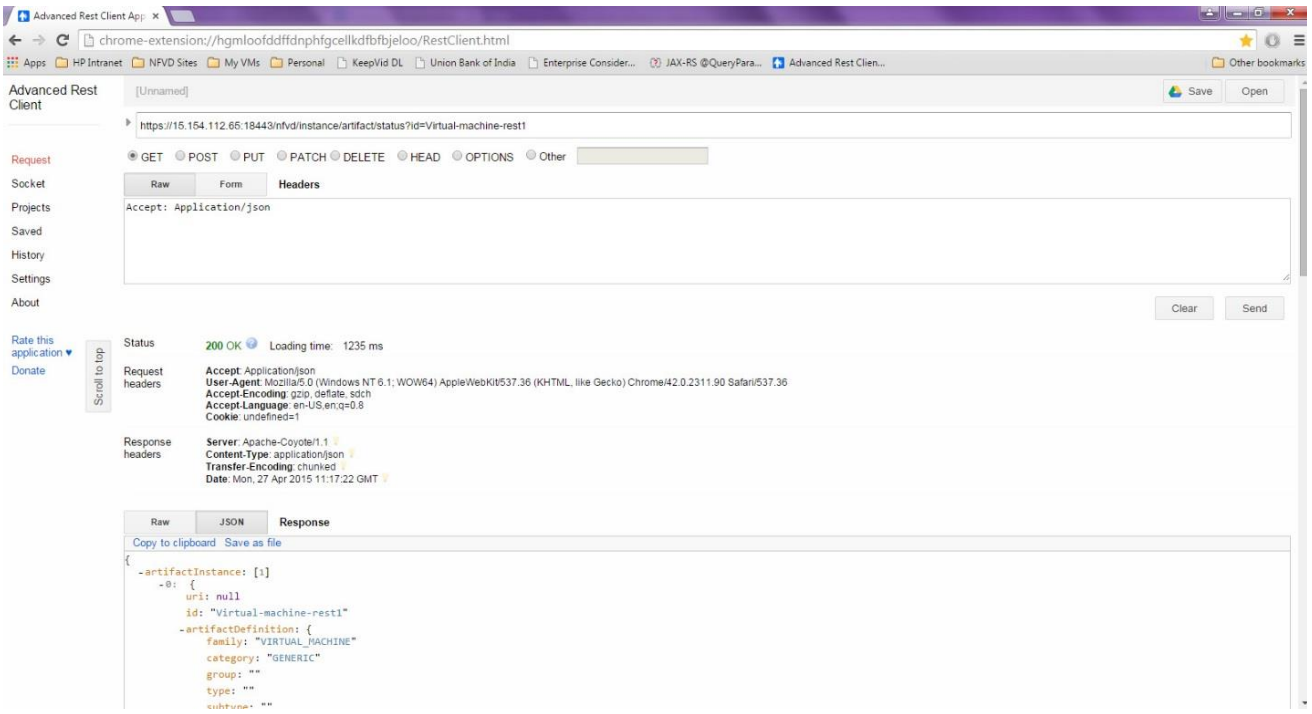
   Make sure to change the https port in assurance env file , which overrides all the ports configuration mentioned in standalone.xml

   File path = /opt/HP/nfvd/bin/nfvd_agw_env.sh

```
[root@nfvdvm25 bin]# cat nfvd_agw_env.sh
#!/bin/bash

#Place holders for directories, to be replaced during postinstall
NFVD_DEPLOY_DIR=/opt/HP/nfvd/tpp/jboss/standalone/deployments/
NFVD_DATA_DIR=
NFVD_CONF_DIR=/var/opt/HP/nfvd/conf
NFVD_BIN_DIR=
NFVD_HOME_DIR=/opt/HP/nfvd
NFVD_JBOSS_HOME=/opt/HP/nfvd/tpp/jboss
NFVD_JBOSS_BIND_ADDRESS=0.0.0.0
NFVD_JBOSS_MANAGEMENT_NATIVE_PORT=19999
NFVD_JBOSS_MANAGEMENT_HTTP_PORT=19990
NFVD_JBOSS_MANAGEMENT_HTTPS_PORT=19443
NFVD_JBOSS_HTTP_PORT=18080
NFVD_JBOSS_HTTPS_PORT=18443
NFVD_JBOSS_AJP_PORT=18009
NFVD_JBOSS_OSGI_HTTP_MANAGEMENT_PORT=18090
NFVD_JBOSS_REMOTING_PORT=14447
NFVD_JBOSS_TXN_RECOVERY_ENV_PORT=14712
NFVD_JBOSS_TXN_STATUS_MANAGER_PORT=14713
```

8.4.3 Sample HTTPS REST call


Advanced Rest Client App
chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo/RestClient.html
Apps · HP Intranet · NFVD Sites · My VMs · Personal · KeepVid DL · Union Bank of India · Enterprise Consider... · JAX-RS @QueryPara... · Advanced Rest Clien... · Other bookmarks
Advanced Rest Client
[Unnamed]
Save
Open
Request
Socket
Projects
Saved
History
Settings
About
Rate this application ♥
Donate
Scroll to top
Clear
Send
Raw
Form
Headers
Raw
JSON
Response
Copy to clipboard
Save as file

https://15.154.112.65:18443/nfvd/instance/artifact/status?id=Virtual-machine-rest1

[✓ GET] POST PUT PATCH DELETE HEAD OPTIONS Other

```
Accept: Application/json
```

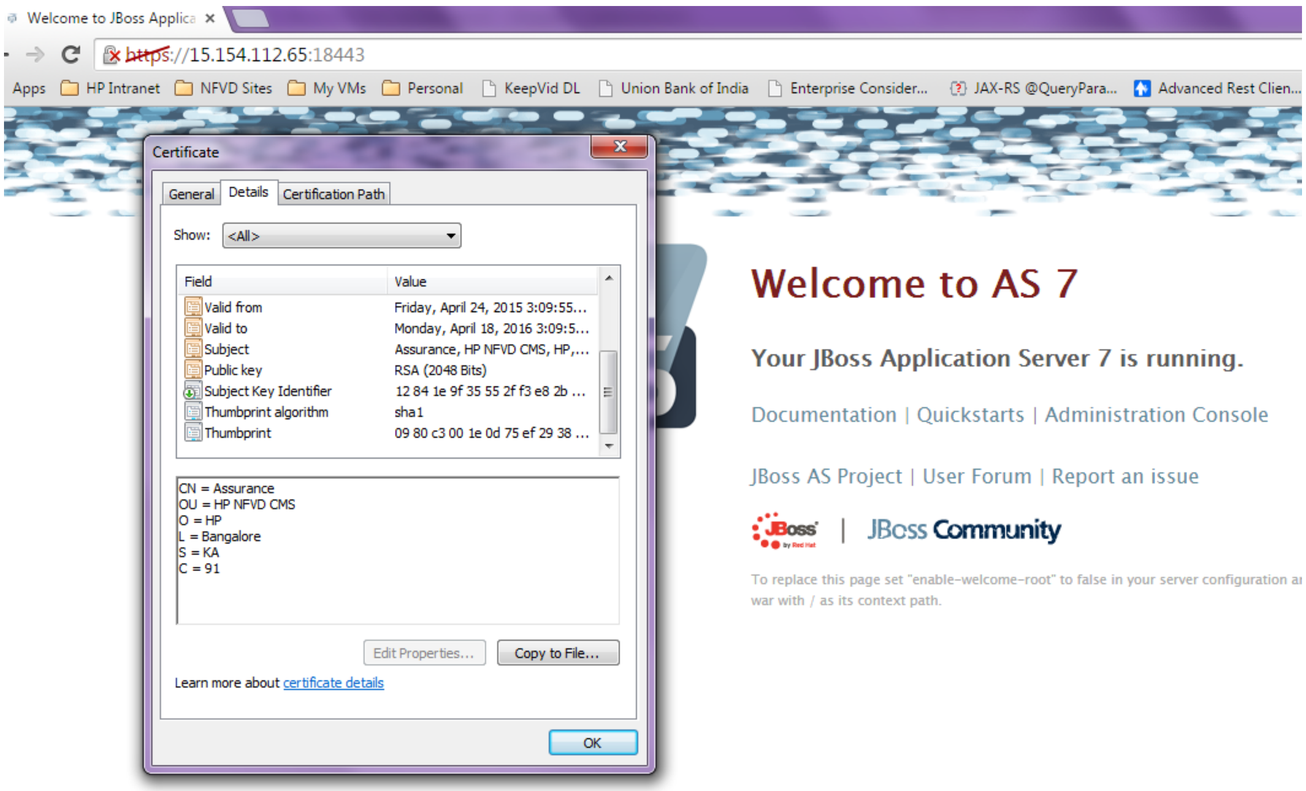
Status 200 OK Loading time: 1235 ms

Request headers
Accept: Application/json
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.90 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cookie: undefined=1

Response headers
Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Mon, 27 Apr 2015 11:17:22 GMT

```
{
  -artifactInstance: [1]
     -0:  {
          uri: null
          id: "Virtual-machine-rest1"
         -artifactDefinition: {
             family: "VIRTUAL_MACHINE"
             category: "GENERIC"
             group: ""
             type: ""
             subtype: ""
```

8.4.4 Certificate information:


Welcome to JBoss Applica
~~https~~://15.154.112.65:18443
Apps HP Intranet NFVD Sites My VMs Personal KeepVid DL Union Bank of India Enterprise Consider... JAX-RS @QueryPara... Advanced Rest Clien...
Certificate
General
Details
Certification Path
Show:
<All>
Edit Properties...
Copy to File...
Learn more about certificate details
OK

Field	Value
Valid from	Friday, April 24, 2015 3:09:55...
Valid to	Monday, April 18, 2016 3:09:5...
Subject	Assurance, HP NFVD CMS, HP,...
Public key	RSA (2048 Bits)
Subject Key Identifier	12 84 1e 9f 35 55 2f f3 e8 2b ...
Thumbprint algorithm	sha1
Thumbprint	09 80 c3 00 1e 0d 75 ef 29 38 ...

```
CN = Assurance
OU = HP NFVD CMS
O = HP
L = Bangalore
S = KA
C = 91
```

Welcome to AS 7

Your JBoss Application Server 7 is running.

Documentation | Quickstarts | Administration Console

JBoss AS Project | User Forum | Report an issue

JBoss | JBoss Community

To replace this page set "enable-welcome-root" to false in your server configuration a
war with / as its context path.

Glossary

NFV: Network Function Virtualization

VNF: Virtual Network Function

VNFD: Virtual Network Function Descriptor

NFVD: Network Function Virtualization Director

VNFM: Virtual Network Function Manager

VM: Virtual Machine

VNFC: Virtual Network Function Component

EMS: Element Manager System

VIMS: Virtual Infrastructure Manager

NS: Network Service

NBI: North Bound Interface

UCA: Unified Correlation Analyzer

EBC: Event Based Correlation

IP: Installation Package for HP NFV Director V1.0

JDK:	Java Development Kit
JMS:	Java Messaging Service
JMX:	Java Management extension, used to access or process action on the NFV Director product
JNDI:	Java Naming and Directory Interface
JRE:	Java Runtime Environment
DRL:	Drools Rule file
XML:	Extensible Markup Language
XSD:	Schema of an XML file, describing its structure
X733:	Standard describing the structure of an Alarm used in telecommunication environment
EVP:	NFV Director Value Pack