HP NFV Director

HP NFV Director

Version 1.0

Integration Guide

Edition: 1.1

For the Linux (RHEL6.4) Operating System

September 2014

Legal Notices

Warranty

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

License Requirement and U.S. Government Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

Adobe®, Acrobat® and PostScript® are trademarks of Adobe Systems Incorporated.

Red Hat® and the Red Hat "Shadow Man" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Java™ is trademark of Oracle and/or its affiliates.

Microsoft®, Windows® and Windows NT® are U.S. registered trademarks of Microsoft Corporation.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

X/Open® is a registered trademark, and the X device is a trademark of X/Open Company Ltd. in the UK and other countries.

Contents

Figures

Tables

Preface

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 onboarding ,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.

Please read this document before installing or using this software.

Intended 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 1.0	Red Hat Enterprise Linux Server release 6.4

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.

Reference Documents

- HP NFV Director Installation Guide
- HP NFV Director User Guide
- HP NFV Director Release Notes
- 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

Support

Please visit our HP Software Support Online Web site at www.hp.com/go/hpsoftwaresupport for contact information, and details about HP Software products, services, and support.

The Software support area of the Software Web site includes the following:

- Downloadable documentation
- Troubleshooting information
- Patches and updates
- Problem reporting
- Training information
- Support program information

Chapter 1

HP NFV Director

1.1 HP NFV Director (NFVD) functionality

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

- Orchestrates the allocation and release of resources to be used by VNF.
- Applies policies for orchestration and resource allocation.
- Installs and instantiates VNF software images on virtual, physical machines, or both.
- Configures virtual or physical networks to connect instantiated machines.
- Assumes the role of a VNF Manager in the absence of VNF Manager for a VNF.
- Monitors and accounts NFVI resources for a given VNF, network service, or both.
- Checks the usage of a resource against policies.
- Collects performance data for NFVI compute, storage, and network resources.
- 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, NS, or both.
- Provides a framework for fault correlation and a root-cause analysis process to determine the reason for fault conditions and their impact on VNFs, network services, or both.
- Provides a framework to determine corrective actions and triggers such actions at one or more action points within the NFV framework or to OSS.
- Defines and provides an interface to external entities (for example, OSS and NMS) for:
    - VNF on-boarding
    - Lifecycle management of VNF instances (in co-ordination with VNF Managers)
    - NFV Policy Management
    - Performance Data of VNF, NS, or both.
    - NFVI resource usage accounting

Note: See the `HP NFV Director Release Notes` for the features supported in the 1.0 release.

1.2 HP OpenNFV program

NFVD is a part of HP OpenNFV program, operating NFV through an NFV orchestrator with embedded VNF manager capabilities.

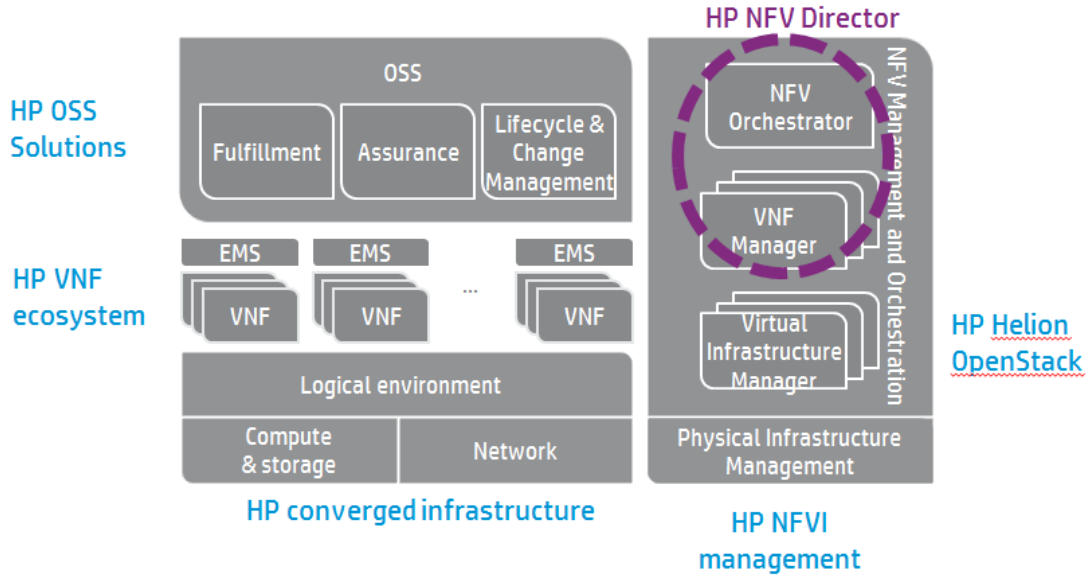


Figure 1 HP OpenNFV Program

1.3 Architecture

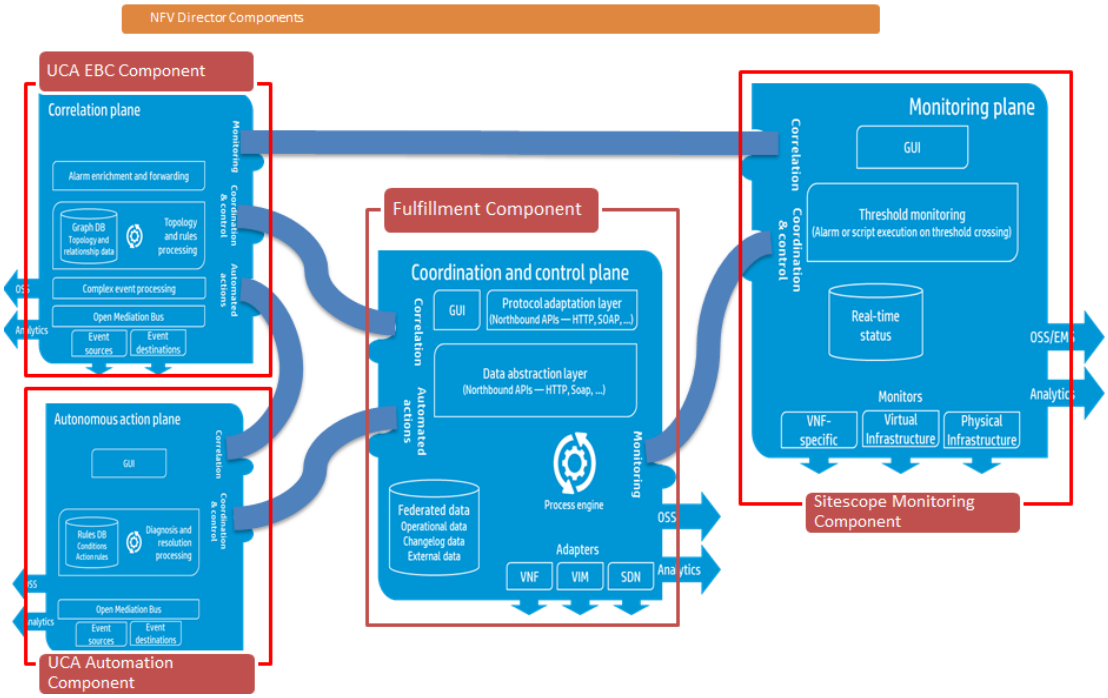


Figure 2 NFV Director Components

1.3.1 Fulfillment component

At a high level, the NFVD fulfillment is based on four components as described in the following table.

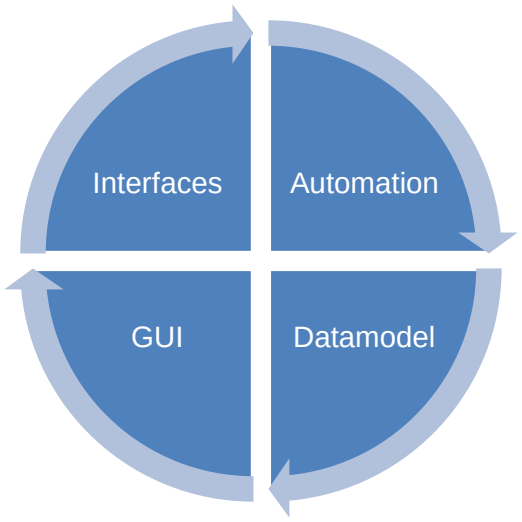


Figure 3 Fulfilment Components

Interfaces	GUI	Data model	Automation
• Northbound ○ Generic (Web service) ○ OpenStack (Rest) ○ Custom (project based through Protocol adapters)	• HPSA standard JSPs	• HPSA Standard Beans	• HPSA standard workflows
• Southbound • HPSA Plug-ins (using Templates)	• HPSA XPMAPS • Dynamic forms	• Generic Artifact/Relationships Beans	• GPM Process

Table 2 Fulfillment Components

These are solutions based on the HPSA platform. To customize any of the above tasks, a good

understanding of the HPSA platform, the HPSA workflow, and the SOSA framework is a prerequisite.

1.3.2 Monitoring component

1.3.2.1 Assurance gateway

- Receives and processes the VNF topology information from fulfillment, and updates UCA-EBC graph database for correlation.
- Receives and processes the monitoring notification from fulfillment, and delegates appropriate action with all the data to SiteScope agent-less monitoring component.

1.3.2.2 Resource Monitoring

NFVD includes the agent-less monitoring component. Built using HP SiteScope, this component 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 NFVD as the VNF is provisioned or modified. As an agent-less solution, NFVD does not require the 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 KPIs provided by NFVD. When the VNF management functionality is provided by NFVD (through the embedded VNF manager), it may be necessary to collect application-specific monitoring information. SiteScope provides the capability to develop custom monitors for VNF resources and VNF applications.

SiteScope templates are used to standardize a set of monitor types and configurations in a single structure. Then, this structure can 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 the single-operation deployment of groups, monitors, alerts, remote servers, and configuration settings.

1.3.2.3 Mediation

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 this context, the NFVD Open Mediation is used to integrate with SiteScope, EMS, VNF Manager, or any other source of events.

1.3.3 Correlation component

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 rules execution (the rules are written in a Java-based language). Any Value Pack can support and use predefined functionalities, for example, alarm collection, filtering, life cycle, and generic actions.
In terms of functionalities, UCA for EBC can:

- Collect alarms (the Alarm model is based on a mix of X733 and OSS/J Fault Management Model) and map them into the Operator Alarm model (Alarm).
- Run several scenarios (rule engines) in parallel, in sequence, or both 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 to the different scenarios.
- Execute rules based on 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, for example, storing to a database, creating a trouble ticket, creating a new alarm, group alarms, forwarding an alarm to another scenario, or executing a generic action through the OSS Open Mediation V6.2 (NOM V6.2) layer.

Rules files are nothing but JBoss rules files. Therefore, both JBoss Expert and Fusion rules are supported in UCA for EBC rule files. JBoss Drools Expert and JBoss Drools Fusion are JBoss Drools basic modules.

On top of 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 coming from any number of sources (NMS) and performing actions in return.

1.3.4 Autonomous action component

The UCA automation software, which is a combination of both business rule engine, and the workflow engine enables a clear separation of what to automate and how to automate. All the complexities of the actual automation, for example, how to access a network resource (can be a network element, an element component, or an EMS or NMS), what its credentials may be, which specific transport mechanism to use to connect to the resource, which specific OS version of the device must be supported, what specific commands must be sent, are abstracted from the business rules. This enables the administrators to create, update, and read the business rules with utmost clarity and maintain them efficiently. This empowers the administrators to store the knowledge gained regarding the automation in the form of business rules focusing on the actual part without bothering about the procedure. Another advantage of the UCA automation software is, for most of the resolution automations, it requires only the operator to know the business rules. The operator does not need the knowledge of the business rules technologies to implement day-to-day operational changes for the decisions.

Therefore, the UCA automation system is a platform for building value added resolution automations based on a judicious combination of business rules and workflows. The following diagram shows the overall architecture of the UCA automation system.

1.4 Interaction between HP NFV Director components

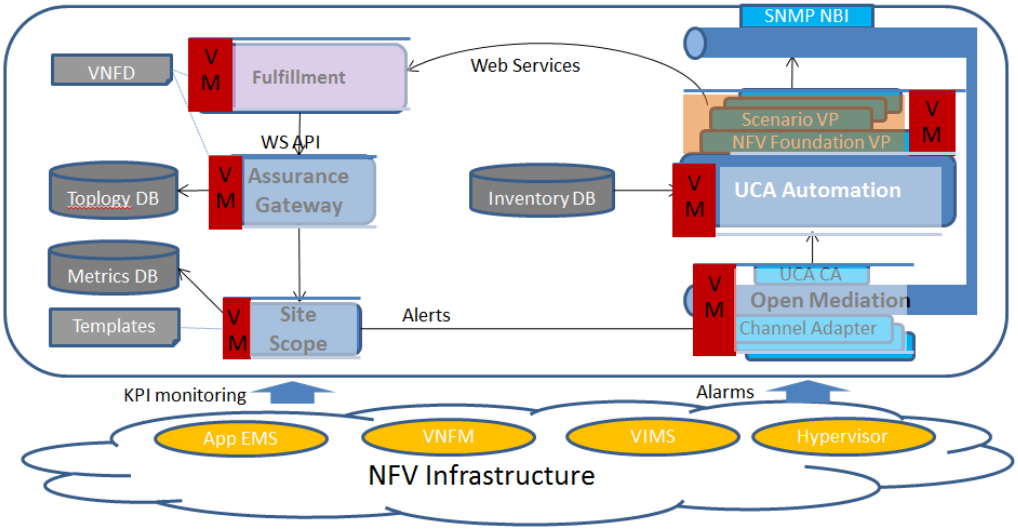


Figure 4 Interaction between HP NFV Director Components

- On instantiation of VNF, the fulfillment component processes the VNF descriptor, applies policies and creates the required VNF and its subcompoents (for example, virtual machines) using Virtual Infrastructure Manager (VIM).
- Fulfillment also sends topogology information of the created VNF to the monitoring component.
- The monitoring component processes this information and stores the topology information in the togpology database for later correlation of events.
- At the behest of the fufillment component the monitoring component deploys and activates the monitors associated to a VNF, its subcomponents, or both (for example, VM).
- The monitoring component continously 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.

1.5 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 5 Customization and Integration points for NFV Director

1.5.1 Integration points

1.5.1.1 Integration using REST and WS API

- NFV Director provides WebService interface for VNF management.

  The list of supported WS API is described in the section North Bound API Provided by HP NFV Director.

- The NFV Director also provides basic OpenStack Rest based API.

  The list of OpenStack support API is described in Appendix B.

- The NFV Director is extensible and any new interface can be developed on demand to integrate with external components and other products.

1.5.1.2 Integration of events for correlation

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.

1.5.2 Customization

1.5.2.1 VNFD

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

1.5.2.2 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.

1.5.2.3 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.

1.5.2.4 OpenMediation channel adaptors

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

1.5.2.5 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.

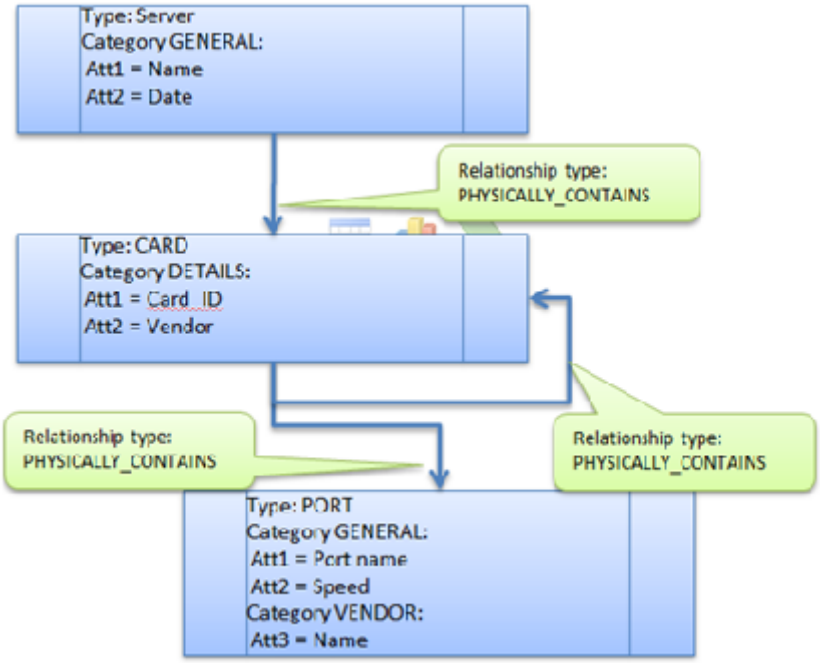


Figure 6 Generic Model Example

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 nor 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 to recreate 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.

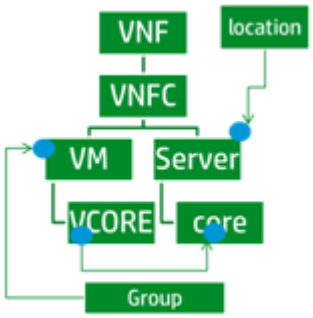


Figure 7 Artifact and Relationship Definition

- 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.

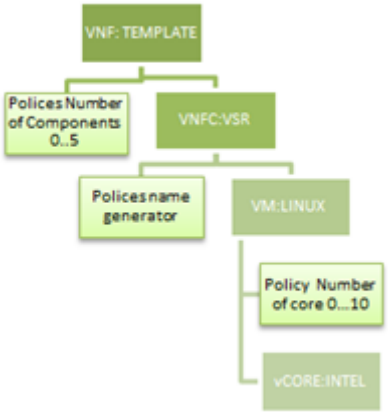


Figure 8 Artifact and Relationship Template

- Instance (data)

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

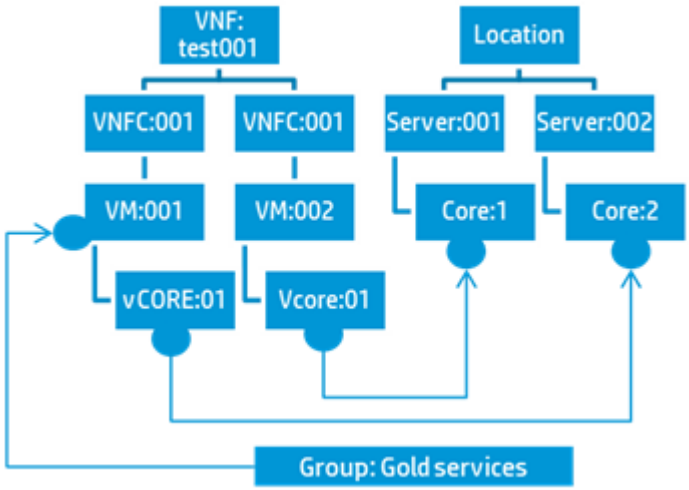


Figure 9 Artifact and Relationship template

2.2 North Bound API Provided by HP NFV Director

- VM management
  1. NFVD | CREATE | VM (for creating a VM)
  2. NFVD | DELETE| VM (for deleting a VM)
  3. NFVD | START| VM (for starting a VM)
  4. NFVD | STOP| VM (for stopping a VM)
  5. NFVD | REBOOT| VM (for rebooting a VM)
  6. NFVD | SCALE_UP| VM (for scale up a VM)
  7. NFVD | SCALE_DOWN| VM (for scale down a VM)
- VNF management
  1. NFVD | CREATE | VNF (for creating a VNF)
  2. NFVD | DELETE| VNF(for deleting a VNF)
  3. NFVD | SCALE_IN| VM (for scale in a VNF)
  4. NFVD | SCALE_OUT| VM (for scale out a VNF)
  5. NFVD | SCALE_UP| VNF (for scale up a VNF)
  6. NFVD | SCALE_DOWN| VNF (for scale down a VNF)
  7. NFVD | START| VNF(for starting a VNF)
  8. NFVD | STOP| VNF(for stopping a VNF)
  9. NFVD | REBOOT| VNF(for stopping a VNF)
- Monitor
  1. NFVD | CREATE | MONITOR (for creating a MONITOR)
  2. NFVD | DEPLOY| MONITOR (for deploying a MONITOR)
  3. NFVD | START| MONITOR (for starting a MONITOR)
  4. NFVD | STOP| MONITOR (for stopping a MONITOR)

Please refer to Appendix A for the syntax and semantics to invoke the above Web Service APIs.

Chapter 3

VNF on-boarding

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.

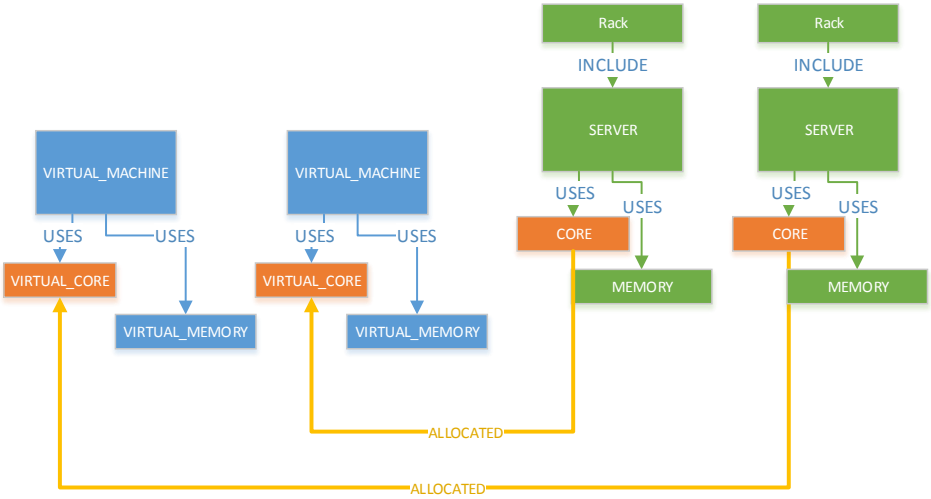


Figure 10 Virtual machine Vcores assigned (allocated) to Cores of Servers

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.

3.1.1 Virtual network function model

The VNF model is composed by a set of artifacts and relationships described as follows:

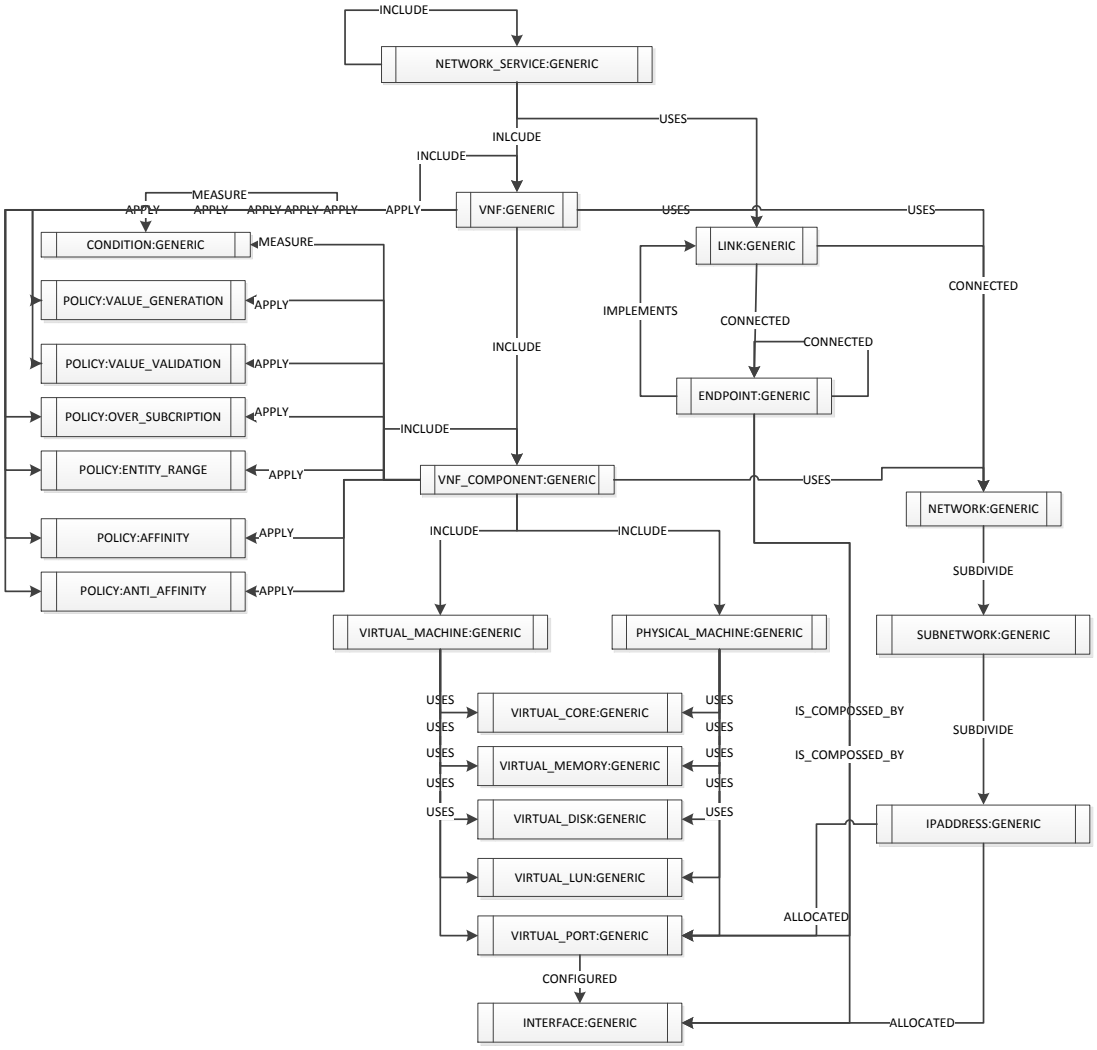


Figure 11 VNF Model

Template example with range policy

Templates

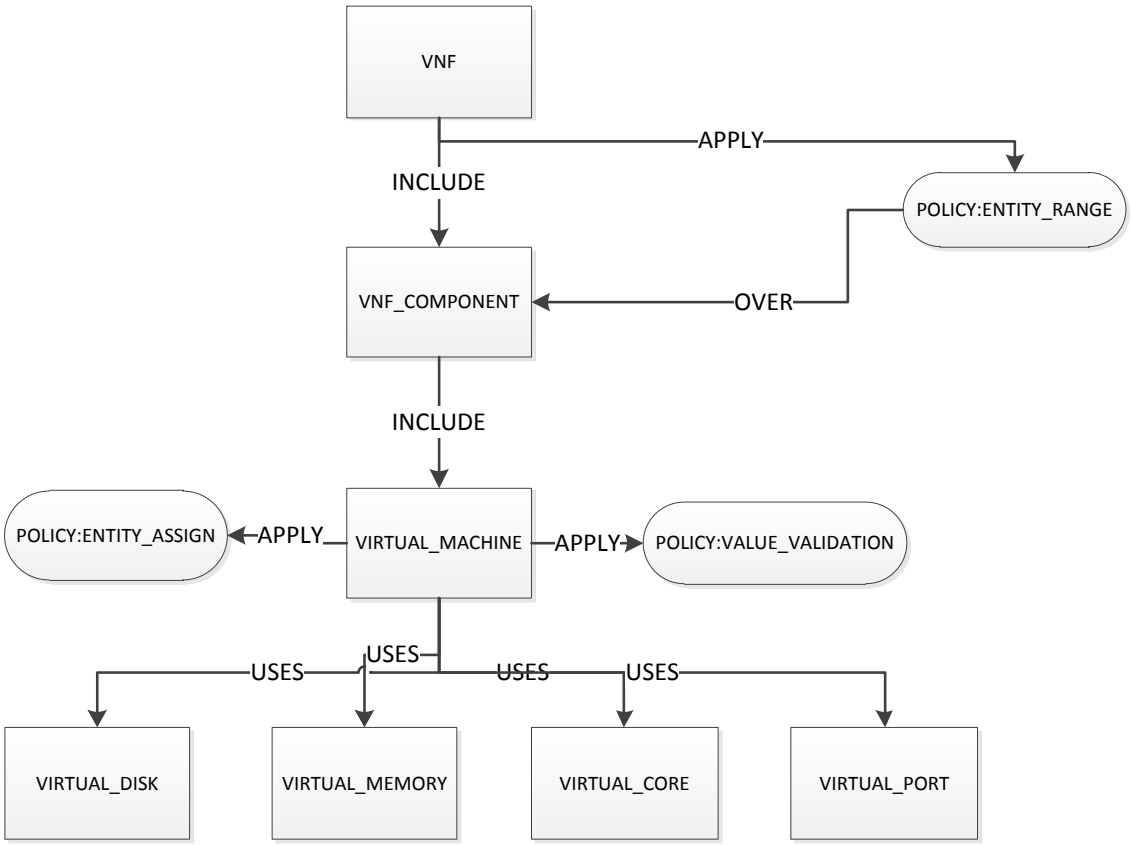


Figure 12 Template example with range policy

POLICY:ENTITY_RANGE parameters

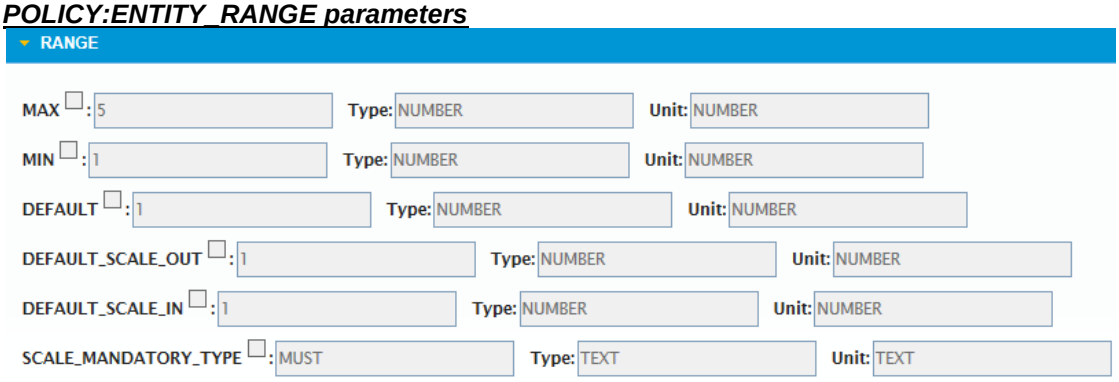


Figure 13 EntityRange Policy parameters

Instances

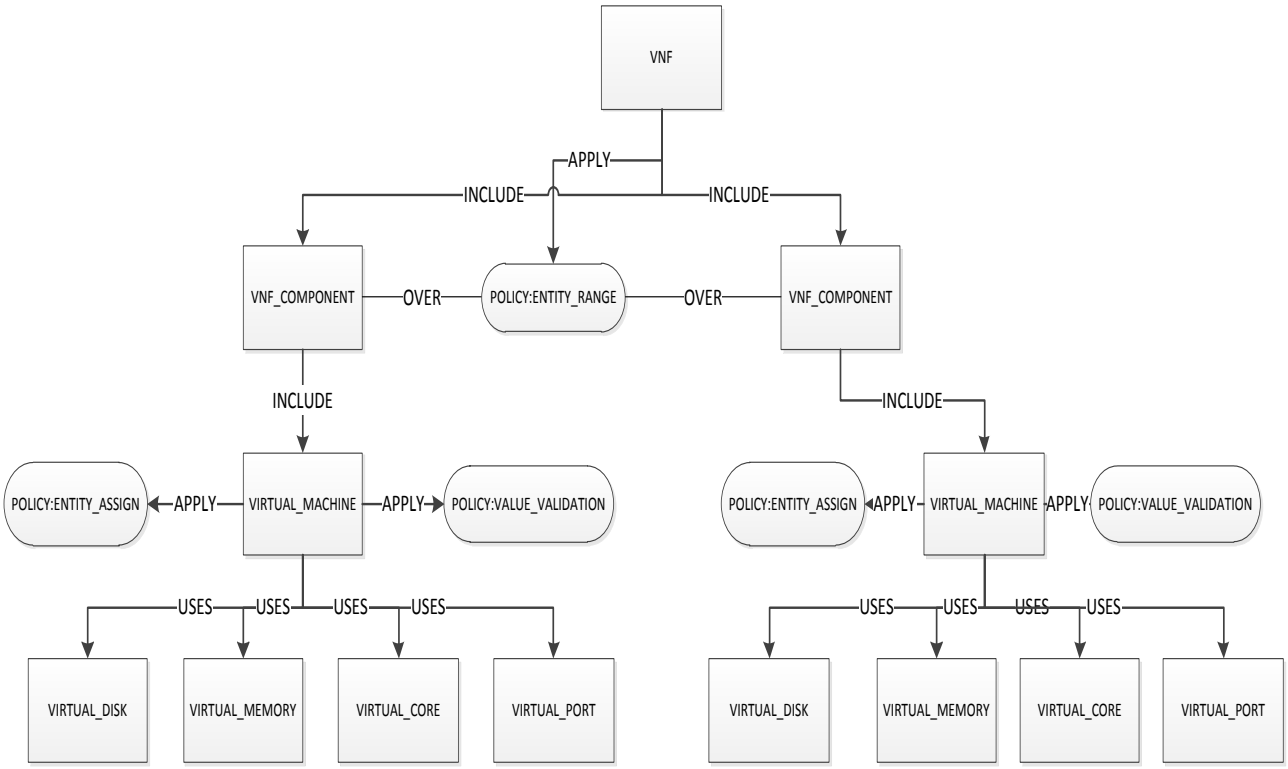


Figure 14 Instance result of template example with range policy

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.

Template example with assign policy

Templates

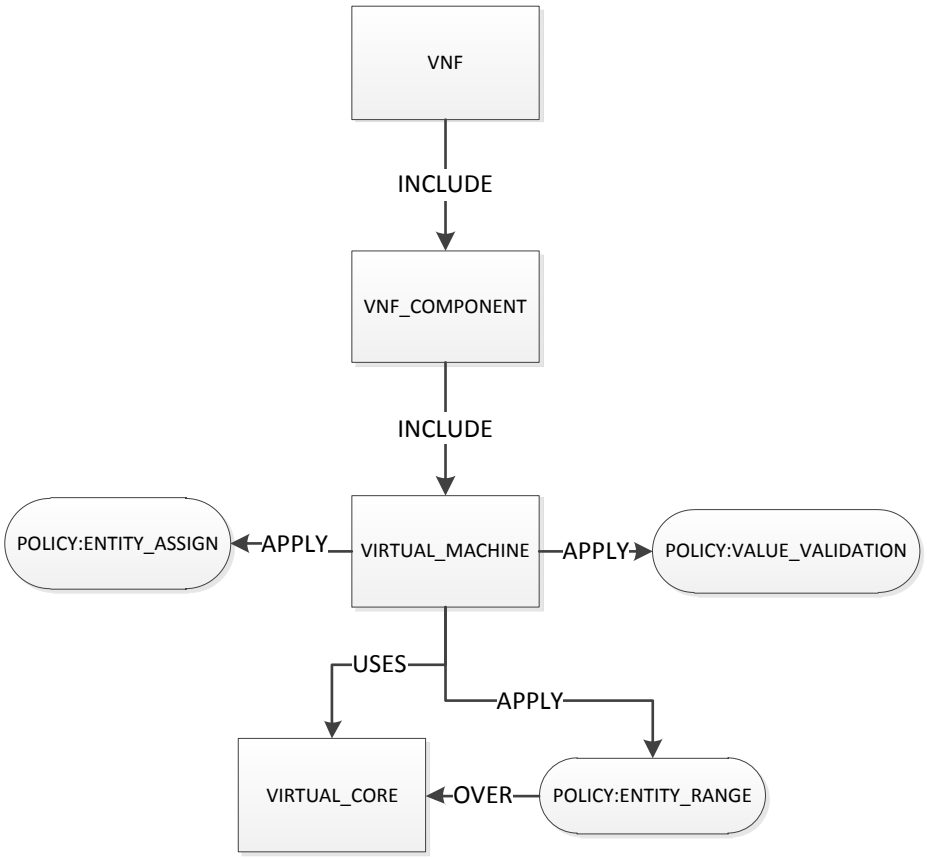


Figure 15 Template example with assign policy

POLICY:ENTITY_ASSIGN parameters

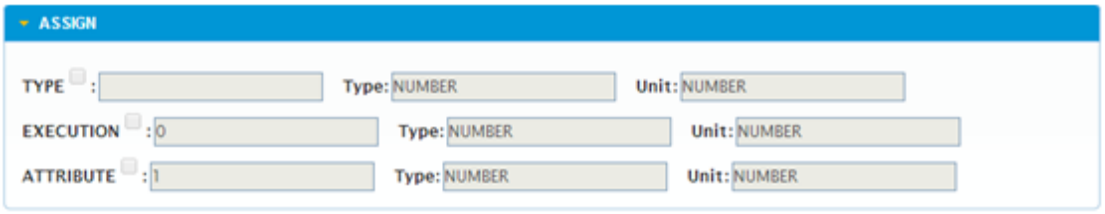


Figure 16 Assign Policy parameters

Instances

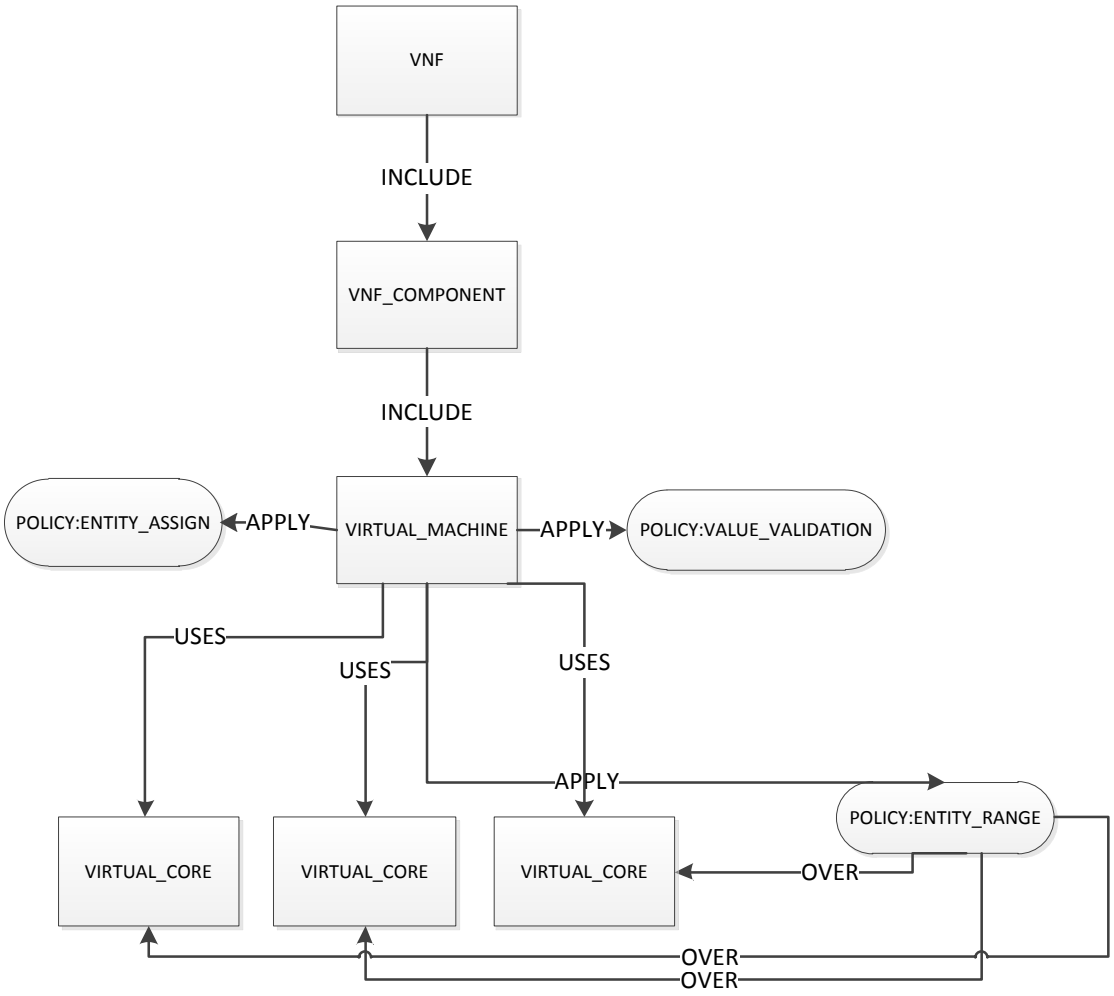


Figure 17 Instance result of Template example with assign policy

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 by a set of artifacts and its relationships are described as follows.

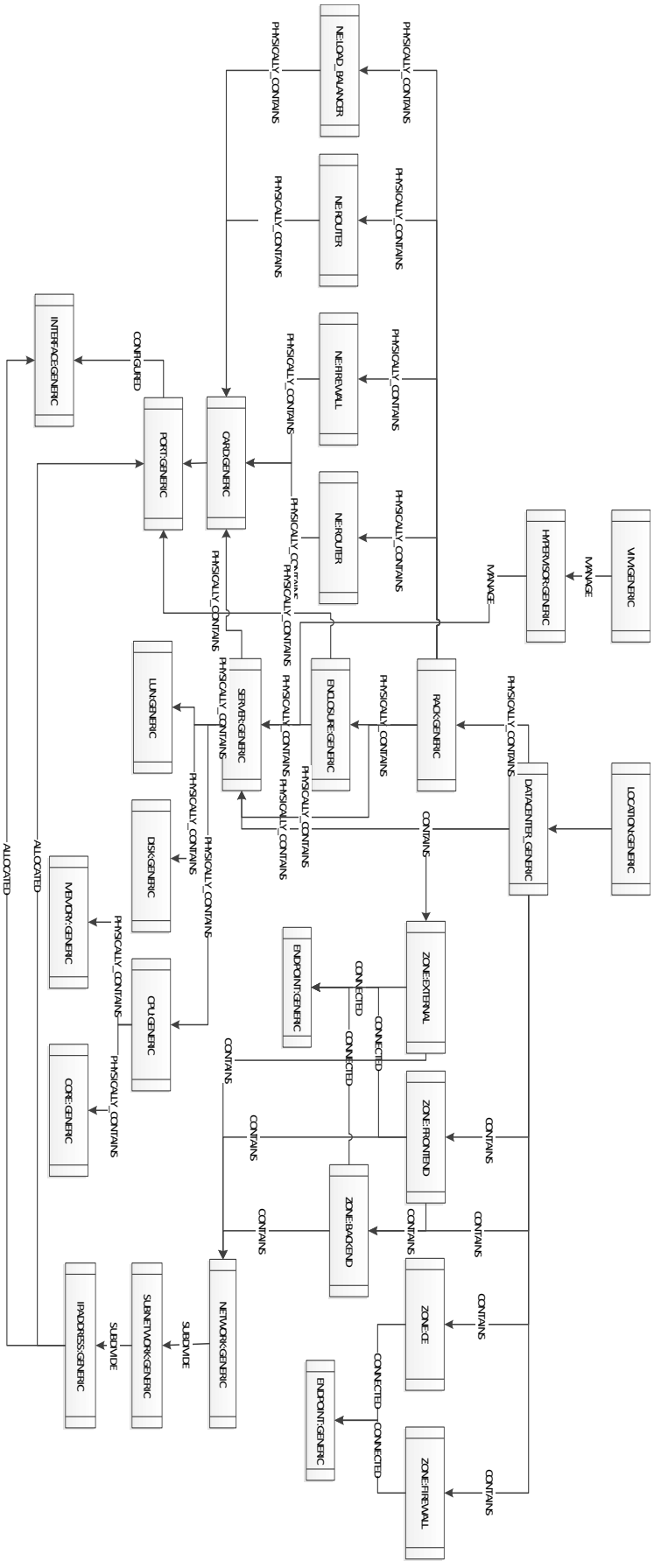


Figure 18 Resources Model

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 by a set of artifacts and its relationships are described as follows.

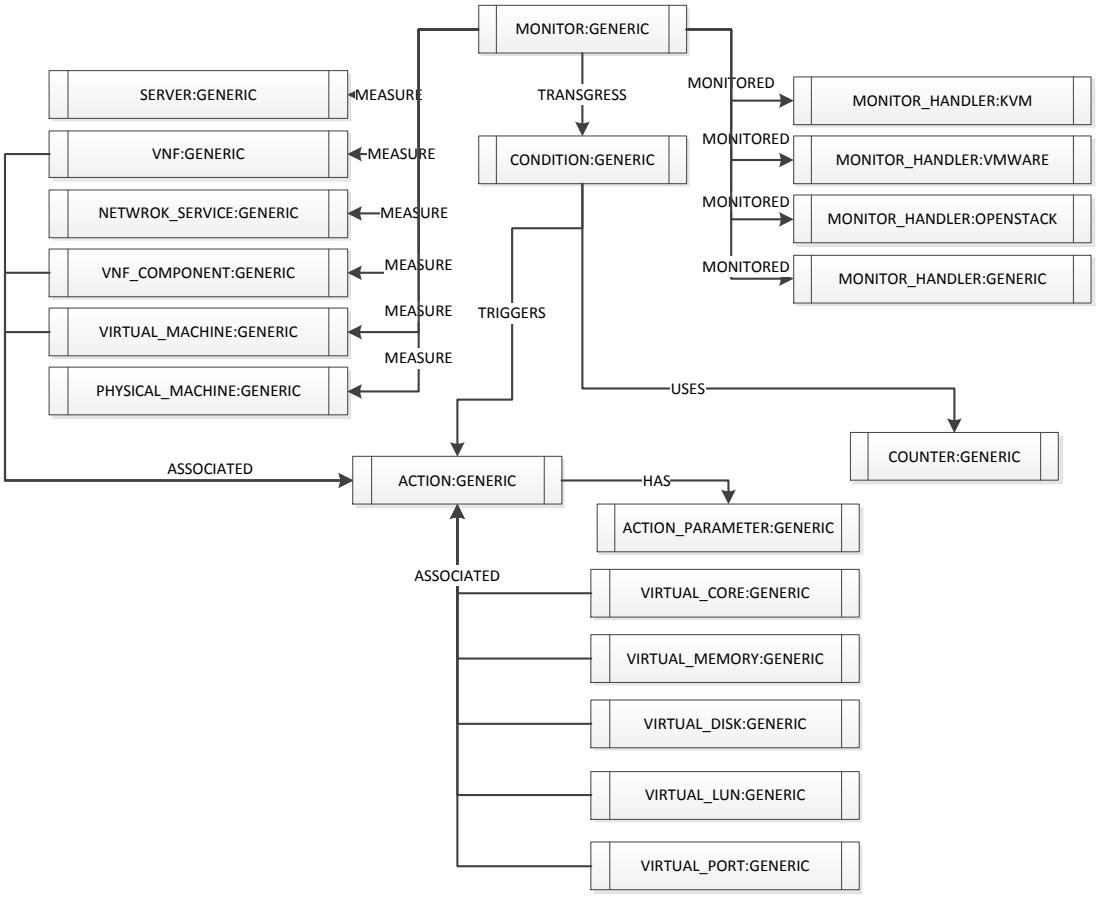


Figure 19 Monitoring Model

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).

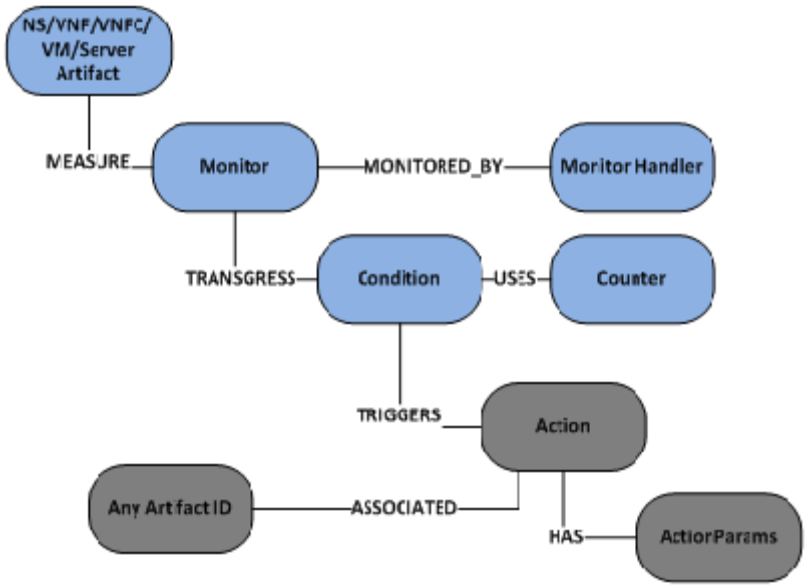


Figure 20 Monitoring Model

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.

Attribute	Description	Constraints
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. Please 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 Monitor handler artifact

Out of the box different flavors of Monitor handler artifact are available and are described as follows.

3.2.4.1 VMware monitor handler

Attribute	Description	Constraints
Name	Human readable name	Mandatory : Yes
Host	VMware ESXi host-name	Mandatory: only when moni-toring KPI is associated to a EXSI server.
Type	Future Use	
isVcenter	A flag to tell if vCenter is present or not.	Mandatory : Yes
vCenterIP	IP Address or Host-name of the vCenter	Mandatory : Yes
Login	Login name of vCenter	Mandatory : Yes
Password	Password of vCenter	Mandatory : Yes

Table 6 VMware Monitor handler attributes

3.2.4.2 KVM monitor handler

Attribute	Description	Constraints
Name	A human readable name	Mandatory: Yes
Type	Future Use	
Host	KVM hostname	Mandatory : Yes
Login	KVM host login name	Mandatory : Yes
Password	KVM host password	Mandatory: Yes

Table 7 KVM Monitor handler attributes

3.2.4.3 OpenStack monitor handler

Attribute	Description	Mandatory
Name	A human readable name	Yes
Type	Future Use	
URI	Keystone URI	Yes
Tenant Name	Name of the tenant owning the moni-tored resource	Yes

Login	Openstack Login name	Yes
Password	Openstack password	Yes

Table 8 OpenStack monitor handler attributes

3.2.4.4 GENERIC monitor handler

Attribute	Description	Mandatory
Name	A human readable name	Yes
Type	Future Use	
URI	URI to connect to fetch KPI of a Monitored resource	Depending upon the SiteScope custom monitor, this attribute is either mandatory or optional
Login	Login name	Depending upon the SiteScope custom monitor, this attribute is either mandatory or optional.
Password	Password	Depending upon the SiteScope custom monitor, this attribute is either mandatory or optional.

Table 9 Generic Monitor handler attributes

A Generic Monitor Handler can take any number of arguments.

Attribute	Description	Mandatory
Name	A Name	Depending upon the SiteScope custom monitor, this attribute is either mandatory or optional.
Value	Value associated to the name	Depending upon the SiteScope custom monitor, this attribute is either mandatory or optional.

Table 10 Generic Monitor handler attributes

3.3 North bound API for monitoring

The following operations are associated with a Monitor:

- Deploy a monitor

- Start a Monitor
- Stop a Monitor
- UnDeploy a Monitor

The Syntax and Semantics for the operations is provided in Appendix A.

3.4 Out of box monitors provided by NFV Director

	VMware ESXi		VMware VCenter		KVM		Openstack
	Host	*VM*	*Host*	*VM*	*Host*	*VM*	*VM*
CPU	✓	✓	✓	✓	✓	✓	✓
DiskRead	✓	✓	✓	✓	--	✓	✓
DiskWrite	✓	✓	✓	✓	--	✓	✓
Memory	✓	✓	✓	✓	✓	✓	--
NetworkRx	✓	✓	✓	✓	--	✓	✓
NetworkTx	✓	✓	✓	✓	--	✓	✓

Figure 21 Buitlin monitors

The first column represents the name of SiteScope monitoring template.

The remaining columns describe if this monitor template is capable to monitor the required KPI for VM that are created on different hypervisors or cloud management systems like the open stack.

The following KPIs and counters supported matrix for various hypervisors with units is by default provided by the NFV Director.

Monitor	*Counters*	*Hypervisor*					*Unit*	*Details*
		VMWare		KVM		Openstack		
		Host/ VCenter	VM	Host	VM	VM		
CPU	cpu_usage_average	✓	✓	✓	✓	✓	%	Percentage of CPU used
Memory	memory_usage_average	✓	✓	✓	✓	✗	%	Percentage of Memory used
DiskRead	disk_read_rate_average	✓	✓	✗	✗	✗	kB/s	Average number of kilobytes read from disk
	disk_read_requests	✗	✗	✗	✓	✓	request	Disk read requests
DiskWrite	disk_write_rate_average	✓	✓	✗	✗	✗	kB/s	Average number of kilobytes written to disk
	disk_write_requests	✗	✗	✗	✓	✓	request	Disk write requests
NetworkRx	network_bytes_received	✗	✗	✗	✓	✗	bytes	Network bytes received
	network_packets_received	✗	✗	✗	✗	✓	packet	Number of incoming packets
	network_data_rx_rate_average	✓	✓	✗	✗	✗	kB/s	Average rate at which data was received
NetworkTx	network_bytes_transmitted	✗	✗	✗	✓	✗	bytes	Network bytes transmitted
	network_packets_transmitted	✗	✗	✗	✗	✓	packet	Number of outgoing packets
	network_data_tx_rate_average	✓	✓	✗	✗	✗	kB/s	Average rate at which data was transmitted

Figure 22 KPI supported for builtin monitors

Please note that in future releases more number of monitors along with their counters will be supported.

3.5 Action modeling

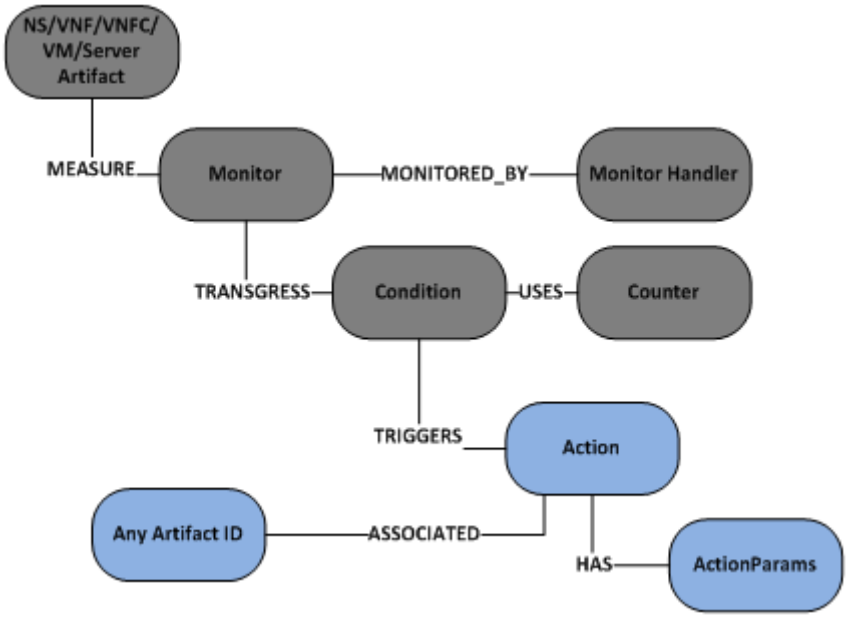


Figure 23 Action Modeling Artifacts

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 exe-cute	Valid only when Type is Script

Table 11 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 12 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 Integration between SiteScope alerts and UCA-EBC

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

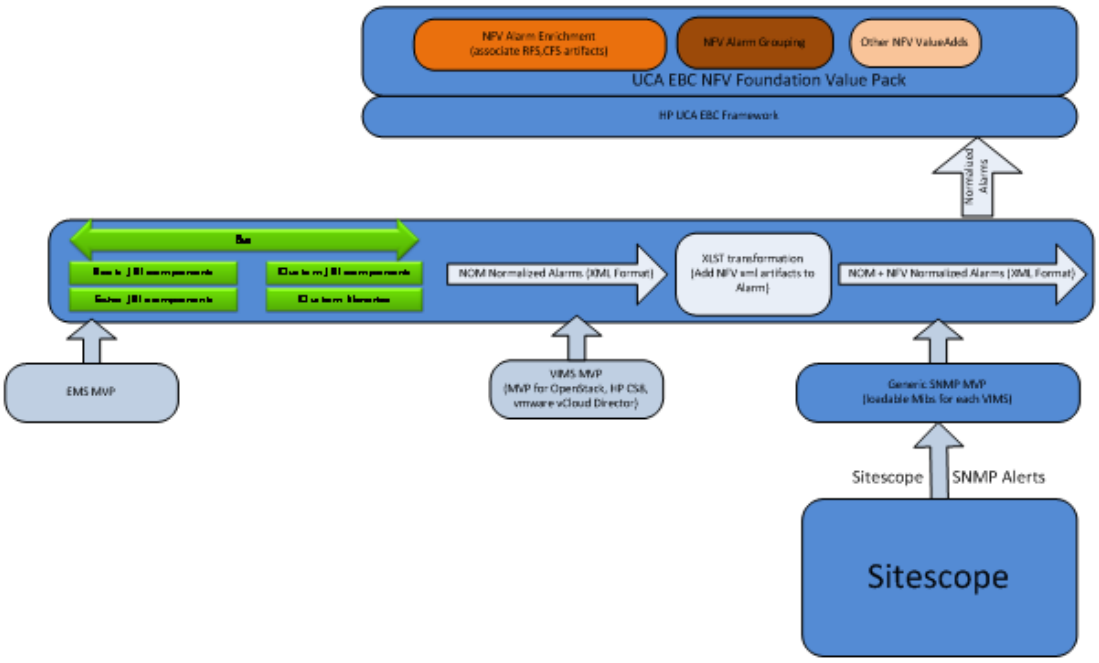


Figure 24 Integration of monitoring threshold crossing alerts for Correlation

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 NOM Bus, the NOM 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 NOM Bus.

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

See the HP NFVD User Guide for the procedure to integrate `NFVD_SNMP_TEMPLATE_XML.xml` SNMP alert for any monitor.

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 NOM Bus.

3.7 Embedded VNF manager

3.7.1 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
    a. Create a VNF instance (need a template id) in the database using a template following the policies defined in the template.
    b. This operation creates only the VNF structure in the database.
2. WF_NFVD_DELETE_INSTANCE
    a. The equivalent delete operation of the previous one (need an instance id).
3. ACTIVATE/DEACTIVATE
    a. 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
    a. This workflow allows allocating resources from the servers of a datacenter, for the Virtual machines of the VNF.
    b. 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.
    c. 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
    a. 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
    a. These two workflows scale in or out a VNF (they need the instance id).
    b. They look at the configuration of the instance policies and the configuration of the template policies.

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 NOM 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 5 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 NFVD1.0, 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 NFVD1.0 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 Moni-
  tor|groupdescription=CPU|groupID=201070542|id=1</additionalText>
    <notificationIdentifier>0</notificationIdentifier>
    <correlationNotificationIdentifiers>Attribute not availa-
  ble</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 13 NFVD Alarm attributes

4.4 Custom automated action

4.4.1 Defining NFVD problem-action framework in UCA automation

See section 7 in HP UCA Automation V1.0 - Administrator and User Interface Guide V1.0 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.

4.4.2 Inventory data populated in UCA automation foundation value pack inventory

NFVD domain specific service definition

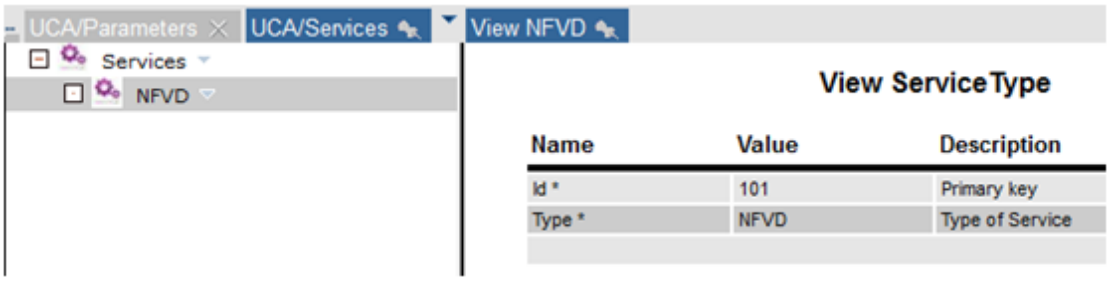


Figure 25 UCA Automation Foundation Inventory – UCA/Services

Action definitions for NFVD

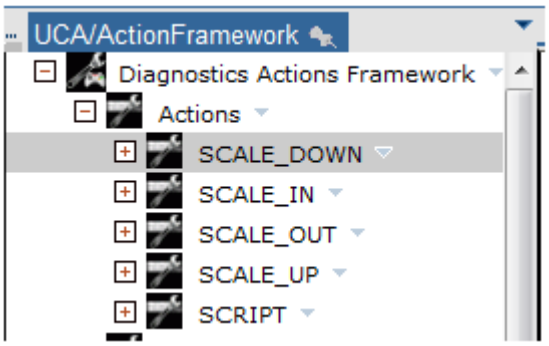


Figure 26 UCA Automation Foundation Inventory – UCA/Services

Problem definitions for NFVD:

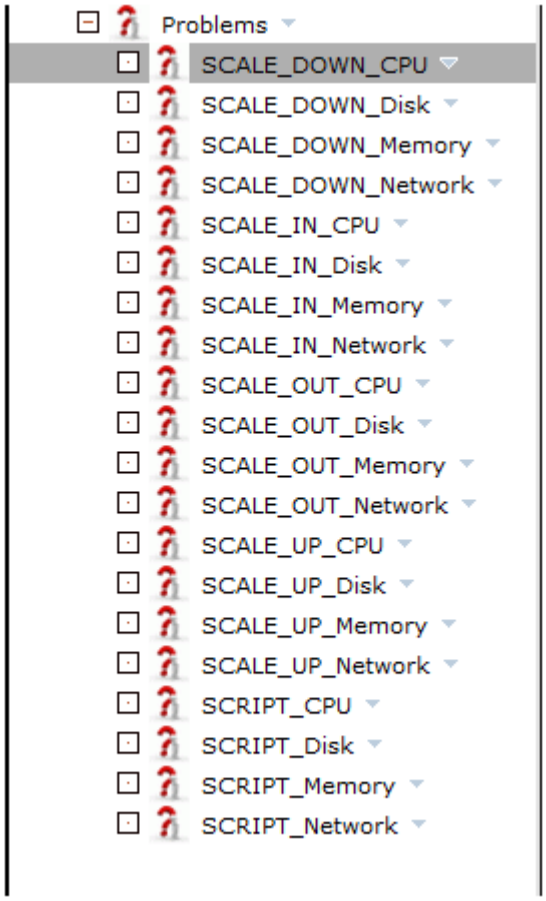


Figure 27 UCA Automation Foundation Inventory – UCA/ActionFramework (Problems)

NFVD Controller workflow mapping definition:

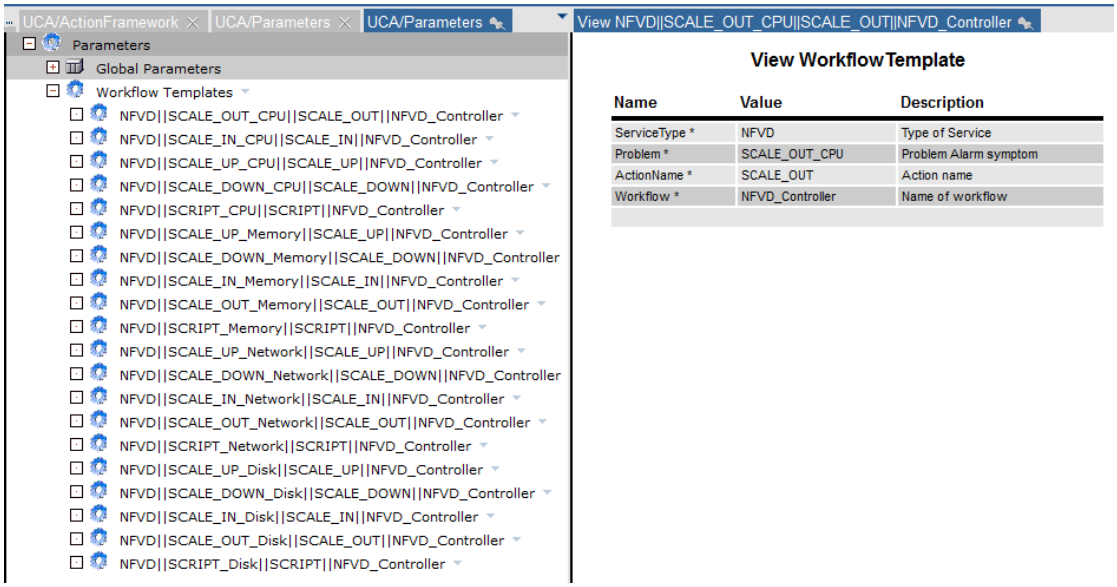


Figure 28 UCA Automation Foundation Inventory – UCA/Parameters

4.4.3 Inventory data populated in NFVD value pack inventory

Action mapping to NFVD child workflows

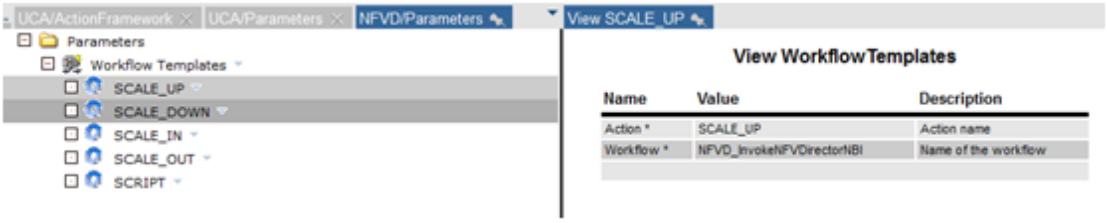
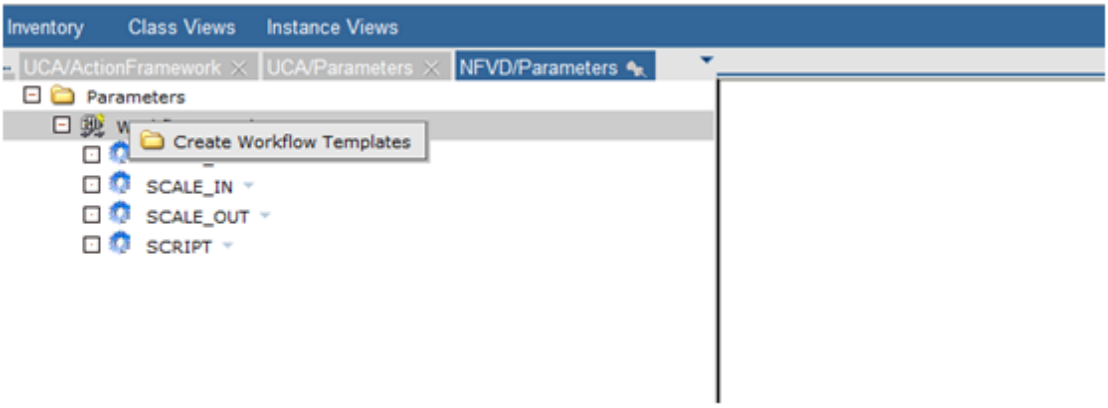


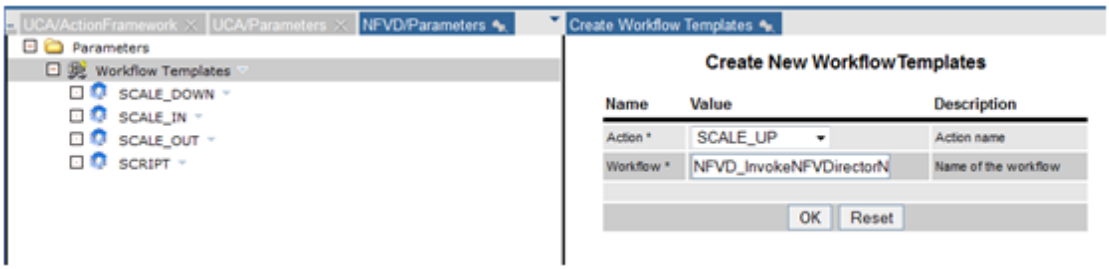
Figure 29 NFVD Inventory – NFVD/Parameters

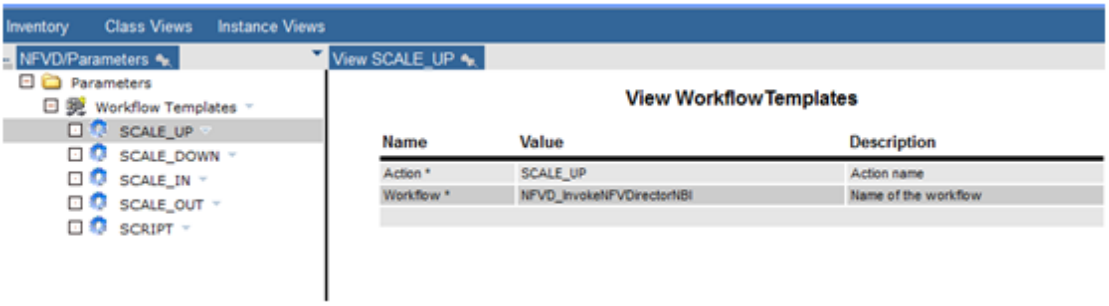
4.4.4 NFVD/Parameters

NFVD/Parameters provides mapping of the actions to the child workflows of NFVD.
Create a new workflow template by performing a right click on NFVD/Parameters → Parameters → Create Workflow Template.



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





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

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.

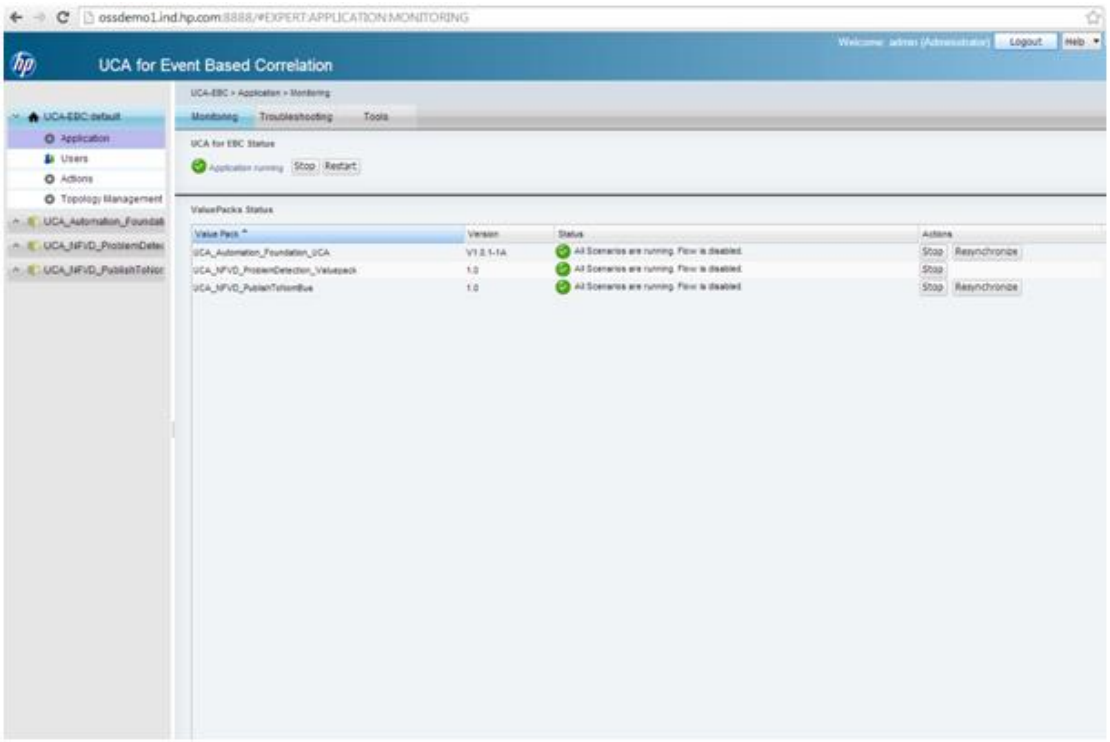
When you invoke the VNF CREATE WebService API, the new custom monitors will get deployed and activated as part of VNF CREATE process.

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.



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

Appendix A

VNF management

A.1 VNF operation

1. **NFVD | CREATE | VNF (to creat a VNF)**

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ngws="http://www.hp.com/sosa/protocoladapter/ngws">
    <soapenv:Header/>
    <soapenv:Body>
      <ngws:startServiceOrderAsync>
        <ngws:type>NFVD</ngws:type>
        <ngws:name>VNF</ngws:name>
        <ngws:action>CREATE</ngws:action>
        <!--Optional:-->
        <ngws:inputParams>
```

```
<!--Zero or more repetitions:-->
<ngws:param>
  <ngws:name>templateID</ngws:name>
  <ngws:value>TEST_2_Server</ngws:value>
</ngws:param>
<ngws:param>
  <ngws:name>parentArtifactId</ngws:name>
  <ngws:value>14020746143391</ngws:value>
</ngws:param>
<ngws:param>
  <ngws:name>relationshipType</ngws:name>
  <ngws:value>CONTAINS</ngws:value>
</ngws:param>
<ngws:param>
  <ngws:name>assignmentRelationshipID</ngws:name>
  <ngws:value>14018960162001</ngws:value>
</ngws:param>
<ngws:param>
  <ngws:name>resourceTreeID</ngws:name>
  <ngws:value>14019920442251</ngws:value>
</ngws:param>
</ngws:inputParams>
<ngws:user>hpsa</ngws:user>
</ngws:startServiceOrderAsync>
</soapenv:Body>
</soapenv:Envelope>
```

2. **NFVD | DELETE| VNF(to delete a VNF)**

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ngws="http://www.hp.com/sosa/protocoladapter/ngws">
<soapenv:Header/>
<soapenv:Body>
  <ngws:startServiceOrderAsync>
    <ngws:type>NFVD</ngws:type>
  <ngws:name>VNF</ngws:name>
  <ngws:action>DELETE </ngws:action>
  <!--Optional:-->
  <ngws:inputParams>
    <!--Zero or more repetitions:-->
```

```
            <ngws:param>
              <ngws:name>templateID</ngws:name>
              <ngws:value>TEST_2_Server</ngws:value>
            </ngws:param>
          </ngws:inputParams>
          <ngws:user>hpsa</ngws:user>
        </ngws:startServiceOrderAsync>
      </soapenv:Body>
    </soapenv:Envelope>
```

3. **NFVD | SCALE_IN| VM (to scale in a VNF)**

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ngws="http://www.hp.com/sosa/protocoladapter/ngws"
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
    <soapenv:Header/>
    <soapenv:Body>
      <ngws:startDynamicOrderSync>
        <dyn:serviceRequest>
          <dyn:services mode="parallel" onerror="rollback" persis-
tence="enable" name="INVENTORY" type="NFVD" action="SCALE_IN">
            <dyn:service>
              <dyn:name>INVENTORY</dyn:name>
              <dyn:type>NFVD</dyn:type>
              <dyn:action>SCALE_IN</dyn:action>
              <dyn:characteristics>
                <dyn:characteristic>
                  <dyn:name>ArtifactInstanceId</dyn:name>
                  <dyn:value>14018985927261</dyn:value>
                </dyn:characteristic>
              </dyn:characteristics>
            </dyn:service>
          </dyn:services>
        </dyn:serviceRequest>
        <ngws:user>dummy</ngws:user>
      </ngws:startDynamicOrderSync>
    </soapenv:Body>
  </soapenv:Envelope>
```

4. **NFVD | SCALE_OUT| VM (to scale out a VNF)**

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

```
xmlns:ngws="http://www.hp.com/sosa/protocoladapter/ngws"
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
<soapenv:Header/>
 <soapenv:Body>
    <ngws:startDynamicOrderSync>
      <dyn:serviceRequest>
 <dyn:services mode="parallel" onerror="rollback" persistence="enable"
name="INVENTORY" type="NFVD" action="SCALE_OUT">
  <dyn:service>
 <dyn:name>INVENTORY</dyn:name>
 <dyn:type>NFVD</dyn:type>
            <dyn:action>SCALE_OUT</dyn:action>
            <dyn:characteristics>
              <dyn:characteristic>
                <dyn:name>ArtifactInstanceId</dyn:name>
                <dyn:value>14018985927261</dyn:value>
              </dyn:characteristic>
            </dyn:characteristics>
          </dyn:service>
        </dyn:services>
      </dyn:serviceRequest>
      <ngws:user>dummy</ngws:user>
    </ngws:startDynamicOrderSync>
  </soapenv:Body>
</soapenv:Envelope>
```

A.2 Monitor

1. **NFVD | DEPLOY| MONITOR (to deploy a MONITOR)**

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ngws="http://www.hp.com/sosa/protocoladapter/ngws"
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
  <soapenv:Header/>
  <soapenv:Body>
  <ngws:startDynamicOrderSync>
<dyn:serviceRequest
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
 <dyn:services type="NFVD" name="MONITOR" action="DEPLOY">
 <dyn:service>
 <dyn:name>MONITOR</dyn:name>
```

```
<dyn:type>NFVD</dyn:type>
<dyn:action>DEPLOY</dyn:action>
<dyn:characteristics>
<dyn:characteristic>
<dyn:name>ArtifactInstanceId</dyn:name>
<dyn:value>memoryMonitor-002</dyn:value>
</dyn:characteristic>
</dyn:characteristics>
</dyn:service>
</dyn:services>
</dyn:serviceRequest>
<ngws:user>12</ngws:user>
</ngws:startDynamicOrderSync>
</soapenv:Body>
</soapenv:Envelope>
```

2. **NFVD | UNDEPLOY| MONITOR (to undeploy a MONITOR)**

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ngws="http://www.hp.com/sosa/protocoladapter/ngws"
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
 <soapenv:Header/>
 <soapenv:Body>
<ngws:startDynamicOrderSync>
<dyn:serviceRequest
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
  <dyn:services type="NFVD" name="MONITOR" action="UNDEPLOY">
   <dyn:service>
   <dyn:name>MONITOR</dyn:name>
   <dyn:type>NFVD</dyn:type>
        <dyn:action>UNDEPLOY</dyn:action>
        <dyn:characteristics>
          <!--1 or more repetitions:-->
          <dyn:characteristic>
            <dyn:name>ArtifactInstanceId</dyn:name>
            <dyn:value> memoryMonitor-002</dyn:value>
          </dyn:characteristic>
        </dyn:characteristics>
      </dyn:service>
  </dyn:services>
```

```
</dyn:serviceRequest>
<ngws:user>12</ngws:user>
</ngws:startDynamicOrderSync>
</soapenv:Body>
    </soapenv:Envelope>
```

3. **NFVD | START| MONITOR (to start a MONITOR)**

```
    <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ngws="http://www.hp.com/sosa/protocoladapter/ngws"
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
  <soapenv:Header/>
  <soapenv:Body>
    <ngws:startDynamicOrderSync>
<dyn:serviceRequest
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
  <dyn:services type="NFVD" name="MONITOR" action="START">
    <dyn:service>
      <dyn:name>MONITOR</dyn:name>
      <dyn:type>NFVD</dyn:type>
      <dyn:action>START</dyn:action>
      <dyn:characteristics>
        <dyn:characteristic>
          <dyn:name>ArtifactInstanceId</dyn:name>
          <dyn:value>memoryMonitor-002</dyn:value>
        </dyn:characteristic>
      </dyn:characteristics>
    </dyn:service>
  </dyn:services>
</dyn:serviceRequest>
      <ngws:user>12</ngws:user>
    </ngws:startDynamicOrderSync>
  </soapenv:Body>
    </soapenv:Envelope>
```

4. **NFVD | STOP| MONITOR (for stopping a MONITOR)**

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ngws="http://www.hp.com/sosa/protocoladapter/ngws"
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
      <soapenv:Header/>
```

```
        <soapenv:Body>
     <ngws:startDynamicOrderSync>
<dyn:serviceRequest
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
         <dyn:services type="NFVD" name="MONITOR" action="STOP">
       <dyn:service>
         <dyn:name>MONITOR</dyn:name>
         <dyn:type>NFVD</dyn:type>
         <dyn:action>STOP</dyn:action>
         <dyn:characteristics>
            <dyn:characteristic>
               <dyn:name>ArtifactInstanceId</dyn:name>
               <dyn:value> memoryMonitor-002</dyn:value>
            </dyn:characteristic>
         </dyn:characteristics>
       </dyn:service>
   </dyn:services>
</dyn:serviceRequest>
        <ngws:user>12</ngws:user>
     </ngws:startDynamicOrderSync>
  </soapenv:Body>
    </soapenv:Envelope>
```

Appendix B

OpenStack plug-in operations

The following operations are possible.

- Create, Edit, Query, and Delete a Virtual Machine.
- Query for an Image.
- Create, Query and Delete a Flavor.
- Create, Query, Edit and Delete Networks and Subnets.

All listed operations can be executed from CS8 User Interface. However, automation is not available for these processes. The Rest Northbound Interface is implemented for that purpose.

B.1 OpenStack templates

OpenStack plug-in uses templates for communicating the commands to the hypervisor. These templates should be of the same format as the JSON request for OpenStack. These expected request formats can be found in the OpenStack API documentation at:

`http://developer.OpenStack.org/api-ref-compute-v2.html`

You should create a new template for each new operation to be implemented. Templates can be created or listed and edited through the HPSA Solution Container: `Administrator> ECP > Activation Commands Template > Template`.

Figure 30 HPSA Solution Container ECP command template

The following is an example of creating a server template.

```
[TEMPLATE:Config]
http.operation=POST

#http.url.sufix=/v2/${tenant_id}/servers

http.url.sufix=/servers

openstack.endpointtype=compute

[TEMPLATE:Do]

[TEMPLATE:Section 0]
{

   "server":{

      "name":"${SERVER_NAME}",

      "imageRef":"${IMAGE}",

      "flavorRef":"${FLAVOR}",

      "max_count":1,

      "min_count":1,

      "networks":[

         {

            "uuid":"${SERVER_NETWORK_ID}"

         }

      ],

      "security_groups":[

         {

            "name":"default"

         }
```

Figure 31 Example server template

The template is organized in two sections:

- Config: In the Config section, the following details are provided.
  - Operation:
    - POST for creating
    - PUT for editing
    - GET for querying
    - DELETE for deleting
    - http.url.sufix: refers to the path in the API.
  - Type of endpoint (depending on the operation):

- Compute for virtual machines and images
- Network for networks

- 0: Section 0 is specified as the concrete JSON request expected by the OpenStack API. The JSON is a compounded structure with pairs (variable:value). Variables like `${SERVER_NAME}` can be inserted in the template and the plug-in replaces it with a value passed through the specific workflow.

B.2 OpenStack workflows

In the current version, a unique activation workflow is deployed for each necessary operation. The structure in the workflow is always the same.

- Get values from outside
  - Authentication Values
  - Activation Values (Server Name, Network UUID)
- Add Activation Values inside a HashMap.
- Invoke the plug-in with those values.
  - Authenticate
  - Execute concrete operation
- Check correct activation.
- If activation was `OK` get the OpenStack Response into an object.
- Send the object to the workflow caller.

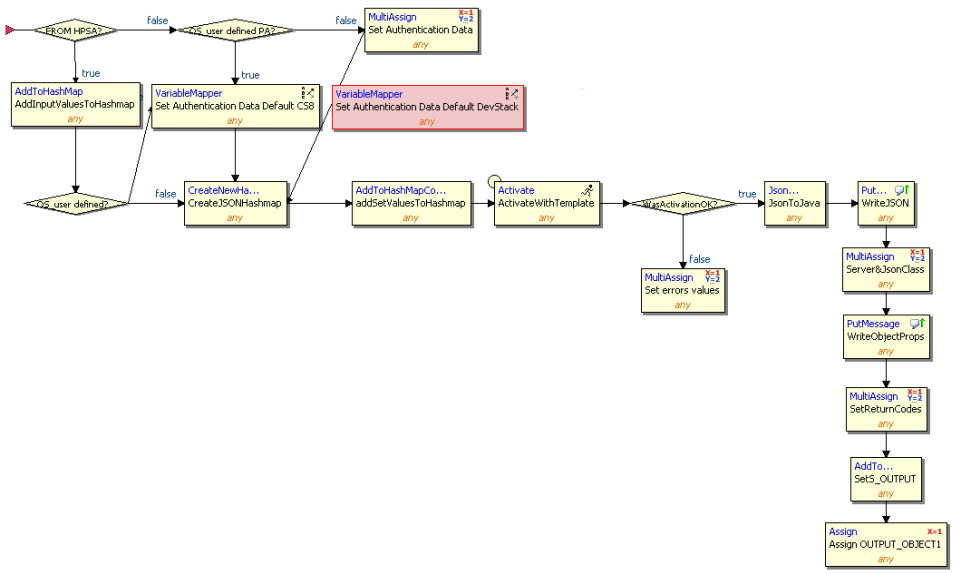


Figure 32 Example: Create Server Workflow

B.2.1 CS8-REST interface

The Rest interface helps to automate the OpenStack operations. A Rest client is required to run those operations. This section explains the procedure to call operations using the Firefox Rest Client.

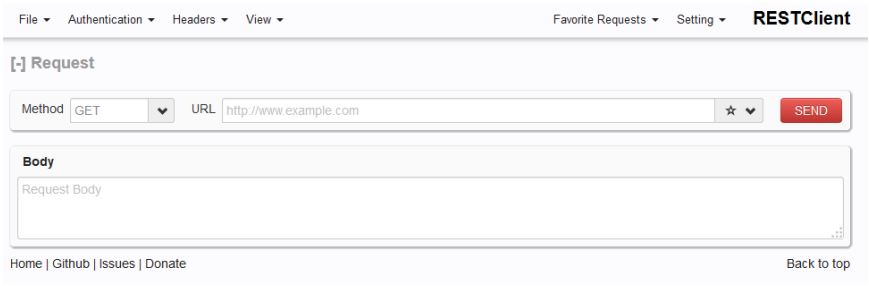


Figure 33 CloudSystem Firefox Rest Client

You must indicate proper headers.

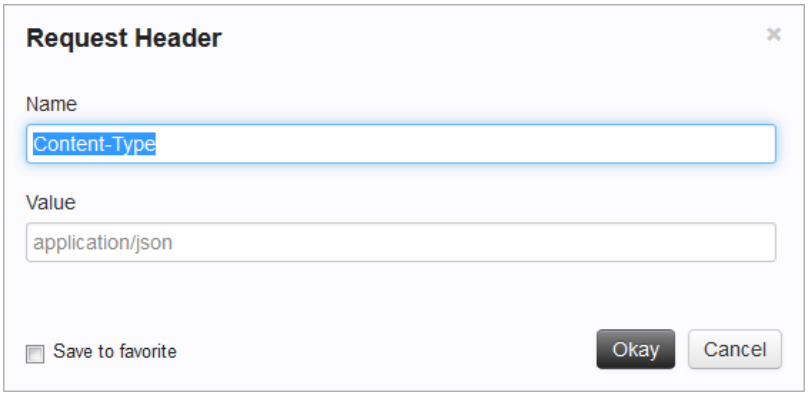


Figure 34 CloudSystem Rest Client Request Header

B.2.2 Operations

B.2.2.1 Create server

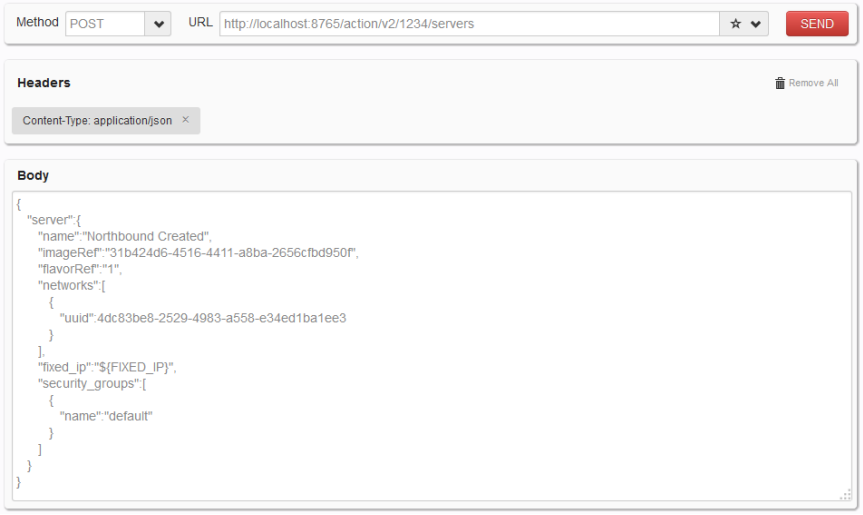


Figure 35 CloudSystem Create Server operation

URL: `http://localhost:8765/action/v2/1234/servers`
`http:$host:$Protocol_Adapter_port/action/$version_api/$tenant/servers`

B.2.2.2 Edit server

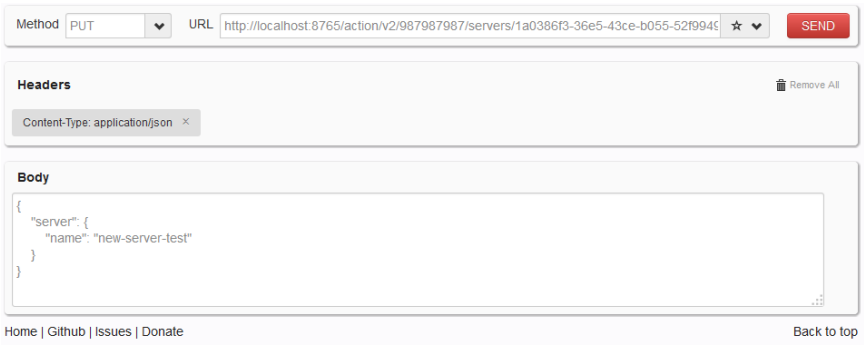


Figure 36 CloudSystem Edit Server operation

URL:

`http://localhost:8765/action/v2/987987987/servers/1a0386f3-36e5-43ce-b055-52f994924e36`

`http:$host:$Protocol_Adapter_port/action/$version_api/$tenant/servers/$serverId`

B.2.2.3 Get server by ID

Figure 37 CloudSystem Query Server operation

URL:

`http://localhost:8765/action/v2/987987987/servers/1a0386f3-36e5-43ce-b055-52f994924e36`

`http:$host:$Protocol_Adapter_port/action/$version_api/$tenant/servers/$serverId`

B.2.2.4 Delete server

Figure 38 CloudSystem Delete Server operation

URL:

`http://localhost:8765/action/v2/987987987/servers/1a0386f3-36e5-43ce-b055-52f994924e36`

`http:$host:$Protocol_Adapter_port/action/$version_api/$tenant/servers/$serverId`

B.2.2.5 Start server

Figure 39 CloudSystem Start Server operation

URL:

`http://localhost:8765/action/v2/987987987/servers/1a0386f3-36e5-43ce-b055-52f994924e36`

`http:$host:$Protocol_Adapter_port/action/$version_api/$tenant/servers/$serverId`

B.2.2.6 Stop server

Figure 40 CloudSystem Stop Server operation

URL:

`http://localhost:8765/action/v2/987987987/servers/1a0386f3-36e5-43ce-b055-52f994924e36`

`http:$host:$Protocol_Adapter_port/action/$version_api/$tenant/servers/$serverId`

B.2.2.7 Query image

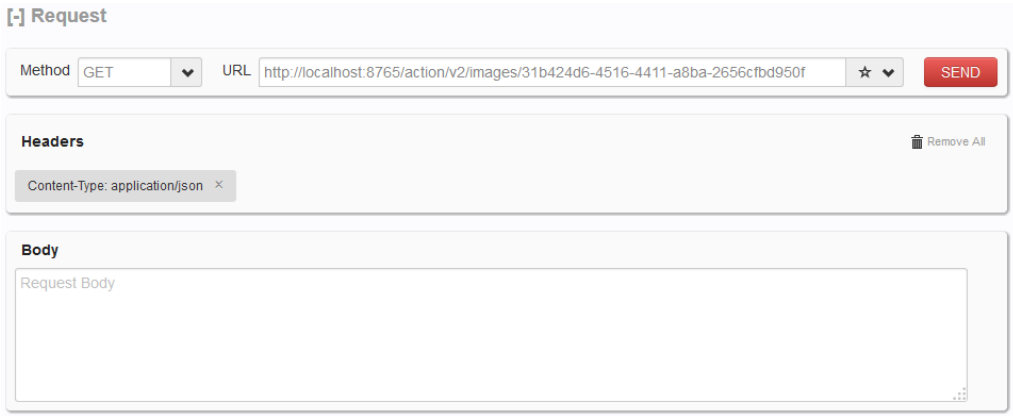


Figure 41 CloudSystem Query Image operation

URL:

`http://localhost:8765/action/v2/images/31b424d6-4516-4411-a8ba-2656cfbd950f`
`http:$host:$Protocol_Adapter_port/action/$version_api/images/$imageId`

B.2.2.8 Create flavor

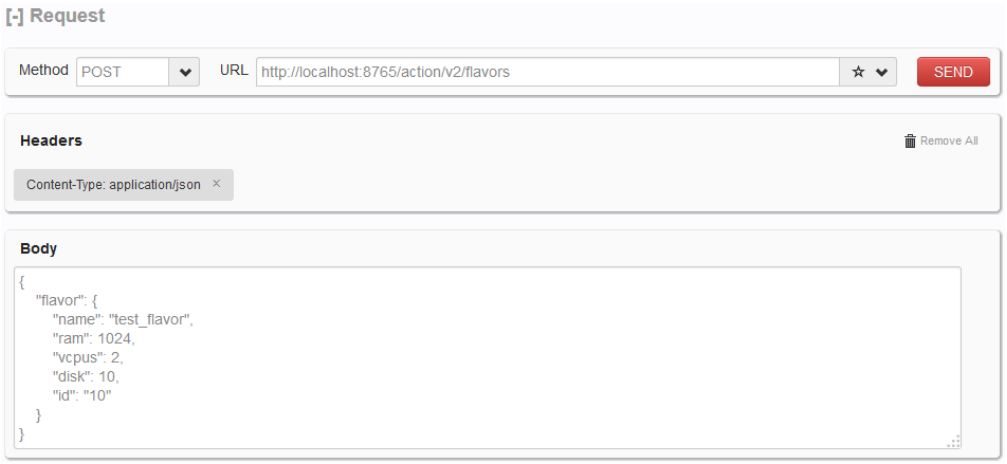


Figure 42 CloudSystem Create Flavour operation

URL:

```
http://localhost:8765/action/v2/flavors
http:$host:$Protocol_Adapter_port/action/$version_api/flavors
```

B.2.2.9 Get flavor

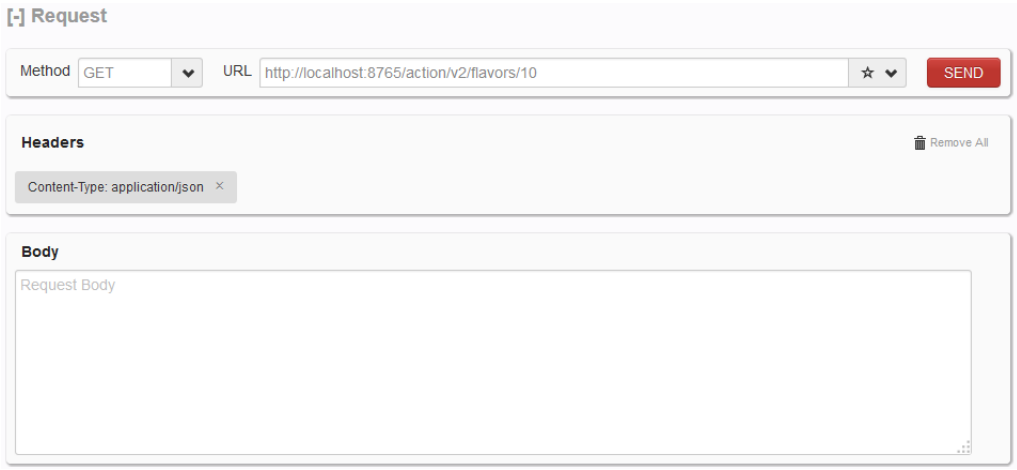


Figure 43 CloudSystem Query Flavor operation

URL:

```
http://localhost:8765/action/v2/flavors/10
http:$host:$Protocol_Adapter_port/action/$version_api/flavors/$flavor
Id
```

B.2.2.10 Get flavor by parameters

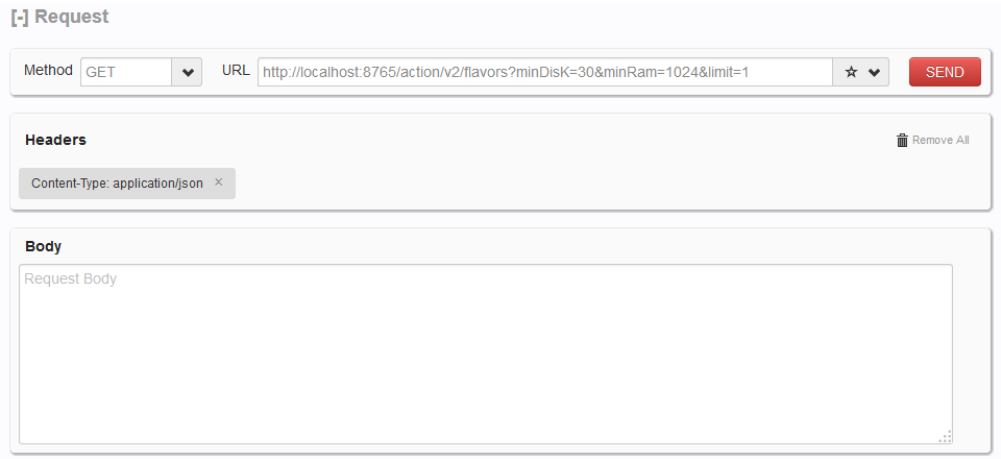


Figure 44 CloudSystem Query Flavor by Parameters operation

URL:

```
http://localhost:8765/action/v2/flavors?minDisK=30&minRam=1024&limit=1
http:$host:$Protocol_Adapter_port/action/?$param1=$value1&$param2=$value2
```

B.2.2.11 Delete flavor

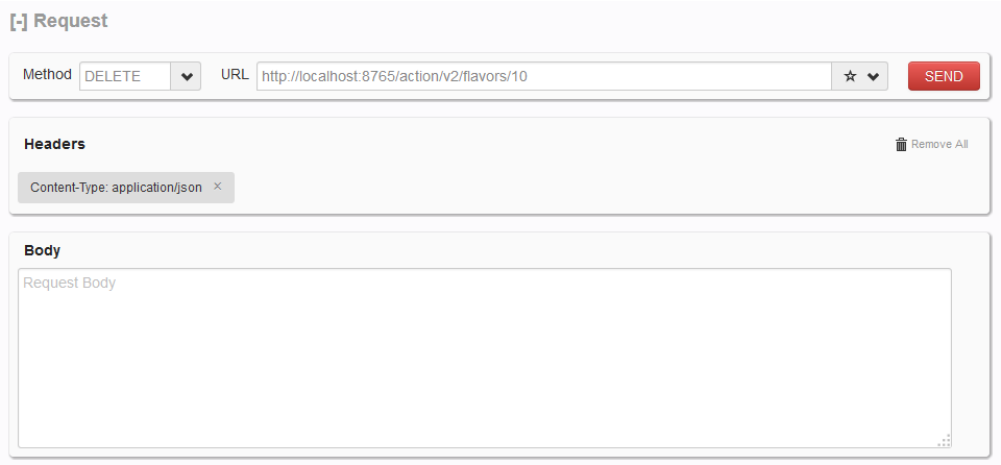


Figure 45 CloudSystem Delete Flavor operation

URL:

```
http://localhost:8765/action/v2/flavors/10
http:$host:$Protocol_Adapter_port/action/$version_api/flavors/$flavorId
```

B.2.2.12 Create network

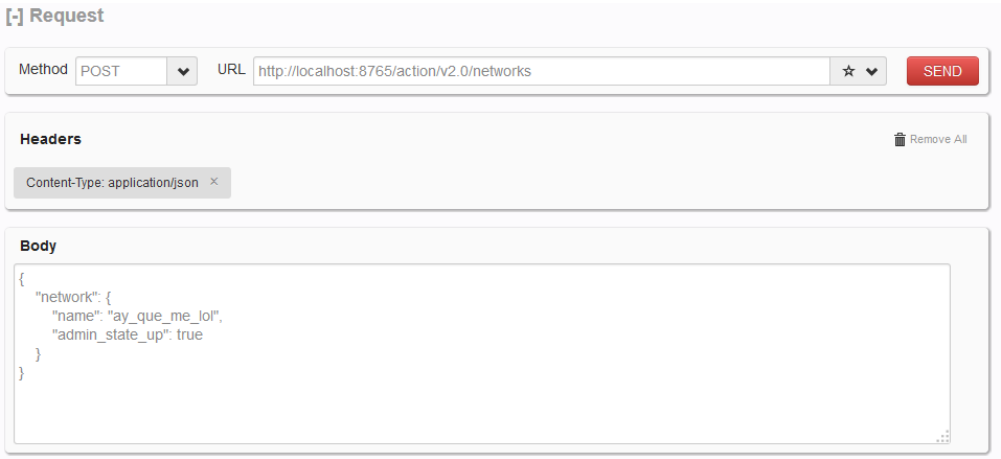


Figure 46 CloudSystem Create Network operation

URL:

```
http://localhost:8765/action/v2.0/networks
http:$host:$Protocol_Adapter_port/action/$version_api/networks
```

B.2.2.13 Edit network

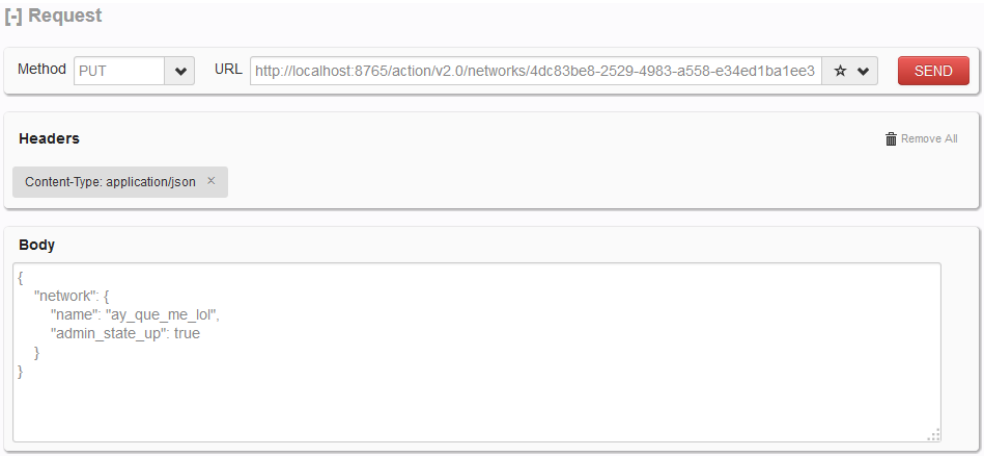


Figure 47 CloudSystem Edit Network operation

URL:

```
http://localhost:8765/action/v2.0/networks/4dc83be8-2529-4983-a558-
e34ed1ba1ee3
http:$host:$Protocol_Adapter_port/action/$version_api/networks/$netwo
rk_id
```

B.2.2.14 Get network by ID

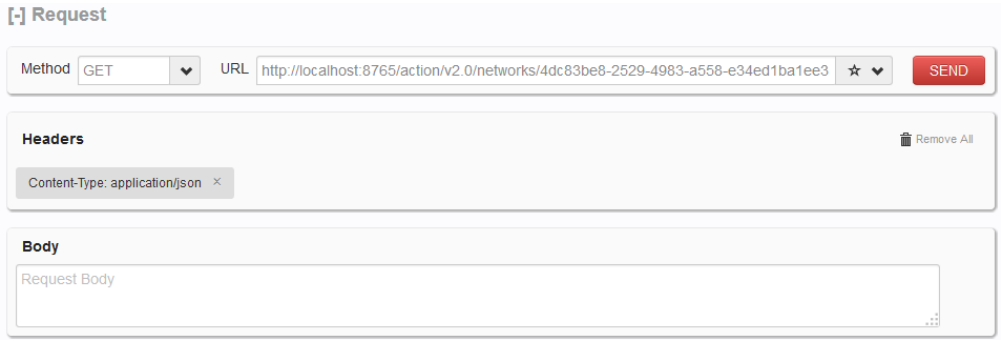


Figure 48 CloudSystem Query Network by ID operation

URL:

```
http://localhost:8765/action/v2.0/networks/4dc83be8-2529-4983-a558-
e34ed1ba1ee3
http:$host:$Protocol_Adapter_port/action/$version_api/networks/$network_id
```

B.2.2.15 Get network by parameters

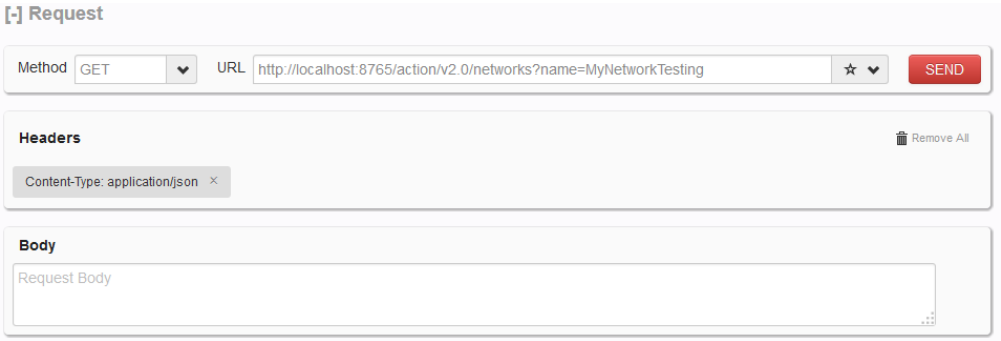


Figure 49 CloudSystem Query Network by Parameters operation

URL:

```
http://localhost:8765/action/v2.0/networks?name=MyNetworkTesting
http:$host:$Protocol_Adapter_port/action/$version_api/networks?$param
1=$value1
```

B.2.2.16 Delete network

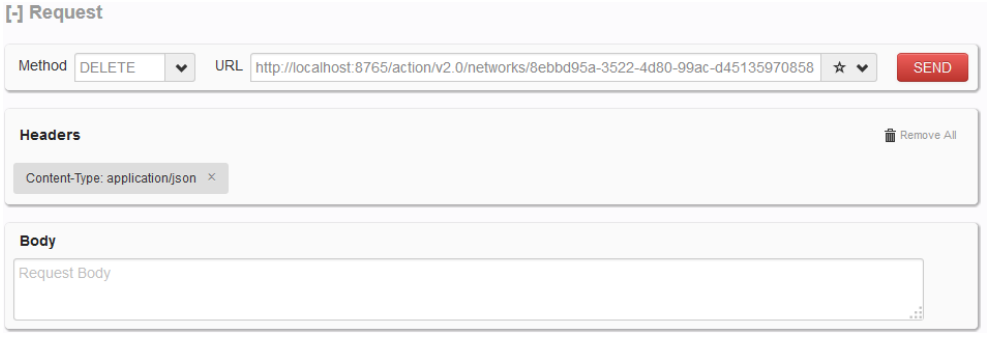


Figure 50 CloudSystem Delete Network operation

URL:

```
http://localhost:8765/action/v2.0/networks/4dc83be8-2529-4983-a558-e34ed1ba1ee3
http:$host:$Protocol_Adapter_port/action/$version_api/networks/$network_id
```

B.2.2.17 Create subnet

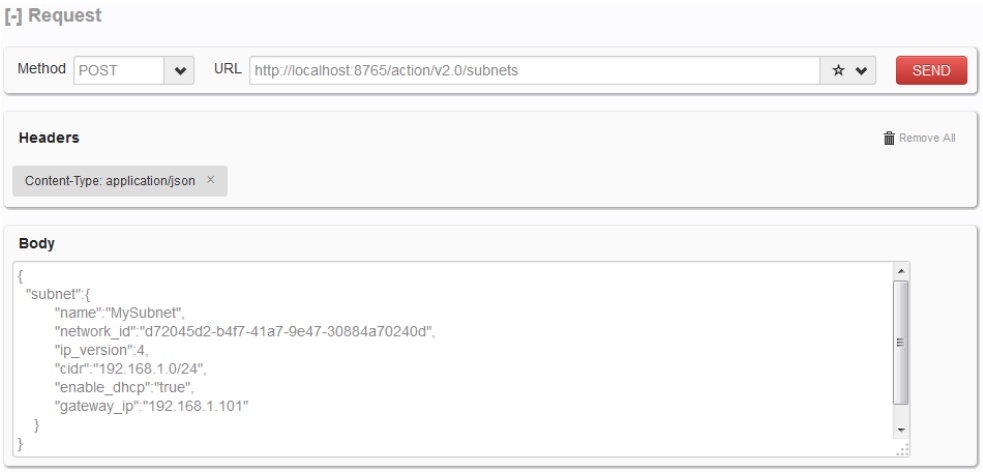


Figure 51 CloudSystem Create Subnet operation

URL:

```
http://localhost:8765/action/v2.0/subnets
http:$host:$Protocol_Adapter_port/action/$version_api/subnets
```

B.2.2.18 Edit subnet

Figure 52 CloudSystem Edit Subnet operation

URL:

```
http://localhost:8765/action/v2.0/subnets
http:$host:$Protocol_Adapter_port/action/$version_api/subnets/$subnet
_id
```

B.2.2.19 Get subnet

Figure 53 CloudSystem Query Subnet operation

URL:

```
http://localhost:8765/action/v2.0/subnets
http:$host:$Protocol_Adapter_port/action/$version_api/subnets/$subnet
_id
```

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 OSS Open Mediation V6.2

JDK: Java Development Kit

JMS: Java Messaging Service

JMX: Java Management extension, used to access or process action on the UCA for EBC 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: UCA for EBC Value Pack