HP NFV Director

HP NFV Director

Version 3.0

User Guide - Advanced

Edition: 1.0

For Red Hat Enterprise Linux Server 6.6

June 2015

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.

HP-UX Release 10.20 and later and HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

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

Microsoft®, Internet Explorer, Windows®, Windows Server 2007®, Windows XP®, and Windows 7® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Firefox® is a registered trademark of the Mozilla Foundation.

Google Chrome® is a trademark of Google Inc.

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

EnterpriseDB® is a registered trademark of EnterpriseDB.

Postgres Plus® Advanced Server is a registered U.S. trademark of EnterpriseDB.

UNIX® is a registered trademark of The Open Group.

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

Red Hat® is a registered trademark of the Red Hat Company.

Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.

Neo4j is a trademark of Neo Technology.

Contents

Legal Notices **2**

Contents **3**

Tables **11**

Figures **12**

Preface **16**

In this Guide 16
Audience 16
Typographical Conventions 17
Associated Documents 17
References 17

Chapter 1 ..... **18**

Introduction **18**

Chapter 2 ..... **19**

NFV Director Data Model **19**

2.1 NFV Director modeling philosophy 19
2.1.1 Artifact 19
2.1.2 Relationship 20
2.2 Flavors of artifacts and relationships 20
2.2.1 Definition 21
2.2.2 Template 22
2.2.3 Instance 23
2.3 Out-of-the-box models 24
2.3.1 Virtual Network Function model 24
2.3.2 Resources model 26
2.3.3 Monitoring model 27
2.3.4 Overview model 31
2.3.5 Datacenter Artifacts 36
2.3.6 Policy Artifacts 36
2.4 Particularities 36
2.4.1 Relationships 36
2.5 Artifact and Relationship 37
2.5.1 Artifacts 37
2.5.2 Relationships 39
2.6 Examples 40
2.6.1 Example 1: Basic VNF 40
2.6.2 Example 2: Application Server 41
2.7 NFV Director Data Model: Soap UI 42
2.7.1 Introduction 42

2.8 Data Model 43

Chapter 3 46

Northbound Interface 46

3.1 Data Model 47
3.2 Automation 47
3.3 SOSA as NFVD operator 47
3.3.1 Understanding SOAP requests 47
3.3.2 Operations 48
3.4 Northbound Interface – Automation 54
3.4.1 VNFs actions 55
3.4.2 Inventory actions 59
3.4.3 Monitoring actions 62
3.4.4 Network Service actions 62
3.4.5 NS actions 62
3.4.6 Virtual Core actions 63
3.4.7 Virtual Memory actions 63
3.4.8 VNF Manager actions 63
3.4.9 Orchestration actions 64
3.5 Alarms and notifications interface 67
3.6 Access to NFV Director: Soap UI 69
3.6.1 Data Model 70
3.6.2 Automation 73

Chapter 4 75

Southbound Interface 75

4.1 GENERIC # VNFM # NFVO_TO_VNFM_v1 77
4.1.1 CHANGE_FLAVOR command 77
4.1.2 CREATE_VNF command 77
4.1.3 CREATE_VNF_DESCRIPTOR command 78
4.1.4 DELETE_VNF command 78
4.1.5 GET_JOB_STATUS command 79
4.1.6 GET_LIST_ORCHESTATOR_REGISTERED command 79
4.1.7 GET_VNF_DESCRIPTOR_DETAILS command 79
4.1.8 GET_VNF_DESCRIPTOR_LIST command 79
4.1.9 GET_VNF_DETAILS command 79
4.1.10 GET_VNF_LIST command 79
4.1.11 REGISTER_ORCHESTATOR command 80
4.1.12 SCALE_DOWN_VDU command 80
4.1.13 SCALE_DOWN_VM command 81
4.1.14 SCALE_DOWN_VNF command 81
4.1.15 SCALE_IN_VDU command 82
4.1.16 SCALE_IN_VM command 82
4.1.17 SCALE_IN_VNFcommand 83
4.1.18 SCALE_OUT_VDU command 83
4.1.19 SCALE_OUT_VM command 84
4.1.20 SCALE_OUT_VNF command 84
4.1.21 SCALE_UP_VDU command 85
4.1.22 SCALE_UP_VM command 85
4.1.23 SCALE_UP_VNF command 86

4.1.24 UNREGISTER_ORCHESTATOR command 86
4.2 HP # HP_Juno_v2 # HP_NEUTRON 86
4.2.1 ATTACH_VIP_SERVER command 86
4.2.2 CREATE_NETWORK command 87
4.2.3 CREATE_PORT command 88
4.2.4 CREATE_SG command 88
4.2.5 CREATE_SG_RULE command 88
4.2.6 CREATE_SUBNET command 89
4.2.7 CREATE_VIP command 89
4.2.8 DELETE_NETWORK command 90
4.2.9 DELETE_SUBNET command 90
4.2.10 QUERY_FLOATING_IP command 90
4.2.11 QUERY_NETWORK command 91
4.2.12 QUERY_NETWORK_BY_PARAMS command 91
4.2.13 QUERY_PORT command 91
4.2.14 QUERY_SUBNET command 91
4.2.15 UPDATE_PORT_SECURITY_GROUP command 91
4.2.16 UPDATE_SUBNET command 92
4.3 HP # HP_Juno_v2 # HP_NOVA 92
4.3.1 CREATE_FLAVOR command 92
4.3.2 CREATE_SERVER command 93
4.3.3 DELETE_FLAVOR command 93
4.3.4 DELETE_SERVER command 94
4.3.5 QUERY_FLAVOR command 94
4.3.6 QUERY_FLAVOR_BY_PARAMS command 94
4.3.7 QUERY_FROM_SERVER command 94
4.3.8 QUERY_IMAGE command 94
4.3.9 QUERY_IMAGE_BY_PARAMS command 95
4.3.10 QUERY_SERVER command 95
4.3.11 SERVER_ACTIONS command 95
4.3.12 START_SERVER command 96
4.3.13 STOP_SERVER command 96
4.3.14 UPDATE_SERVER command 96
4.4 OpenStack # OS_Havana_v2 # OS_NEUTRON 97
4.4.1 ATTACH_VIP_SERVER command 97
4.4.2 CREATE_NETWORK command 97
4.4.3 CREATE_SG command 98
4.4.4 CREATE_SG_RULE command 98
4.4.5 CREATE_SUBNET command 98
4.4.6 CREATE_VIP command 99
4.4.7 DELETE_NETWORK command 99
4.4.8 DELETE_SUBNET command 100
4.4.9 QUERY_FLOATING_IP command 100
4.4.10 QUERY_NETWORK command 100
4.4.11 QUERY_NETWORK_BY_PARAMS command 100
4.4.12 QUERY_PORT command 101
4.4.13 QUERY_SUBNET command 101
4.4.14 UPDATE_PORT_SECURITY_GROUP command 101
4.4.15 UPDATE_SUBNET command 101
4.5 OpenStack # OS_Havana_v2 # OS_NOVA 102
4.5.1 CREATE_FLAVOR command 102

4.5.2 CREATE_SERVER command 102
4.5.3 DELETE_FLAVOR command 103
4.5.4 DELETE_SERVER command 103
4.5.5 QUERY_FLAVOR command 103
4.5.6 QUERY_FLAVOR_BY_PARAMS command 104
4.5.7 QUERY_FROM_SERVER command 104
4.5.8 QUERY_IMAGE command 104
4.5.9 QUERY_IMAGE_BY_PARAMS command 104
4.5.10 QUERY_SERVER command 104
4.5.11 SERVER_ACTIONS command 105
4.5.12 START_SERVER command 105
4.5.13 STOP_SERVER command 105
4.5.14 UPDATE_SERVER command 106
4.6 OpenStack # OS_Icehoust_v2 # OS_NEUTRON 106
4.6.1 ATTACH_VIP_SERVER command 106
4.6.2 CREATE_NETWORK command 107
4.6.3 CREATE_SG command 107
4.6.4 CREATE_SG_RULE command 108
4.6.5 CREATE_SUBNET command 108
4.6.6 CREATE_VIP command 109
4.6.7 DELETE_NETWORK command 109
4.6.8 DELETE_SUBNET command 109
4.6.9 QUERY_FLOATING_IP command 109
4.6.10 QUERY_NETWORK command 110
4.6.11 QUERY_NETWORK_BY_PARAMS command 110
4.6.12 QUERY_PORT command 110
4.6.13 QUERY_SUBNET command 110
4.6.14 UPDATE_PORT_SECURITY_GROUP command 111
4.6.15 UPDATE_SUBNET command 111
4.7 OpenStack # OS_Icehouse_v2 # OS_NOVA 111
4.7.1 CREATE_FLAVOR command 111
4.7.2 CREATE_SERVER command 112
4.7.3 DELETE_FLAVOR command 113
4.7.4 DELETE_SERVER command 113
4.7.5 QUERY_FLAVOR command 113
4.7.6 QUERY_FLAVOR_BY_PARAMS command 113
4.7.7 QUERY_FROM_SERVER command 113
4.7.8 QUERY_IMAGE command 114
4.7.9 QUERY_IMAGE_BY_PARAMS command 114
4.7.10 QUERY_SERVER command 114
4.7.11 SERVER_ACTIONS command 114
4.7.12 START_SERVER command 115
4.7.13 STOP_SERVER command 115
4.7.14 UPDATE_SERVER command 115
4.8 OpenStack plug-in 116
4.8.1 OpenStack CS8 user interface 116
4.8.2 OpenStack plug-in operations 116
4.8.3 OpenStack templates 118
4.8.4 OpenStack Workflows 120
4.8.5 Workflows supported by NFV Director 120
4.9 CS8 – REST Interface 124

4.9.1 Operations 125
4.10 VIMs configuration 134

Chapter 5 135

Creating VNF 135
5.1 Creating instances 135
5.2 Generating a template 136
5.3 Policies 136
5.3.1 Assignment 136
5.3.2 Validation 138
5.3.3 Range 139
5.4 Virtual Machines and VNF lifecycle 139
5.5 Examples 141
5.5.1 Example with Range Policy 141
5.5.2 Example with Assign Policy 143
5.6 Inside Orchestration 143

Chapter 6 144

VNF Resource assignment 144
6.1 Assignment Process 144
6.2 Policies 144
6.2.1 Assignment relationship 144
6.2.2 Over_subscription 145
6.2.3 Affinity 146
6.3 Process Description 147
6.4 Inside Orchestration 151

Chapter 7 152

Activation 152
7.1 Checking and getting the Tenant and VIM 153
7.2 Checking a flavor with RAM and disk 153
7.3 Getting Image ID and network ID 154
7.4 Creating Server 154
7.5 Updating Status 154
7.6 Activating workflow parent 155
7.6.1 Testing 155
7.7 Pre and post processing actions 155
7.8 Deactivating workflow 156
7.9 Error Recap 156
7.10 Inside Orchestration 156

Chapter 8 157

VNF Scaling 157
8.1 Scale-in 157
8.1.1 Scale in operation 158
8.1.2 End-to-End Example 160
8.1.3 Recap of End Messages 161
8.2 Scale-out 162

8.2.1 Scale-out operation 163
8.2.2 End-to-End Example 164
8.2.3 Recap of End Messages 165
8.3 Scale Up 166
8.3.1 Scale Up operation 166
8.3.2 End-to-End Example 167
8.3.3 Recap of End Messages 168
8.4 Scale Down 170
8.4.1 Scale Down operation 170
8.4.2 End-to-End Example 170
8.4.3 Recap of End Messages 171

Chapter 9 172

Network Service 172
9.1 Network Service 173
9.2 Supported Scenarios 174
9.2.1 Two VMs in same network 174
9.2.2 Two VMs connected through a third one 175
9.2.3 Two VMs connected with an external network through a VM 176

Chapter 10 179

Administering NFV Director Components 179
10.1 Stopping/Starting NFV Director 179
10.1.1 Starting all components 180
10.1.2 Stopping all NFV-Director components 180
10.1.3 Getting status of NFV-D components 180
10.1.4 Restarting NFV-Director components 180
10.2 Fulfillment components 180
10.2.1 Starting the Activator 180
10.2.2 Getting status of the Activator 180
10.2.3 Stopping the activator 180
10.2.4 Starting HP SOSA 180
10.2.5 Getting status of HP SOSA 180
10.2.6 Stopping HP SOSA 181
10.2.7 HP Equipment connection pool 181
10.2.8 Stopping HP ECP 181
10.2.9 Getting status of HP ECP 181
10.2.10 Starting HP Lock Manager 181
10.2.11 Getting status of HP Lock Manager 181
10.2.12 Stopping HP Lock Manager 181
10.3 Assurance components 182
10.3.1 Starting Assurance Gateway 182
10.3.2 Stopping Assurance Gateway 182
10.3.3 Getting status of Assurance Gateway 182
10.3.4 Restarting NFVD Assurance Gateway 182
10.3.5 Starting Open Mediation 182
10.3.6 Stopping Open Mediation 182
10.3.7 Getting status of Open Mediation 182
10.4 Monitoring components 182
10.4.1 Starting HP SiteScope 182

10.4.2 Stopping HP SiteScope 183
10.4.3 Getting status of HP SiteScope 183
10.4.4 Restarting HP SiteScope 183
10.5 Automation components 183
10.5.1 Starting HP UCA-EBC 183
10.5.2 Getting status of HP UCA-EBC 183
10.5.3 Stopping HP UCA-EBC 183
10.5.4 Starting HP UCA-automation 183
10.5.5 Getting status of HP UCA-automation 183
10.5.6 Stopping HP UCA-automation 184
10.6 NFVD GUI Components 184
10.6.1 Starting UOC 184
10.6.2 Getting status of UOC 184
10.6.3 Stopping UOC 184

Chapter 11 185

VNF Manager 185
11.1 Interaction modes 185
11.1.1 Indirect VNF Manager 185
11.1.2 Direct VNF Manager 186
11.1.3 Direct VNF Manager (Proxy) 186
11.2 VNF Manager 187
11.3 NFV Director - VNF Manager Interactions 187
11.3.1 VNF Manager Protocol Adapter: from VNFM to NFVD 187
11.3.2 VNF Manager Plugin: from NFVD to VNFM 190
11.4 VNF Manager Operations 190
11.4.1 Deploy and Register a VNF Manager 191
11.4.2 Instantiate a VNF through a VNF Manager 193
11.4.3 Delete a VNF through a VNF Manager 194

Chapter 12 195

State Propagation 195
12.1 State propagation functionality 195
12.2 Flow 195
12.3 State propagation process 196
12.3.1 Site Scope alarms 196
12.3.2 NFVD database and Fulfillment configurations 198
12.3.3 Parent hierarchy configuration 199
12.3.4 VNFM alarms 200
12.3.5 Key attributes in VNFM alarms 201

Chapter 13 202

Self Monitoring 202
13.1 Components managed 202
13.2 Monitoring process 202
13.2.1 Site scope templates 203
13.2.2 Modelling for self-monitoring 203
13.2.4 Triggering self-monitoring 207
13.2.5 NFVD configuration for Self-monitoring 207

13.2.6 Stopping self-monitoring 207

Chapter 14 208

Extending Monitors using Site Scope....208

14.1 MultiSiteScope support: 208
14.2 Tagging of monitors in SiteScope 208
14.3 Accessing SiteScope 209
14.4 Overview of SiteScope dashboard 209
14.5 Analyzing data in SiteScope dashboard 211
14.6 Dashboard - status and availability levels....212
14.7 SiteScope templates and monitoring 213
14.7.1 Creating custom templates 219
14.8 Alerts section....223
14.9 Configuring an alert....223
14.9.1 Prerequisites 223
14.9.2 Creating/copying an alert 223
14.10 SNMP preferences....224
14.11 Sending SiteScope Alerts 227
14.12 User management preferences 229
14.13 Managing certificates 230

Chapter 15 231

Extending Action capabilities using UCA-Automation....231

15.1 Correlation and autonomous process 231
15.1.1 Alarm enrichment....232
15.1.2 Autonomous action 236
15.1.3 Status of Action and Reporting 237

Chapter 16 239

Discovery and Reconciliation....239

16.1 Capacity Management 239
16.2 Exporting Topology and Metrics Information 247
16.2.1 Exporting Topology Information....247
16.2.2 Exporting Metrics Information 249

Appendix A....252

Cloud System 8 Operations 252

A.1 Login 252
A.2 Overview 252
A.3 Virtual machines (Instances)....253
A.4 Images 254
A.5 Networks 254

Appendix B....255

Relationships in NFV Director V3.0 255

Tables

Table 1 Artifacts in NFV Director V3.0 39
Table 2 Relationships in NFV Director V3.0 40
Table 3 Definitions 49
Table 4 Templates 52
Table 5 Instances 54
Table 6 Automation processes 55
Table 7 VNFs actions 56
Table 8 Inventory actions 59
Table 9 Monitoring actions 62
Table 10 Network_Service actions 62
Table 11 NS actions 63
Table 12 Virtual Core actions 63
Table 13 Virtual Memory actions 63
Table 14 VNF Manager actions 64
Table 15 Alarm field description 69
Table 16 VIMs status 76
Table 17 Version for COMPUTE Artifacts 116
Table 18 Version for NETWORK Artifacts 117
Table 19 Operations in CS8 supported by NFV Director 118
Table 20 Workflows supported 124
Table 21 SiteScope template objects 214
Table 22 SNMP User Interface Elements 225
Table 23 SNMP Preference Settings 226
Table 24 SNMP Preferences Advanced Settings 227
Table 25 User Management Preferences 230
Table 26 Alarm attributes for Autonomous action 237
Table 27 CSV File content 250

Figures

Figure 1 XML data model architecture 19
Figure 2 Example of artifact relationship 20
Figure 3 Definition, Instances and Templates relation 21
Figure 4 General structure for an artifact DEFINITION 21
Figure 5 General structure for an relationship DEFINITION 22
Figure 6 Example of artifact relationship definition 22
Figure 7 Instance generation process from template 23
Figure 8 Example of artifact relationship template 23
Figure 9 Example of artifact relationship instance 24
Figure 10 Example of artifact relationship instance 24
Figure 11 VNF Model 25
Figure 12 Resources Model 26
Figure 13 Monitoring Model 27
Figure 14 Self Monitoring Model 30
Figure 15 Open Loop Action 30
Figure 16 Open Loop Interaction for Approve/Disapprove 30
Figure 17 Overview Model 32
Figure 18 Overview Model: Virtual Resources 33
Figure 19 Overview Model: Virtual Infrastructure 34
Figure 20 Overview Model: Physical Resources 35
Figure 21 Datacenter Artifacts 36
Figure 22 Policy Artifacts 36
Figure 23 Simple VNF 41
Figure 24 Application Server 41
Figure 25 Application Server with policies and monitors 42
Figure 26 SoapUI 42
Figure 27 SoapUI: NFVD Model Rest API 43
Figure 28 SoapUI: Get all definitions 44
Figure 29 Northbound Architecture 46
Figure 30 NBI: Start VM 56
Figure 31 NBI: Stop VM 57
Figure 32 NBI: Scale-In 60
Figure 33 NBI: Scale-Out 61
Figure 34 Scale up Virtual Core 63
Figure 35 Notification flow diagram 67
Figure 36 Artifact Status 68
Figure 37 SoapUI 70
Figure 38 SoapUI: NFVD Model Rest API 71
Figure 39 SoapUI: Get all definitions 71
Figure 40 SoapUI: NFVD Automation 73
Figure 41 SoapUI: Activate VNF example 74
Figure 42 SoapUI: Activate VNF example (II) 74
Figure 43 HP NFV Director Detailed Modules 75
Figure 44 Southbound Interface - Detailed 76
Figure 45 HPSA Solution Container ECP command template 118
Figure 46 Example server template 119
Figure 47 Example: Create Server Workflow 120
Figure 48 CloudSystem Firefox Rest Client 124
Figure 49 CloudSystem Rest Client Request Header 125
Figure 50 CloudSystem Create Server operation 125
Figure 51 CloudSystem Edit Server operation 126
Figure 52 CloudSystem Query Server operation 126
Figure 53 CloudSystem Delete Server operation 127

Figure 54 CloudSystem Start Server operation 127
Figure 55 CloudSystem Stop Server operation....128
Figure 56 CloudSystem Query Image operation 128
Figure 57 CloudSystem Create Flavour operation....129
Figure 58 CloudSystem Query Flavour operation....129
Figure 59 CloudSystem Query Flavour by Parameters operation 130
Figure 60 CloudSystem Delete Flavour operation....130
Figure 61 CloudSystem Create Network operation 131
Figure 62 CloudSystem Edit Network operation 131
Figure 63 CloudSystem Query Network by ID operation 132
Figure 64 CloudSystem Query Network by Parameters operation 132
Figure 65 CloudSystem Delete Network operation 132
Figure 66 CloudSystem Create Subnet operation 133
Figure 67 CloudSystem Edit Subnet operation....133
Figure 68 CloudSystem Query Subnet operation 134
Figure 69 Creating VNF process 135
Figure 70 VNF Template example....136
Figure 71 Procedure for Start virtual machine 140
Figure 72 Virtual machine lifecycle....141
Figure 73 VNF lifecycle....141
Figure 74 Template example with range policy 142
Figure 75 Instance result of template example with range policy....142
Figure 76 Instance result of Template example with assign policy 143
Figure 77 Policy Assignment Relationship hierarchical tree 144
Figure 78 Policy: Apply Relationship 147
Figure 79 Policy Assignment process 147
Figure 80 Policy Assignment Flow 148
Figure 81 Policy Assignment Group 148
Figure 82 Query Target Policies....149
Figure 83 Query artifact for assignment target policy 149
Figure 84 Query Resources for assignment target policy 149
Figure 85 Assign Resource Artifact....150
Figure 86 Query resources in Assignment Group policy....150
Figure 87 Query Target policy and Group policy for artifact and resource 150
Figure 88 Query Assignment Target Policy for artifact children 150
Figure 89 Query Assignment Target Policy for Resource Children....151
Figure 90 Assign artifact to Resource....151
Figure 91 Activation flow 152
Figure 92 Activation: get vDC....153
Figure 93 Activation: get Virtual Memory and Virtual Disk....153
Figure 94 Activation: get Image ID and network ID 154
Figure 95 Activation: update Status 155
Figure 96 Deactivate flow 156
Figure 97 VNF Scales operations 157
Figure 98 VNF Scale In....158
Figure 99 Scale In: get associated template....159
Figure 100 Scale In: get all template children 159
Figure 101 Scale In: Perform operation recursively 159
Figure 102 Scale In: Create new instance from template....160
Figure 103 Scale In: Check policy 160
Figure 104 Scale In Example: Artifact Template Tree 160
Figure 105 Scale In Example: Create Virtual Core Instance....161
Figure 106 Scale In Example: Test Verification....161
Figure 107 Scale Out 162
Figure 108 Scale out: query associated template 163
Figure 109 Scale out: query template children 163
Figure 110 Scale out: apply scale out recursively 163
Figure 111 Scale out: create instances from template 164
Figure 112 Scale out: operation completed 164
Figure 113 Scale out Example: Artifact template tree 164

Figure 114 Scale out Example: Create virtual core 165
Figure 115 Scale out Example: Test verification 165
Figure 116 Scale up flow 167
Figure 117 VNF tree for scale up 168
Figure 118 Memory amount instance 168
Figure 119 Scale up: resize flavour 168
Figure 120 VNF tree for scale down 170
Figure 121 Memory amount instance 171
Figure 122 Scale down: resize flavour 171
Figure 123 Two VNF connected using one network 172
Figure 124 Two VNF connected to the same network 173
Figure 125 Two VNF connected to the same network: Virtual Port 173
Figure 126 Two VNF connected to the same network 174
Figure 127 Two VNF connected to the same network - VIM 175
Figure 128 Two VNF connected through a third VM 175
Figure 129 Two VNF connected through a third VM - VIM 176
Figure 130 Two VNF connected with an external network through VM - VIM 177
Figure 131 Two VNF connected with an external network through VM - VIM 178
Figure 132 VNF Manager 185
Figure 133 VNF Indirect Manager 185
Figure 134 VNF Direct Manager 186
Figure 135 VNF Direct Manager (Proxy) 186
Figure 136 VNF Manager interaction from RESTClient 187
Figure 137 Template for SCALE_OUT_VNF 191
Figure 138 Solution Container Search 191
Figure 139 Solution Container: Deploy and register VNF Manager 192
Figure 140 GPM Task 192
Figure 141 Manual tasks 193
Figure 142 Instantiate VNF 193
Figure 143 VNF DescriptorParameterGroup 193
Figure 144 VNF Selection 194
Figure 145 Delete VNF 194
Figure 146 Propagation modes 195
Figure 147 State propagation flow 196
Figure 148 Self monitoring components 202
Figure 149 Self monitoring Model 204
Figure 150 VNF : NFVD artifact 205
Figure 151 MONITOR : SELF artifact 205
Figure 152 MONITOR : CUSTOM artifact 206
Figure 153 MONITOR : GENERIC artifact 206
Figure 154 MONITOR_ARGUMENTS: GENERIC artifact 206
Figure 155 SiteScope dashboard 210
Figure 156 SiteScope Dashboard context buttons 211
Figure 157 SiteScope Dashboard: view monitor status 211
Figure 158 SiteScope Dashboard: view configured and triggered alerts 212
Figure 159 SiteScope Dashboard: Acknowledge monitors 212
Figure 160 SiteScope Dashboard: view monitor history 212
Figure 161 SiteScope Dashboard: status and availability levels 213
Figure 162 SiteScope: sample template 215
Figure 163 SiteScope: monitor context 215
Figure 164 SiteScope: Create custom templates 219
Figure 165 SiteScope: new template 220
Figure 166 SiteScope: new group 220
Figure 167 SiteScope: new Monitor 220
Figure 168 SiteScope: new Variable 221
Figure 169 SiteScope: new Variable details 221
Figure 170 SiteScope: enter variable to associate with template 222
Figure 171 SiteScope: hierarchy of custom template 222
Figure 172 SiteScope: send alerts always 228
Figure 173 SiteScope: send alert once 229

Figure 174 Correlation and autonomous action process diagram 231
Figure 175 Snapshot of Status of action 237
Figure 176 Snapshot of Reporting of action 238
Figure 177 Capacity Management Design Diagram 240
Figure 178 VIM and Authentication Artifact Instances 240
Figure 179 Discovered Artifacts 247
Figure 180 CloudSystem Login Portal 252
Figure 181 CloudSystem overview 253
Figure 182 CloudSystem VM Instances 253
Figure 183 CloudSystem Images list 254
Figure 184 CloudSystem List of available networks and subnets 254
Figure 185 CloudSystem Subnet 254

Preface

In this Guide

This guide describes the NFV Director for users of the solutions that are based on the product. It is a general overview of the NFV Director.

Operators and system administrators should read the *User Guide for Advanced Users* too. It is focused on the NFV Director Processes in depth detail.

Chapter 1 is a general overview of the NFV Director for the audience of this guide.

Chapter 2 provides a detail view of the data modeling of the NFV Director and how the relationships are defined between the various model objects.

Chapter 3 describes the North Bound Interfaces available in order to send a request for creation of Virtual Machines (VM) and Virtual Network Functions (VNF), Scale in and out operations.

Chapter 4 describes the South Bound Interfaces implemented to perform the NFV Director operations on VIMS and Hypervisors.

Chapter 5, 6, 7 and 8 describe the Virtual Network Functions; Policies applied on the VNFs, VNFs activation and VNF Scaling operations.

Chapter 9 describes network connections in NFV Director and Network Services.

Chapter 10 describes the administration of various NFV Director Components in the solution.

Chapter 11 describes the VNF monitoring aspects of the NFV Directory. It involves configuring and administrating the HP SiteScope product to achieve the VNF monitoring.

Chapter 12 describes the NFV Director process on the alerts triggered by the KPI breach on the VMs and VNFs, including the correlation and the action taken to remedy the root cause of the alert.

Chapter 13, 14 and 15 describe the NFV Director process on the alerts triggered by the KPI breach on the VMs and VNFs, including the correlation and the action taken to remedy the root cause of the alert.

Chapter 15 describes discovery and reconciliation modules, both of them included in the capacity management component.

Note
Read this document before installing or using this software.

Audience

It is recommended for any user of the NFV Director to begin with this document before proceeding with installation or any other aspects of the solution.

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.

Associated Documents

The following documents contain useful reference information:

References

- *HP NFV Director User Guide - Overview.*
- *HP NFV Director API - Reference guide*
- *HP NFV Director WorkFlows - Reference guide*
- *HP Service Activator v6.2 User's and Administrator's Guide*
- *HPSA Extension Pack SOSA v6.0 - Developer Reference*
- *HP NFV Director Release Notes*
- *HP UCA for Event Based Correlation V3.1 - User Interface Guide V1.0 ,*
- *HP UCA for Event Based Correlation V3.1 - Value Pack Development Guide V1.0*
- *HP UCA for Event Based Correlation V3.1 - Topology Extension V1.0*
- *HP UCA Automation 1.2 Integrator's Guide 1.0*
- *OSS Open Mediation Functional Specification*
- *Site Scope MonitorReference*
- *Using Site Scope*
- *SiteScope Integration Best Practices*
- *SiteScope API Reference*

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 on demand.
- 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.

For more information about NFV Director guidelines, refer to the User Guide – Overview document.

Chapter 2

NFV Director Data Model

2.1 NFV Director modeling philosophy

The NFV Director does not follow the regular database modeling rules. Instead, it uses artifacts that can be related between them using relationships.

The NFV director model is an XML based model that changes the current way of working with HPSA.

Each operation (GUI, Northbound, or automation) is translated into XML data after it is stored in the corresponding persistent technology.

The model is not tied to any specific persistent technology.

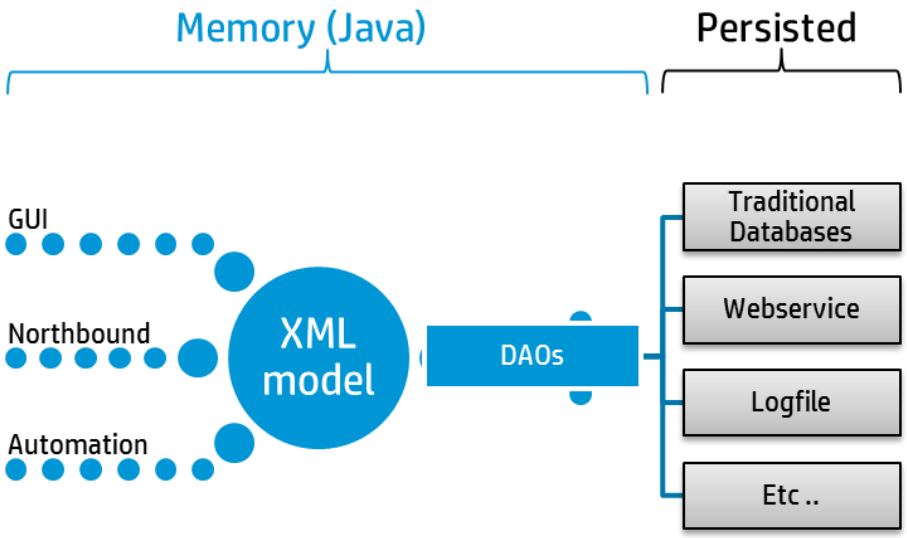


Figure 1 XML data model architecture

The NFV Director uses two master entities to model most of the systems.

2.1.1 Artifact

An artifact is an entity of any type that can have any number of attributes. Also, an artifact has a status.

2.1.2 Relationship

A relationship is a parent-child relationship from one artifact to another. It can be of different types and contain any number of attributes.

A relationship has a direction: an artifact is the parent and the other artifact is the child.

Like artifacts, a relationship has a type, several attributes and a status.

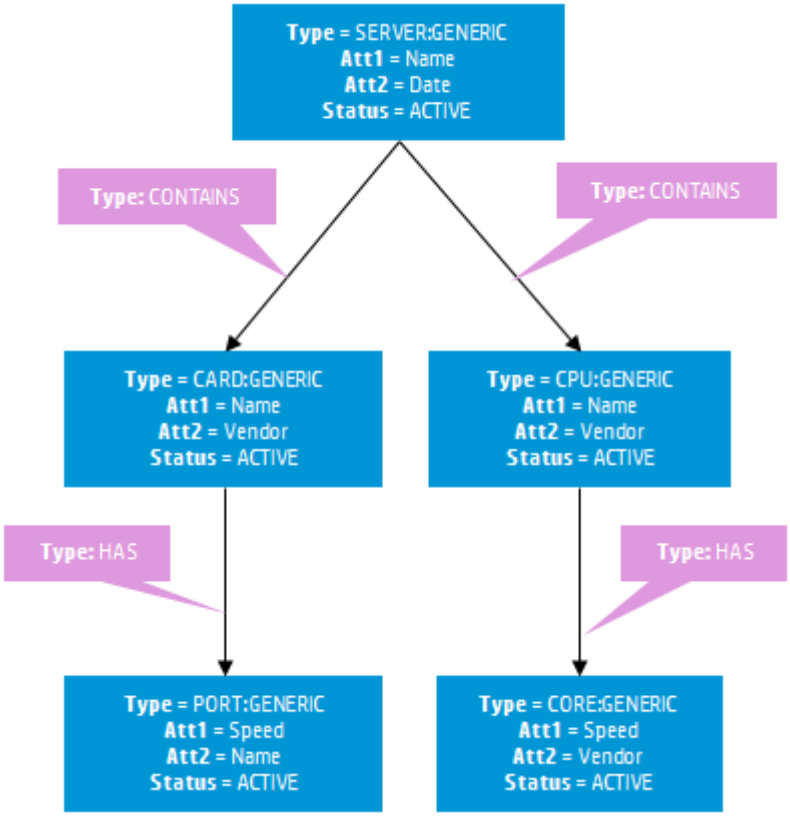


Figure 2 Example of artifact relationship

2.2 Flavors of artifacts and relationships

The three different flavors of artifacts and relationships are the following:

- Definitions
- Template
- Instance

We can resume the relation about definitions, templates and instances with next figure:

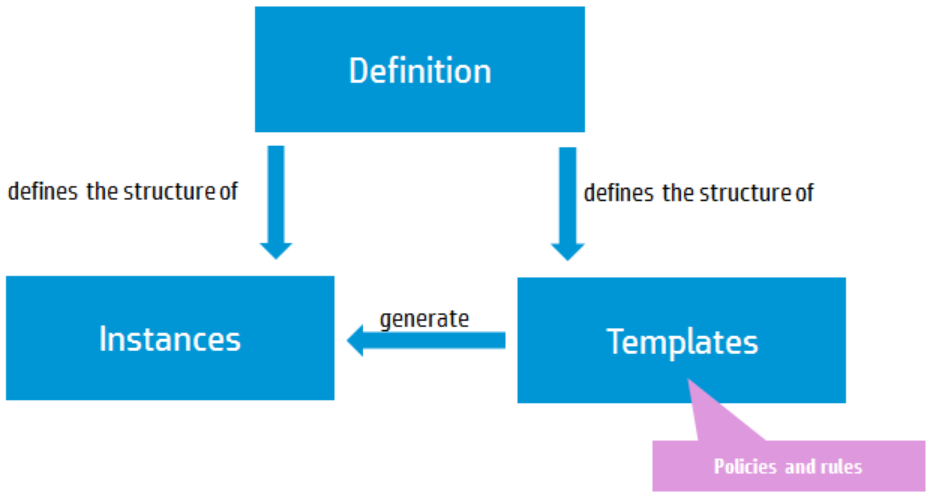


Figure 3 Definition, Instances and Templates relation

2.2.1 Definition

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 DEFINITION table also defines the artifacts that can be parents of different types of relationships to other artifacts and the possible statuses of an artifact too.

Definition Type	Family		
	Category		
	Group		
	Type		
	Subtype		
	Version		
Attributes	Category1	Att1	
		Att2	
		Category2	Att3
			Att4
	Category3	Att5	
		Att6	
Statuses	STATUS1		
	STATUS2		
	STATUS3		

Figure 4 General structure for an artifact DEFINITION

With a relationship DEFINITION we specify that a relation between two kinds of artifacts is possible, and the following values are set:

- Parent Artifact Definition
- Child Artifact Definition
- Relationship type

Also, a relationship

- may have categories and attributes
- one or more statuses.

Parent Definition	Family
	Category
	Group
	Type
	Subtype
	Version
Child Definition	Family
	Category
	Group
	Type
	Subtype
	Version
Relationship Type	TYPE1
Statuses	STATUS1 STATUS2

Figure 5 General structure for an relationship DEFINITION

The DEFINITION tables define what is possible.

A set of definitions are shipped with the product. However, you can easily add new artifacts and relationships, or modify the existing ones.

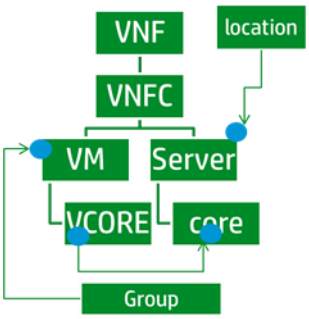


Figure 6 Example of artifact relationship definition

2.2.2 Template

When we talk about artifact templates, we define an artifact template as a special kind of artifact, almost the same than an artifact instance. The difference is an artifact template can generate artifact instances.

Also, an artifact template may have policies and rules associated, that helps to fix the behavior when generating Artifact Instances.

When an artifact instance is generated from an artifact template, the artifact instance will have the ID of the artifact template in its TemplateID attribute.

In relationship case, a relationship template represents the relationship between two artifact templates. Almost the same than a Relationship Instance.

When generating the Artifact Instances from two Artifact Templates related between them using a Relationship Template, a Relationship Instance is also created between the two Artifact Instances created.

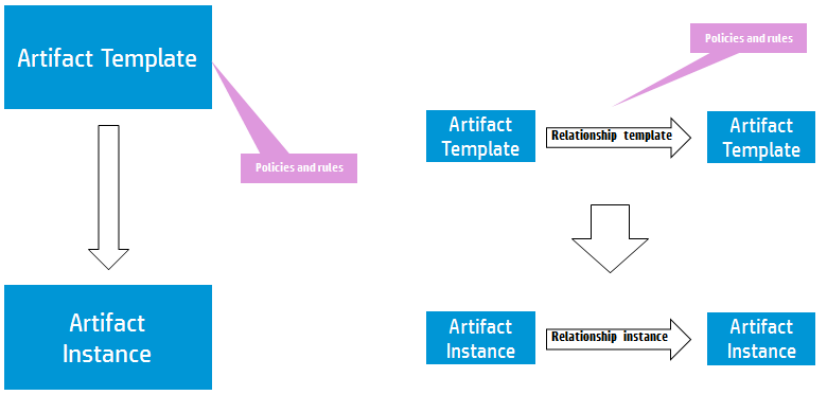


Figure 7 Instance generation process from template

An example of artifact relationship template is shown in next figure:

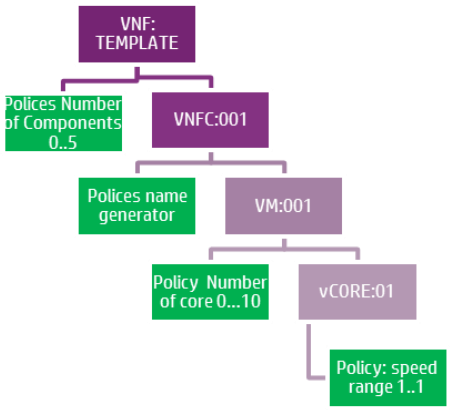


Figure 8 Example of artifact relationship template

2.2.3 Instance

Each definition can have numerous instances with different or equal values for each one of its attributes. The instance information contains the inventory of all provisioned services and resources.

An artifact instance is always based on an Artifact Definition, it is a "working" entity. There can be multiple Artifact Instances of each definition, with different or equal values in its attributes, but always with a unique ID obtained from its TemplateID.

Definition Type	Family	"VIRTUAL_MACHINE"	
	Category	"GENERIC"	
	Group		
	Type		
	Subtype		
	Version		
Attributes	INFO	Name	"quijote"
		Description	"frontend"
	VIM	Name	"ducati"
		URL	"http://"
Statuses	ACTIVE		
ID	29384913		

Figure 9 Example of artifact relationship instance

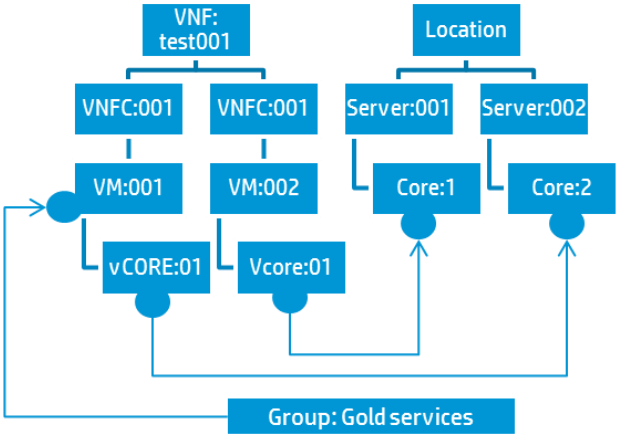


Figure 10 Example of artifact relationship instance

2.3 Out-of-the-box models

The product provides a set of out-of-the-box models, which allow you (for example) to model Virtual Network Functions, Network services, and Datacenters.

2.3.1 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.

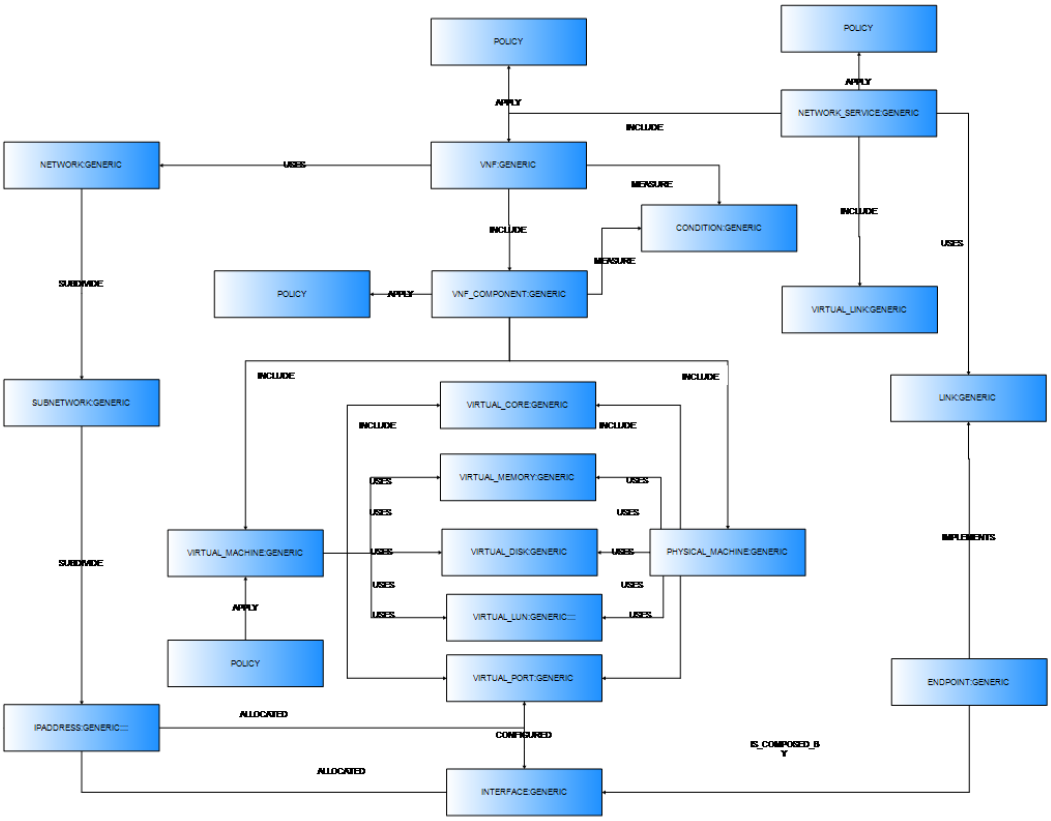


Figure 11 VNF Model

2.3.2 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.

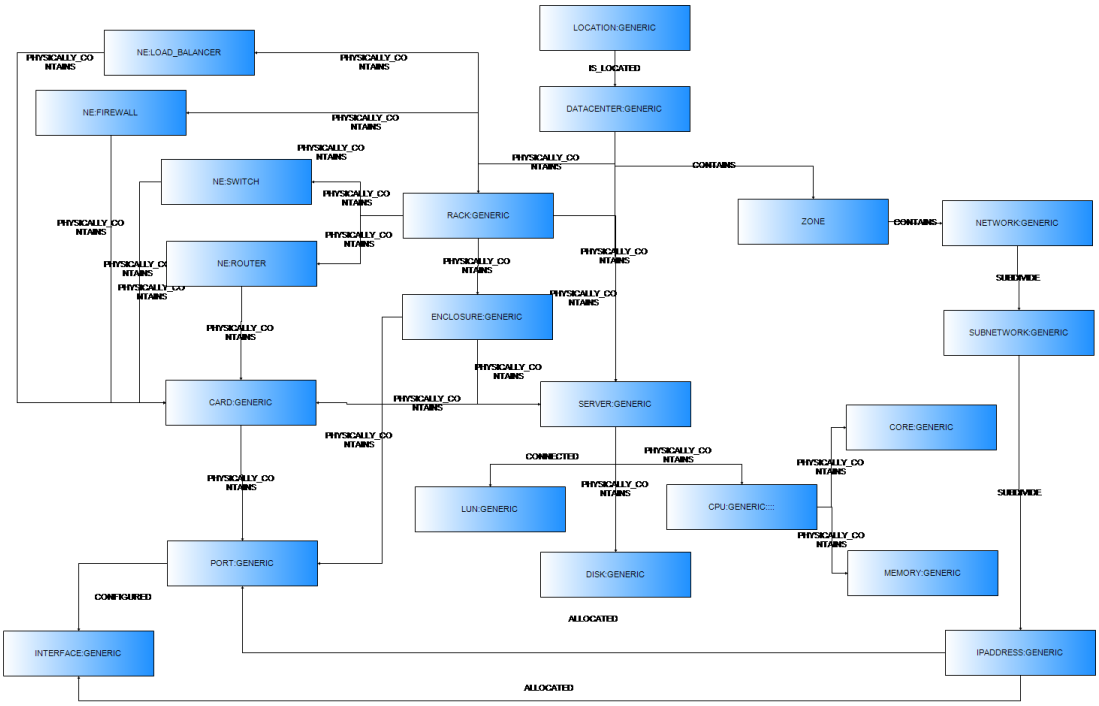


Figure 12 Resources Model

2.3.3 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.

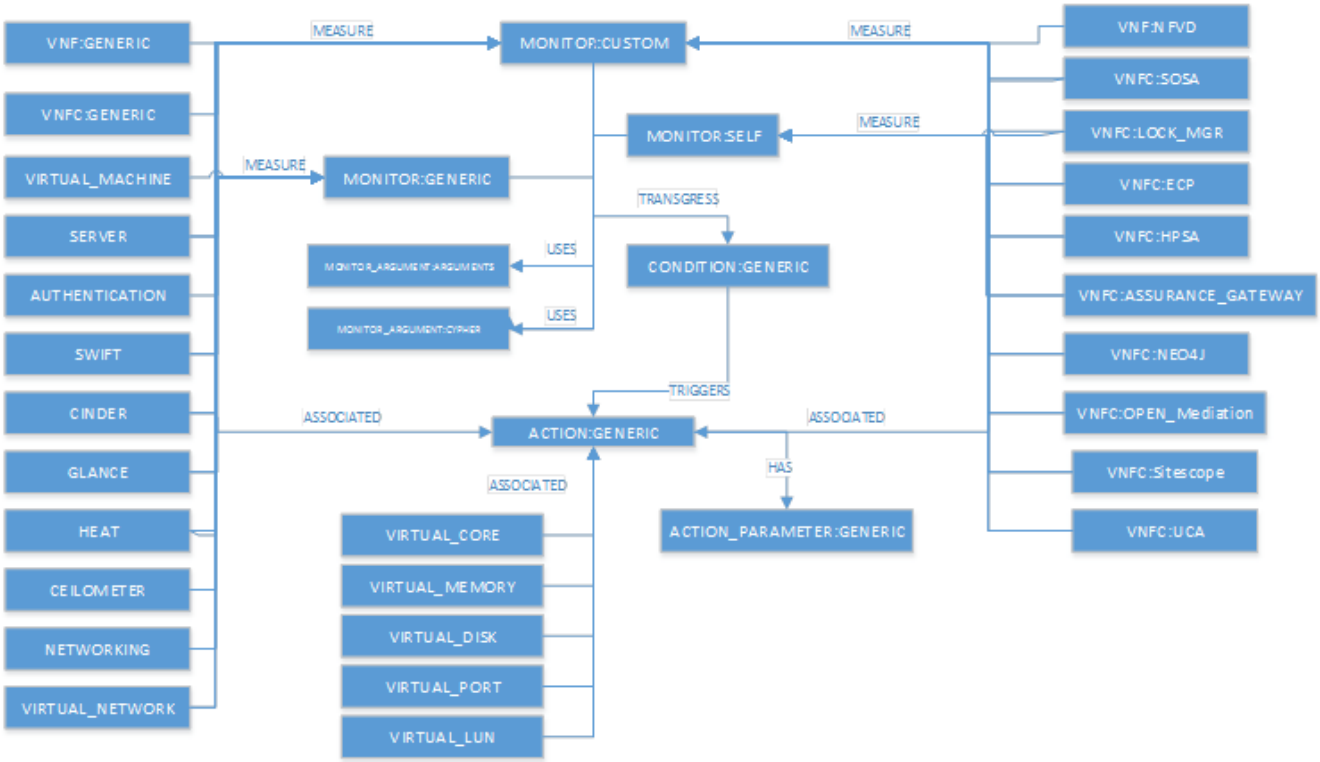


Figure 13 Monitoring Model

NFVD supports out of the box monitors for the below components:

a. VIRTUAL_MACHINE – CPU, Memory,DiskRead,DiskWrite,NetworkRx, NetworkTx
b. SERVER – CPU, Memory,DiskRead,DiskWrite,NetworkRx, NetworkTx
c. AUTHENTICATION – Process, Logs
d. SWIFT – Process
e. CINDER – Process
f. GLANCE – Process
g. HEAT – Process
h. CEILOMETER – Process
i. NETWORKING – Process
j. VIRTUAL_NETWORK - Network

All of the components listed above will have a MONITOR:GENERIC related to them with relationship type “MEASURE”.

Users can create custom monitors MONITOR:CUSTOM for any of the components and relate it to the component with relationship type “MEASURE”.

MONITOR:SELF can be associated to the below VNFC only:

a. VNFC:SOSA – Process, Logs
b. VNFC:LOCK_MGR – Process, Logs
c. VNFC:ECP – Process, Logs
d. VNFC:HPSA – Process, Logs
e. VNFC:ASSURANCE_GATEWAY – Process, Logs
f. VNFC:NEO4J – Process, Logs

g. VNFC:OPEN_MEDIATION – Process, Logs
h. VNFC:SITESCOPE – Process, Logs
i. VNFC:UCA – Process, Logs
j. VNFC:ORACLE – Process, Logs
k. VNFC:POSTGRES – Process, Logs

MONITOR:SELF needs to be associated to the VNFC only in cases where user requires to trigger action based on threshold breach.

Openstack services like Compute, Authentication (i.e Keystone), Swift, Glance, Heat, Cinder, Networking, Ceilometer can be monitored by associating it with MONITOR:GENERIC artifact with relationship "MEASURE". SOSA soap calls need to be executed to deploy and start the monitor. Please check Appendix A in Integration guide to execute SOSA calls for deploy/start/stop/undeploy of monitors.

MONITOR can be associated to MONITOR_ARGUMENTS. MONITOR_ARGUMENTS come in two flavours:

a. MONITOR_ARGUMENT:ARGUMENT - it contains name value pairs, name has to map to mandatory variables required by template in sitescope
b. MONITOR_ARGUMENT: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 so that it matches to the mandatory varialbes in sitescope templates. If alias does not match to the mandatory varaibles used in sitescope template then those values will be simply ignored. 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 charcters, i.e. below mentioned characters needs to be escaped.
   " "
   ' '
   < <
   > >
   & &

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

MONITOR will be associated to CONDITION with relationship type "TRANSGRESS".
CONDITION can be of three types:

a. ERROR
b. WARNING
c. GOOD

CONDITION has an attribute

a. "Expression" under the category "GENERAL" where the expression needs to be specified. Expression contains counter and threshold value
   ex: "cpu_usage_average > 90" . where cpu_usage_average is the counter variable and 90 is the threshold value. In sitescope template the counter value will map to the variable like "cpu_usage_average_error" , "cpu_usage_average_good", "cpu_usage_average_warning"(i.e. counter_ConditionType)
b. "TYPE" under the category "GENERAL" which can have value "ERROR,"WARNING", "GOOD"
   Below is the table showing expression that can be used for various out of the box monitors.

	ERROR	WARNING	GOOD
CPU	cpu_usage_average > THRESHOLD	cpu_usage_average > THRESHOLD	cpu_usage_average < THRESHOLD
DiskRead	disk_read_requests >	disk_read_requests >	disk_read_requests <

	THRESHOLD	THRESHOLD	THRESHOLD
DiskWrite	disk_write_requests > THRESHOLD	disk_write_requests > THRESHOLD	disk_write_requests < THRESHOLD
Memory	memory_usage_average > THRESHOLD	memory_usage_average > THRESHOLD	memory_usage_average < THRESHOLD
NetworkRx	network_bytes_received > THRESHOLD	network_bytes_received > THRESHOLD	network_bytes_received > THRESHOLD
NetworkTx	network_bytes_transmitted > THRESHOLD	network_bytes_transmitted > THRESHOLD	network_bytes_transmitted < THRESHOLD
Process	status == 1	Not Applicable	status == 0
Logs	matches > THRESHOLD	Not Applicable	matches < THRESHOLD
Network	network_bytes_in_received > THRESHOLD	network_bytes_in_received > THRESHOLD	network_bytes_in_received < THRESHOLD
	network_bytes_out_transmitted > THRESHOLD	network_bytes_out_transmitted > THRESHOLD	network_bytes_out_transmitted < THRESHOLD
	network_packets_dropped_by_rate_limit > THRESHOLD	network_packets_dropped_by_rate_limit > THRESHOLD	network_packets_dropped_by_rate_limit < THRESHOLD
	network_packets_in_dropped > THRESHOLD	network_packets_in_dropped > THRESHOLD	network_packets_in_dropped < THRESHOLD
	network_packets_in_fault > THRESHOLD	network_packets_in_fault > THRESHOLD	network_packets_in_fault < THRESHOLD
	network_packets_in_received > THRESHOLD	network_packets_in_received > THRESHOLD	network_packets_in_received < THRESHOLD
	network_packets_out_dropped > THREHOLD	network_packets_out_dropped > THRESHOLD	network_packets_out_dropped < THRESHOLD
	network_packets_out_fault > THRESHOLD	network_packets_out_fault > THRESHOLD	network_packets_out_fault < THRESHOLD
	network_packets_out_transmitted > THRESHOLD	network_packets_out_transmitted > THRESHOLD	network_packets_out_transmitted < THRESHOLD

CONDITION will be associated to ACTION with relationship type “TRIGGERS”

a. GENERAL.Tpe: can have values SCALE_UP, SCALE_DOWN, SCALE_IN, SCALE_OUT, SCRIPT
b. GENERAL.Path: If action type is chosen as script , here user needs to specify the location of the script to be invoked.

GENERAL.Operation_Mode: can be set to CLOSED_LOOP or OPEN_LOOP, by default it is set to CLOSED_LOOP. If user configures OPEN_LOOP, then the action would require approval by user in UCA_AUTOMATION console

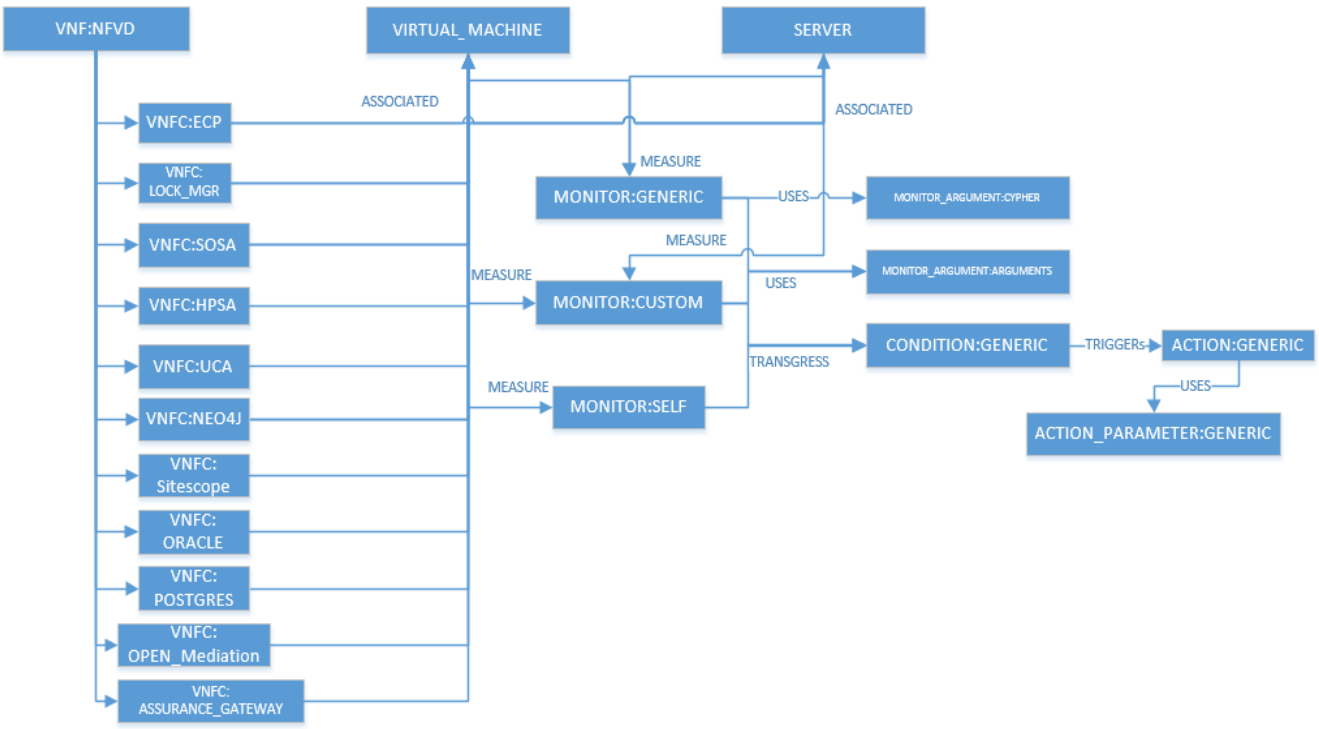


Figure 14 Self Monitoring Model

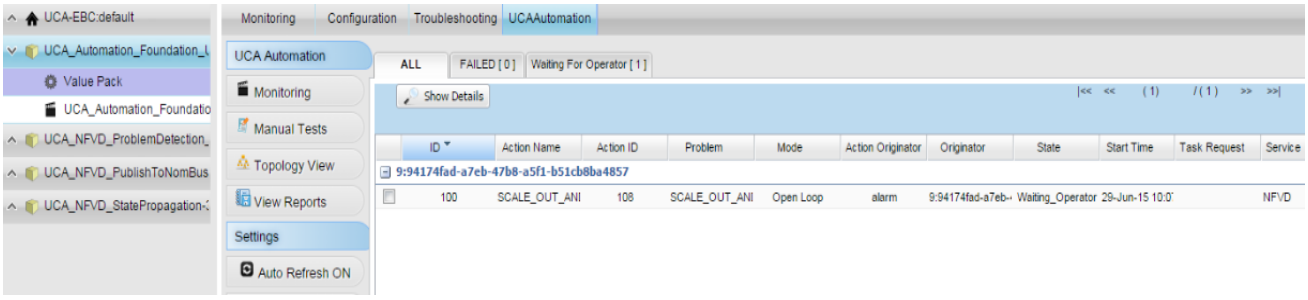


Figure 15 Open Loop Action

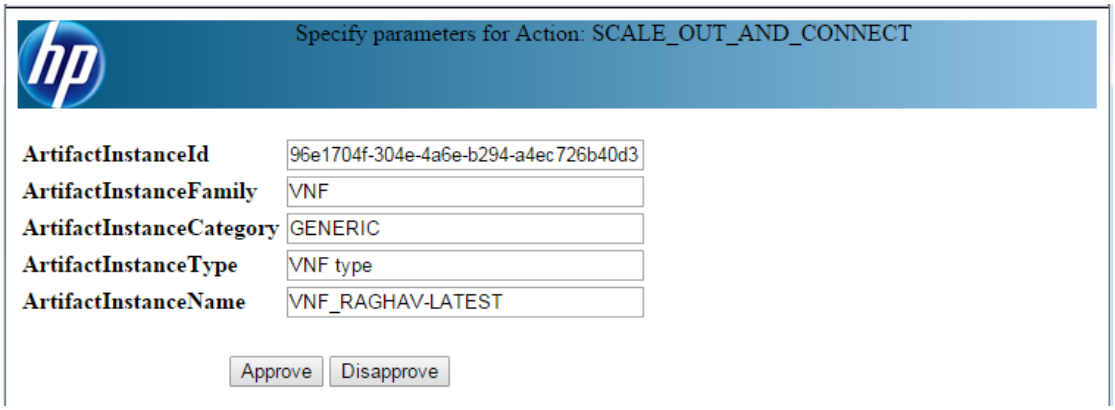


Figure 16 Open Loop Interaction for Approve/Disapprove

2.3.4 Overview model

A high level overview model in NFV Director is described in the following illustration.

It shows interaction between:

- Virtual Resources
- Virtual Infrastructure
- Physical Resources

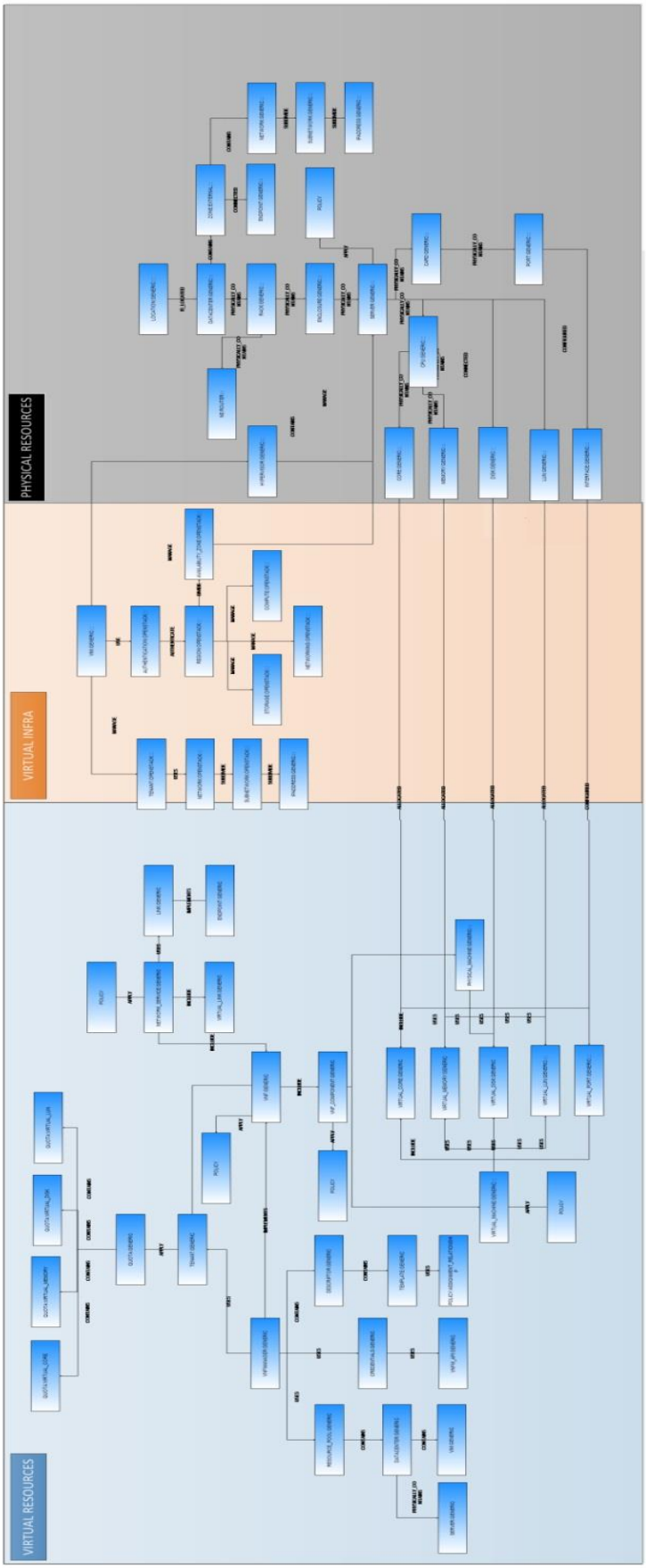


Figure 17 Overview Model

2.3.4.1 Overview model: Virtual Resources

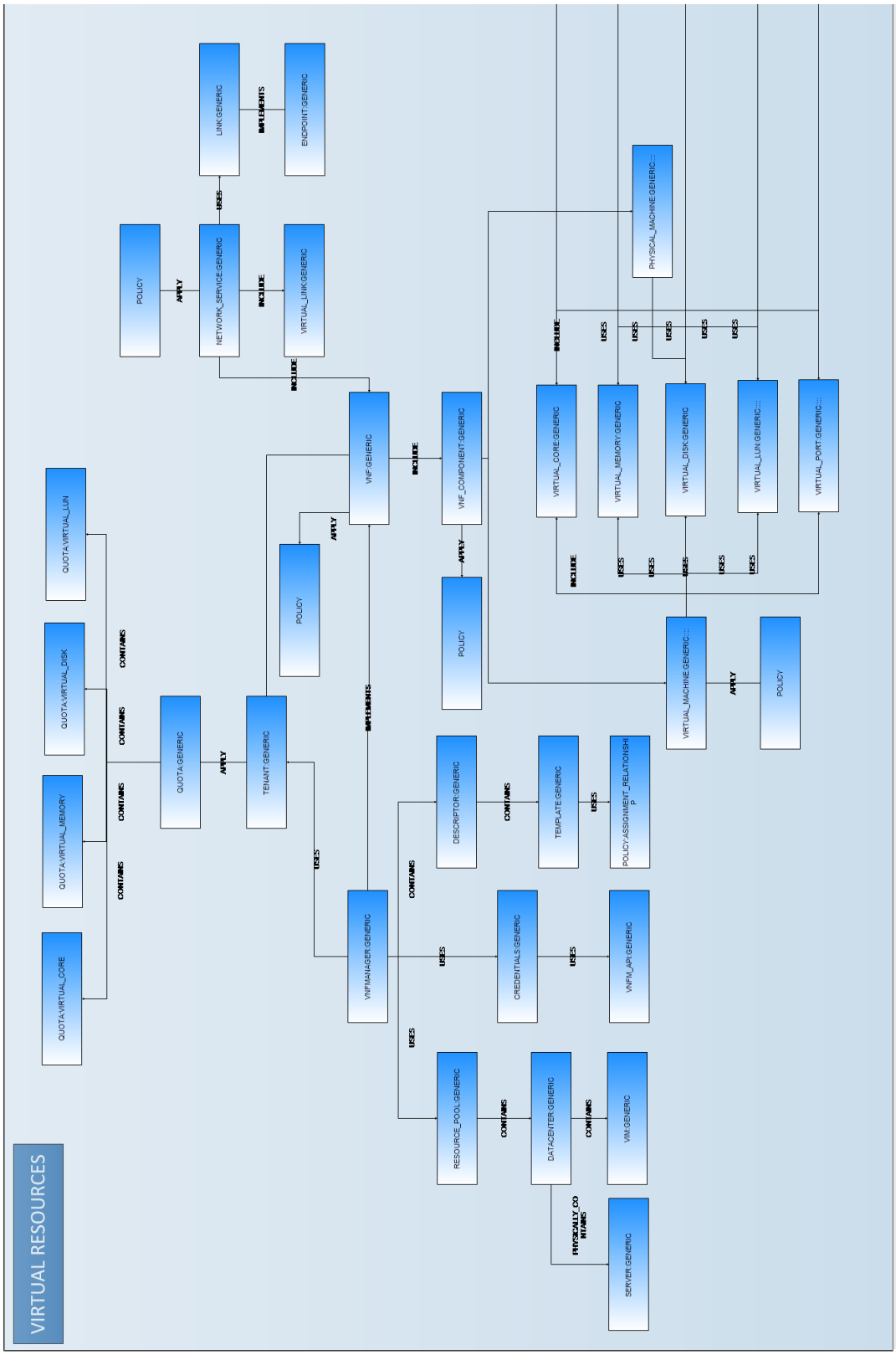


Figure 18 Overview Model: Virtual Resources

2.3.4.2 Overview model: Virtual Infrastructure

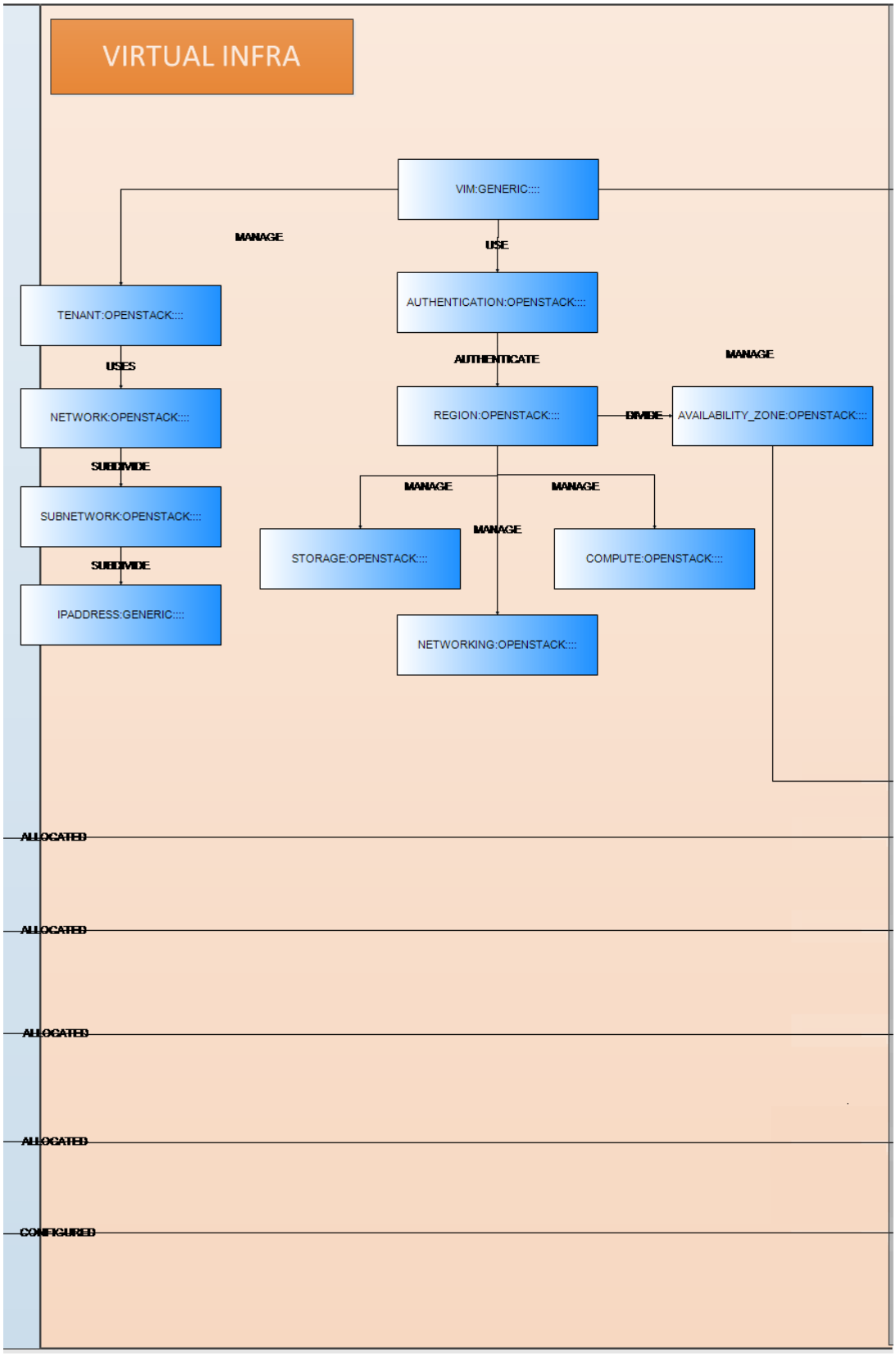


Figure 19 Overview Model: Virtual Infrastructure

2.3.4.3 Overview model: Physical Resources

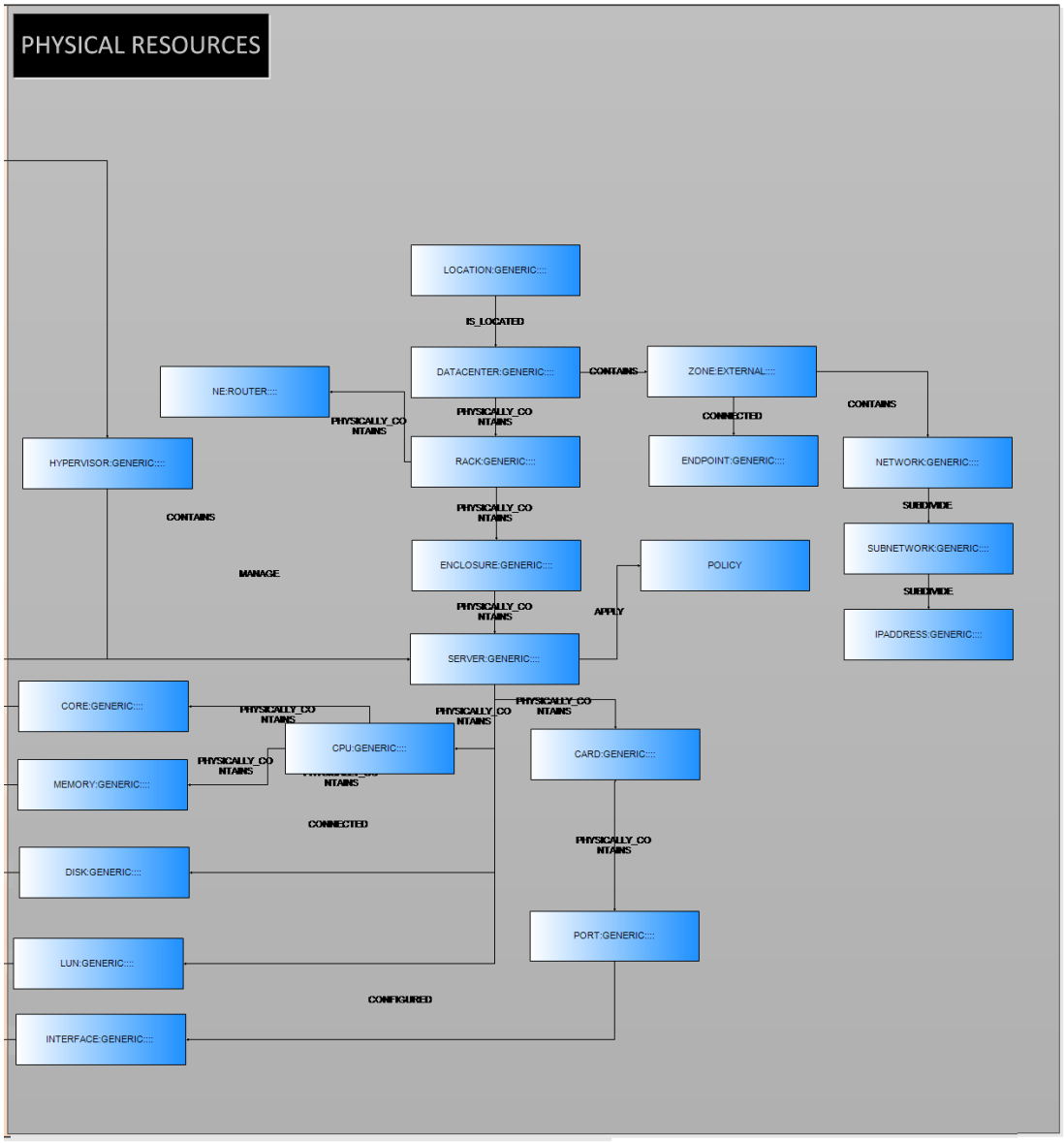


Figure 20 Overview Model: Physical Resources

2.3.5 Datacenter Artifacts

Next figure shows a high level representation of Datacenter Artifacts (virtual and physical datacenters).

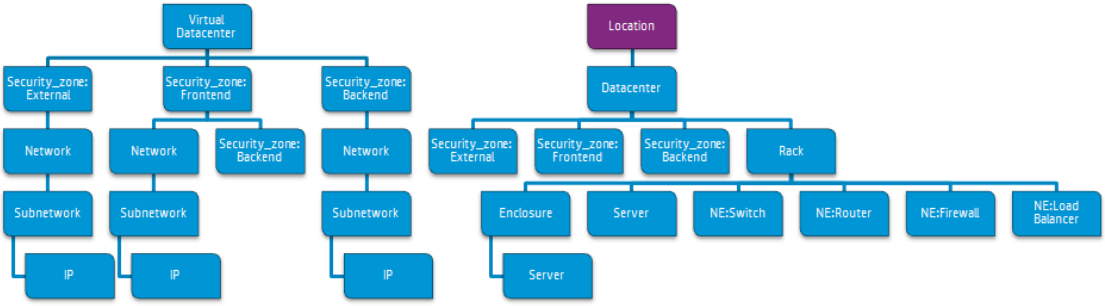


Figure 21 Datacenter Artifacts

2.3.6 Policy Artifacts

Policies define limits and restrictions. They can be applied to any object template.

Policies are applied to artifacts through relationships.

When creating an instance from a template policies will be:

- Created for each artifact and relationship defined on the template
- Applied to validate if the instance is valid within the defined rules

Next figure shows a high level representation of Policy Artifacts.

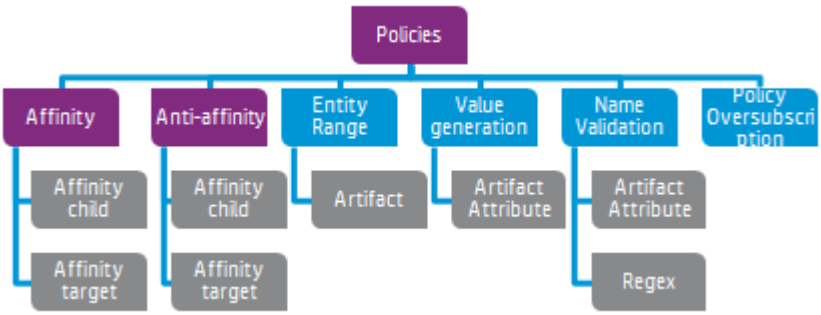


Figure 22 Policy Artifacts

2.4 Particularities

2.4.1 Relationships

As all relationships are stored in the same database table (when using a database as storage mechanism), create your queries by using type or ID (PK) as filters. The tables are partitioned using types. Hence, filtering using a type is similar to querying only a table with the data of that type and not all the relationships in the system.

2.5 Artifact and Relationship

2.5.1 Artifacts

The total account of artifacts in NFV Director V3.0 is 149 different artifacts.

The following table summarize artifacts group by their FAMILY, CATEGORY and GROUP.

	FAMILY	CATEGORY	GROUP	ARTIFACT (FAMILY:CATEGORY:GROUP)
1	ACTION	GENERIC		ACTION:GENERIC:
2	ACTION_PARAMETER	GENERIC		ACTION_PARAMETER:GENERIC:
3	AUTHENTICATION	OPENSTACK		AUTHENTICATION:OPENSTACK:
4	AVAILABILITY_ZONE	OPENSTACK		AVAILABILITY_ZONE:OPENSTACK:
5	BANDWIDTH	GENERIC		BANDWIDTH:GENERIC:
6	CARD	GENERIC		CARD:GENERIC:
7	CEILOMETER	OPENSTACK		CEILOMETER:OPENSTACK:
8	CINDER	OPENSTACK		CINDER:OPENSTACK:
9	COMPUTE	OPENSTACK		COMPUTE:OPENSTACK:
10	CONDITION	GENERIC		CONDITION:GENERIC:
11	CORE	GENERIC		CORE:GENERIC:
12	CPU	GENERIC		CPU:GENERIC:
13	CREDENTIALS	GENERIC		CREDENTIALS:GENERIC:
14	DATACENTER	GENERIC		DATACENTER:GENERIC:
15	DESCRIPTOR	GENERIC		DESCRIPTOR:GENERIC:
16	DISK	GENERIC		DISK:GENERIC:
17	ENCLOSURE	GENERIC		ENCLOSURE:GENERIC:
18	ENDPOINT	GENERIC		ENDPOINT:GENERIC:
19	EXECUTION_TASK	GENERIC		EXECUTION_TASK:GENERIC:
20	EXECUTION_TASK_LIST	GENERIC		EXECUTION_TASK_LIST:GENERIC:
21	FG_ENDPOINT	BIDIRECTIONAL	OPENSTACK	FG_ENDPOINT:BIDIRECTIONAL:OPENSTACK
22	FLAVOR	OPENSTACK		FLAVOR:OPENSTACK:
23	FORWARDING_GRAPH	GENERIC		FORWARDING_GRAPH:GENERIC:
24	GLANCE	OPENSTACK		GLANCE:OPENSTACK:
25	GROUP	GENERIC		GROUP:GENERIC:
26	HEAT	OPENSTACK		HEAT:OPENSTACK:
27	HOSTAGGREGATES	OPENSTACK		HOSTAGGREGATES:OPENSTACK:
28	HYPERVISOR	GENERIC		HYPERVISOR:GENERIC:
29	HYPERVISOR	KVM		HYPERVISOR:KVM:
30	HYPERVISOR	VCENTER		HYPERVISOR:VCENTER:
31	HYPERVISOR	VMWARE		HYPERVISOR:VMWARE:
32	IMAGE	GENERIC		IMAGE:GENERIC:
33	IMAGE	OPENSTACK		IMAGE:OPENSTACK:
34	IMAGE_STORAGE	OPENSTACK		IMAGE_STORAGE:OPENSTACK:
35	INTERFACE	GENERIC		INTERFACE:GENERIC:
36	IPADDRESS	FLOATING		IPADDRESS:FLOATING:
37	IPADDRESS	GENERIC		IPADDRESS:GENERIC:
38	IPADDRESS_POOL	GENERIC		IPADDRESS_POOL:GENERIC:
39	KEYSTONE	OPENSTACK		KEYSTONE:OPENSTACK:
40	LINK	GENERIC		LINK:GENERIC:
41	LOCATION	GENERIC		LOCATION:GENERIC:
42	LUN	GENERIC		LUN:GENERIC:
43	MEMORY	GENERIC		MEMORY:GENERIC:
44	MONITOR	CUSTOM		MONITOR:CUSTOM:
45	MONITOR	GENERIC		MONITOR:GENERIC:
46	MONITOR	SELF		MONITOR:SELF:
47	MONITOR_ARGUMENTS	ARGUMENTS		MONITOR_ARGUMENTS:ARGUMENTS:
48	MONITOR_ARGUMENTS	CYPHER		MONITOR_ARGUMENTS:CYPHER:
49	NE	FIREWALL		NE:FIREWALL:
50	NE	LOAD_BALANCER		NE:LOAD_BALANCER:
51	NE	ROUTER		NE:ROUTER:
52	NE	SWITCH		NE:SWITCH:
53	NETWORK	BACKEND		NETWORK:BACKEND:

54	NETWORK	EXTERNAL		NETWORK:EXTERNAL:
55	NETWORK	FRONTEND		NETWORK:FRONTEND:
56	NETWORK	GENERIC		NETWORK:GENERIC:
57	NETWORK	INTERNAL		NETWORK:INTERNAL:
58	NETWORK	OPENSTACK	EXTERNAL	NETWORK:OPENSTACK:EXTERNAL
59	NETWORK	OPENSTACK	PRIVATE	NETWORK:OPENSTACK:PRIVATE
60	NETWORK	OPENSTACK	PROVIDER	NETWORK:OPENSTACK:PROVIDER
61	NETWORK	OPENSTACK		NETWORK:OPENSTACK:
62	NETWORK_SERVICE	GENERIC		NETWORK_SERVICE:GENERIC:
63	NETWORKING	OPENSTACK		NETWORKING:OPENSTACK:
64	PHYSICAL_MACHINE	GENERIC		PHYSICAL_MACHINE:GENERIC:
65	POLICY	AFFINITY		POLICY:AFFINITY:
66	POLICY	ANTI_AFFINITY		POLICY:ANTI_AFFINITY:
67	POLICY	ASSIGNMENT_GROUP		POLICY:ASSIGNMENT_GROUP:
68	POLICY	ASSIGNMENT_RELATIONSHIP		POLICY:ASSIGNMENT_RELATIONSHIP:
69	POLICY	ASSIGNMENT_TARGET		POLICY:ASSIGNMENT_TARGET:
70	POLICY	ENTITY_ASSIGN		POLICY:ENTITY_ASSIGN:
71	POLICY	ENTITY_RANGE		POLICY:ENTITY_RANGE:
72	POLICY	ENTITY_SCALE		POLICY:ENTITY_SCALE:
73	POLICY	OVER_SUBCRIPTION		POLICY:OVER_SUBCRIPTION:
74	POLICY	POSTPRE_PROCESSING		POLICY:POSTPRE_PROCESSING:
75	POLICY	PROCESSING_ACTION		POLICY:PROCESSING_ACTION:
76	POLICY	PROCESSING_TREE		POLICY:PROCESSING_TREE:
77	POLICY	VALUE_GENERATION		POLICY:VALUE_GENERATION:
78	POLICY	VALUE_VALIDATION		POLICY:VALUE_VALIDATION:
79	PORT	GENERIC		PORT:GENERIC:
80	PROPAGATION_RULE	GENERIC		PROPAGATION_RULE:GENERIC:
81	QUOTA	GENERIC		QUOTA:GENERIC:
82	QUOTA	VIRTUAL_CORE		QUOTA:VIRTUAL_CORE:
83	QUOTA	VIRTUAL_CPU		QUOTA:VIRTUAL_CPU:
84	QUOTA	VIRTUAL_DISK		QUOTA:VIRTUAL_DISK:
85	QUOTA	VIRTUAL_LUN		QUOTA:VIRTUAL_LUN:
86	QUOTA	VIRTUAL_MEMORY		QUOTA:VIRTUAL_MEMORY:
87	RACK	GENERIC		RACK:GENERIC:
88	REGION	OPENSTACK		REGION:OPENSTACK:
89	REPO	IMAGE	LOCAL	REPO:IMAGE:LOCAL
90	RESOURCE_POOL	GENERIC		RESOURCE_POOL:GENERIC:
91	SECURITY_GROUP	OPENSTACK		SECURITY_GROUP:OPENSTACK:
92	SECURITY_GROUP_RULE	OPENSTACK		SECURITY_GROUP_RULE:OPENSTACK:
93	SERVER	GENERIC		SERVER:GENERIC:
94	STORAGE	OPENSTACK		STORAGE:OPENSTACK:
95	SUBNETWORK	GENERIC		SUBNETWORK:GENERIC:
96	SUBNETWORK	OPENSTACK		SUBNETWORK:OPENSTACK:
97	SWIFT	OPENSTACK		SWIFT:OPENSTACK:
98	TASK_DEFINITION	GENERIC		TASK_DEFINITION:GENERIC:
99	TASK_LIST_DEFINITION	GENERIC		TASK_LIST_DEFINITION:GENERIC:
100	TEMPLATE	GENERIC		TEMPLATE:GENERIC:
101	TENANT	GENERIC		TENANT:GENERIC:
102	TENANT	OPENSTACK		TENANT:OPENSTACK:
103	VIM	CS8		VIM:CS8:
104	VIM	GENERIC		VIM:GENERIC:
105	VIM	HELION		VIM:HELION:
106	VIM	ICEHOUSE		VIM:ICEHOUSE:
107	VIM	OPENSTACK		VIM:OPENSTACK:
108	VIRTUAL_CORE	GENERIC		VIRTUAL_CORE:GENERIC:
109	VIRTUAL_DATACENTER	GENERIC		VIRTUAL_DATACENTER:GENERIC:
110	VIRTUAL_DISK	GENERIC		VIRTUAL_DISK:GENERIC:
111	VIRTUAL_IP	GENERIC		VIRTUAL_IP:GENERIC:
112	VIRTUAL_LINK	GENERIC		VIRTUAL_LINK:GENERIC:
113	VIRTUAL_LUN	GENERIC		VIRTUAL_LUN:GENERIC:
114	VIRTUAL_MACHINE	GENERIC		VIRTUAL_MACHINE:GENERIC:
115	VIRTUAL_MACHINE	KVM		VIRTUAL_MACHINE:KVM:
116	VIRTUAL_MACHINE	VMWARE		VIRTUAL_MACHINE:VMWARE:

117	VIRTUAL_MEMORY	GENERIC		VIRTUAL_MEMORY:GENERIC:
118	VIRTUAL_PORT	BACKEND		VIRTUAL_PORT:BACKEND:
119	VIRTUAL_PORT	EXTERNAL		VIRTUAL_PORT:EXTERNAL:
120	VIRTUAL_PORT	FRONTEND		VIRTUAL_PORT:FRONTEND:
121	VIRTUAL_PORT	GENERIC		VIRTUAL_PORT:GENERIC:
122	VIRTUAL_PORT	INTERNAL		VIRTUAL_PORT:INTERNAL:
123	VIRTUAL_PORT	MANAGEMENT		VIRTUAL_PORT:MANAGEMENT:
124	VIRTUAL_PORT	PRIVATE		VIRTUAL_PORT:PRIVATE:
125	VIRTUAL_PORT	PROVIDER		VIRTUAL_PORT:PROVIDER:
126	VL_ENDPOINT	BIDIRECTIONAL	OPENSTACK	VL_ENDPOINT:BIDIRECTIONAL:OPENSTACK
127	VNF	GENERIC		VNF:GENERIC:
128	VNF	NFVD		VNF:NFVD:
129	VNF_COMPONENT	ASSURANCE_GATEWAY		VNF_COMPONENT:ASSURANCE_GATEWAY:
130	VNF_COMPONENT	ECP		VNF_COMPONENT:ECP:
131	VNF_COMPONENT	GENERIC		VNF_COMPONENT:GENERIC:
132	VNF_COMPONENT	HPSA		VNF_COMPONENT:HPSA:
133	VNF_COMPONENT	LOCK_MANAGER		VNF_COMPONENT:LOCK_MANAGER:
134	VNF_COMPONENT	NEO4J		VNF_COMPONENT:NEO4J:
135	VNF_COMPONENT	OPEN_MEDIATION		VNF_COMPONENT:OPEN_MEDIATION:
136	VNF_COMPONENT	ORACLE		VNF_COMPONENT:ORACLE:
137	VNF_COMPONENT	POSTGRES		VNF_COMPONENT:POSTGRES:
138	VNF_COMPONENT	SITESCOPE		VNF_COMPONENT:SITESCOPE:
139	VNF_COMPONENT	SOSA		VNF_COMPONENT:SOSA:
140	VNF_COMPONENT	UCA		VNF_COMPONENT:UCA:
141	VNF_ENDPOINT	BIDIRECTIONAL	OPENSTACK	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK
142	VNFM_API	GENERIC		VNFM_API:GENERIC:
143	VNFMANAGER	GENERIC		VNFMANAGER:GENERIC:
144	ZONE	BACKEND		ZONE:BACKEND:
145	ZONE	CE		ZONE:CE:
146	ZONE	EXTERNAL		ZONE:EXTERNAL:
147	ZONE	FIREWALL		ZONE:FIREWALL:
148	ZONE	FRONTEND		ZONE:FRONTEND:
149	ZONE	INTERNAL		ZONE:INTERNAL:

Table 1 Artifacts in NFV Director V3.0

2.5.2 Relationships

There are 41 different relation types in NFV Director.

Relationships between artifacts and those 41 different relation types produce a total of 593 parent-child relationships in NFV Director V3.0.

Next table shows the 41 different relation types and an example of use.

	Parent Artifact	Relation type	Child Artifact
1	VIRTUAL_DATACENTER:GENERIC:	ALLOCATE	PHYSICAL_MACHINE:GENERIC:
2	CORE:GENERIC:	ALLOCATED	VIRTUAL_CORE:GENERIC:
3	DATACENTER:GENERIC:	APPLY	POLICY:ENTITY_RANGE:
4	IPADDRESS_POOL:GENERIC:	ASSIGNED	INTERFACE:GENERIC:
5	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_DISK:GENERIC:
6	SECURITY_GROUP:OPENSTACK:	ATTACHED	SECURITY_GROUP_RULE:OPENSTACK:
7	AUTHENTICATION:OPENSTACK:	AUTHENTICATE	REGION:OPENSTACK:
8	VIRTUAL_LINK:GENERIC:	BIDIRECTIONAL	VL_ENDPOINT:BIDIRECTIONAL:OPENSTACK
9	PORT:GENERIC:	CONFIGURED	INTERFACE:GENERIC:
10	VL_ENDPOINT:BIDIRECTIONAL:OPENSTACK	CONNECT_THROUGH	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK
11	ZONE:FRONTEND:	CONNECTED	ENDPOINT:GENERIC:
12	VNF_COMPONENT:UCA:	CONNECTED_TO	VNF_COMPONENT:ORACLE:
13	QUOTA:VIRTUAL_CORE:	CONSUMED	VIRTUAL_CORE:GENERIC:
14	QUOTA:GENERIC:	CONTAINS	QUOTA:VIRTUAL_DISK:
15	TENANT:GENERIC:	DEDICATED	NETWORK:GENERIC:

16	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK	DEPLOY	VIRTUAL_PORT:GENERIC:
17	AVAILABILITY_ZONE:OPENSTACK:	DIVIDE	HOSTAGGREGATES:OPENSTACK:
18	EXECUTION_TASK:GENERIC:	EXECUTE	EXECUTION_TASK_LIST:GENERIC:
19	VNF:GENERIC:	EXPOSE	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK
20	GROUP:GENERIC:	GENERIC	LOCATION:GENERIC:
21	VNF_COMPONENT:ASSURANCE_GATEWAY:	HA	VNF_COMPONENT:ASSURANCE_GATEWAY:
22	IMAGE_STORAGE:OPENSTACK:	HAS	IMAGE:OPENSTACK:
23	VNFMANAGER:GENERIC:	IMPLEMENTS	VNF:GENERIC:
24	NETWORK_SERVICE:GENERIC:	INCLUDE	VNF:GENERIC:
25	ENDPOINT:GENERIC:	IS_COMPOSED_BY	INTERFACE:GENERIC:
26	LOCATION:GENERIC:	IS_LOCATED	DATACENTER:GENERIC:
27	VIM:CS8:	MANAGE	HYPERVISOR:KVM:
28	VIRTUAL_MACHINE:KVM:	MEASURE	MONITOR:GENERIC:
29	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	DATACENTER:GENERIC:
30	POLICY:ENTITY_RANGE:	OVER	VIM:HELION:
31	SERVER:GENERIC:	PHYSICALLY_CONTAINS	CPU:GENERIC:
32	RESOURCE_POOL:GENERIC:	PROVIDES	VIRTUAL_CORE:GENERIC:
33	VNF_COMPONENT:ASSURANCE_GATEWAY:	READS	VNF_COMPONENT:NEO4J:
34	TENANT:GENERIC:	SHARED	VIRTUAL_CORE:GENERIC:
35	VNF:GENERIC:	STATUS_CHANGED_BY	PROPAGATION_RULE:GENERIC:
36	NETWORK:GENERIC:	SUBDIVIDE	SUBNETWORK:GENERIC:
37	MONITOR:CUSTOM:	TRANSGRESS	CONDITION:GENERIC:
38	CONDITION:GENERIC:	TRIGGERS	ACTION:GENERIC:
39	VIM:GENERIC:	USE	AUTHENTICATION:OPENSTACK:
40	TENANT:OPENSTACK:	USES	NETWORK:OPENSTACK:
41	VNF_COMPONENT:ASSURANCE_GATEWAY:	WRITES	VNF_COMPONENT:NEO4J:

Table 2 Relationships in NFV Director V3.0

You can see all the 593 existing relationships in NFV Director V3.0 in Appendix C.

2.6 Examples

This section presents some VNF examples using artifacts and relationships shown in previous sections

2.6.1 Example 1: Basic VNF

The simplest VNF is shown in next figure:

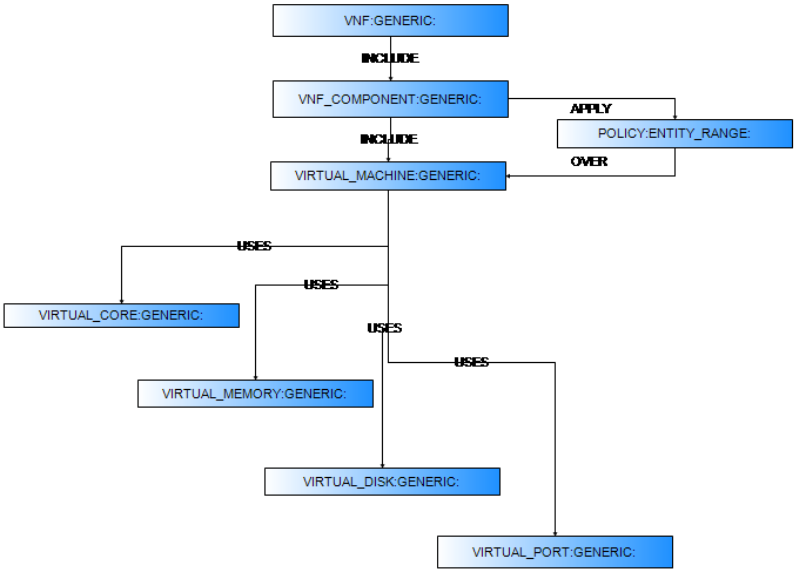


Figure 23 Simple VNF

A basic scenario composed by a generic Virtual Machine with Virtual Core, Virtual Memory, Virtual Disk and Virtual Port, in which it is applied an entity range policy.

2.6.2 Example 2: Application Server

It is composed by two Virtual Machines: a Web Server and a Database Server.

2.6.2.1 Initial configuration

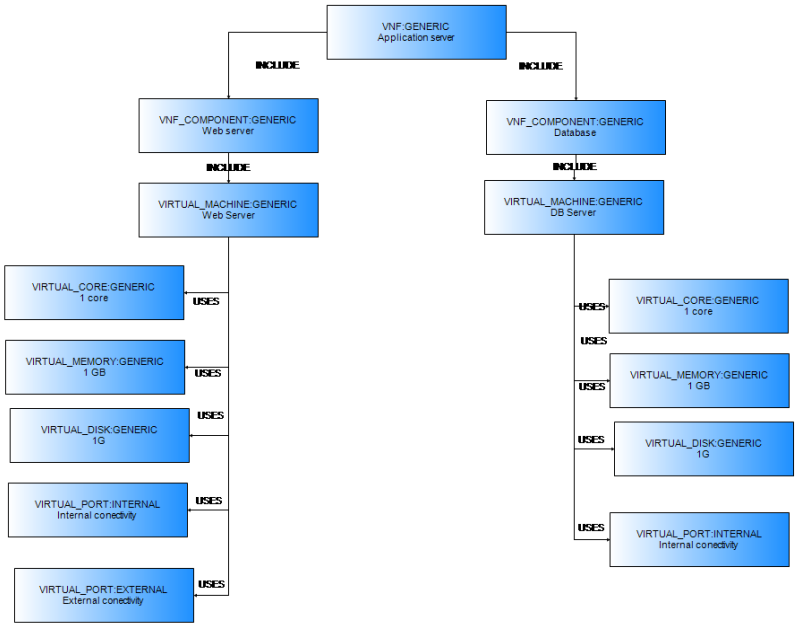


Figure 24 Application Server

2.6.2.2 Application Server Behavior

Added policies (scale in/scale out) and monitors to previous example.

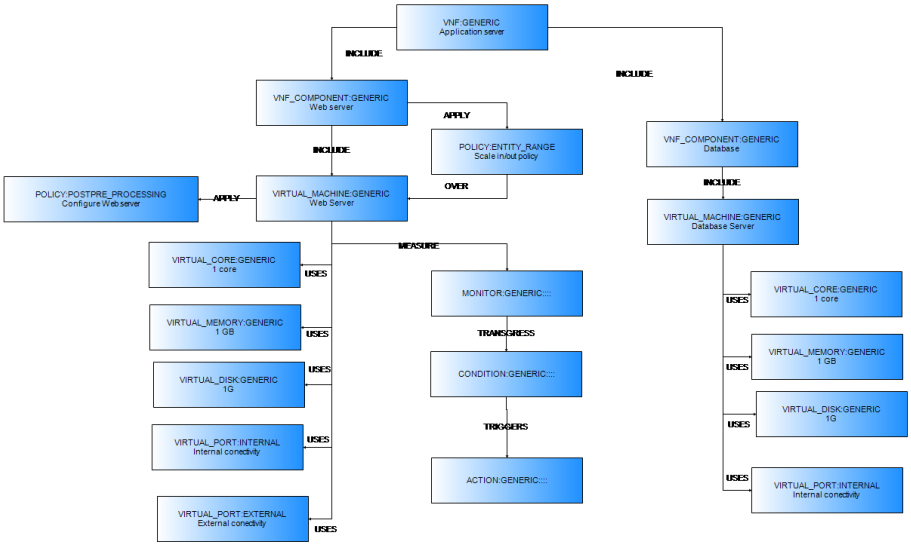


Figure 25 Application Server with policies and monitors

2.7 NFV Director Data Model: Soap UI

SoapUI is a tool that allows you to play with NFV Director Data Model.

You can access to NFV Director Data Model using this tool and executing all the available operations on the model.

2.7.1 Introduction

SoapUI application (http://soapui.org/) is a Web Service Simulator for making petitions to our system, simulating external ones.

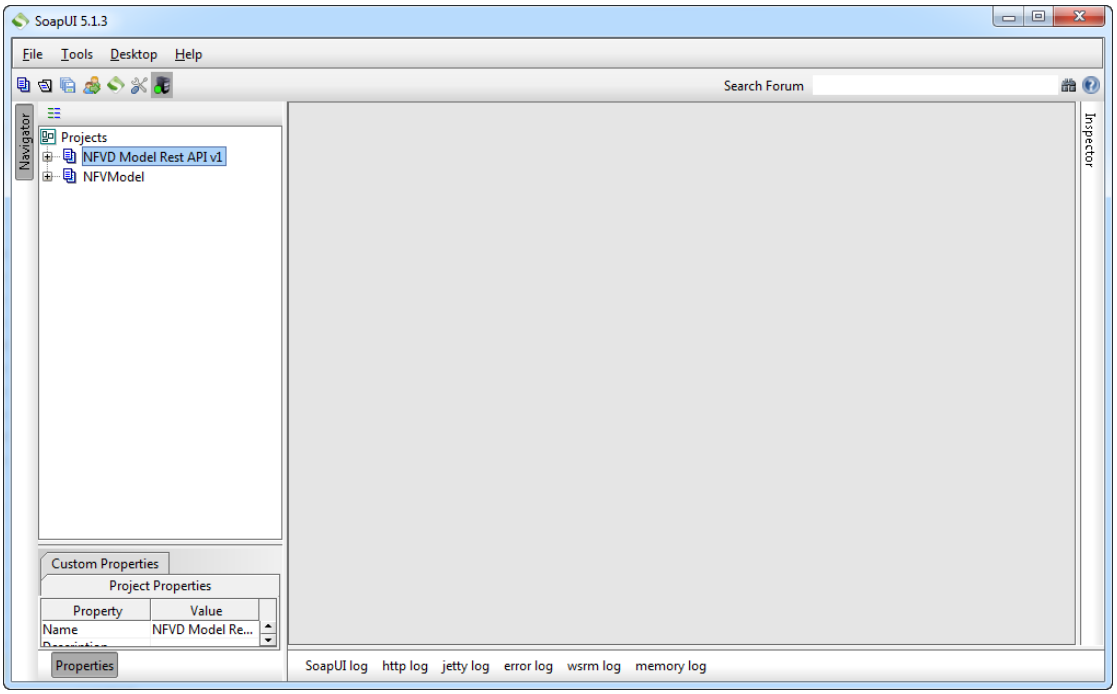


Figure 26 SoapUI

We can send instructions to HPSA via REST API.

Note

Read *HP NFV Director v3.0 Rest API v1 - Reference guide* for more detailed info about NFV Director v3.0 REST API.

The first step is to import an example project provided in the virtual machine.

This project is located at /opt/OV/ServiceActivator/solutions/NFVModel/docs/testing/NFVD-soapui-project.xml.

Left click File > Import Project and search the location above.

2.8 Data Model

The REST API for accessing to NFV Director Data Model is organized in three main groups corresponding to different artifacts present in NFV Director:

- Definition
- Template
- Instances

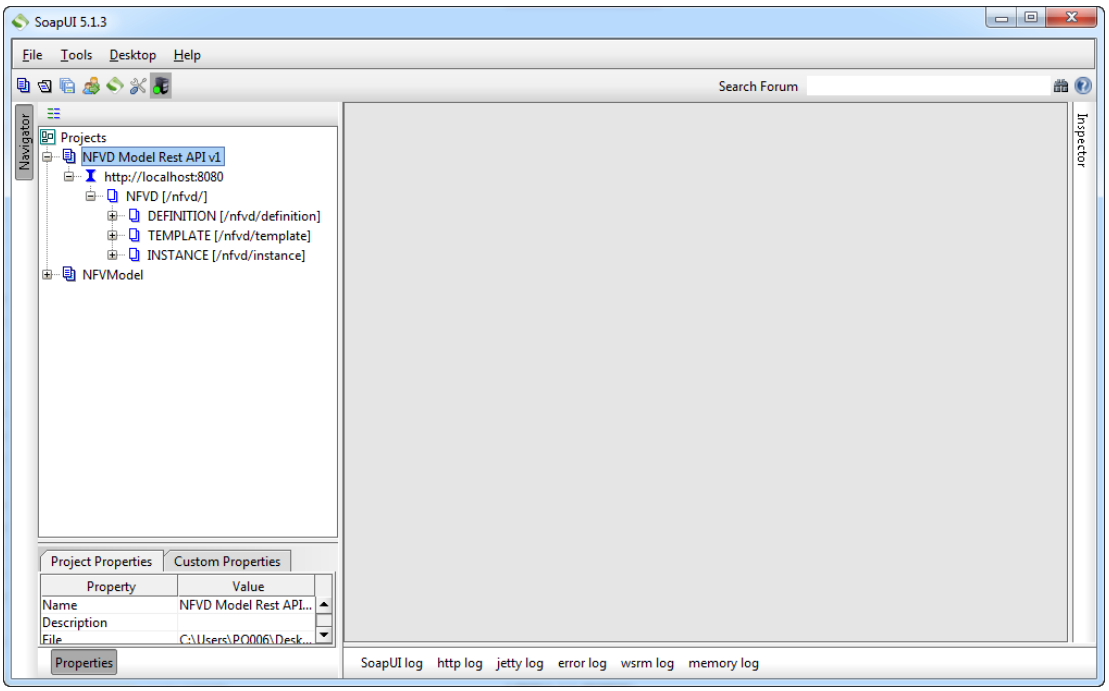


Figure 27 SoapUI: NFVD Model Rest API

Every action in NFV Director REST API is associated to corresponding REST protocol action. For example, to get all the definitions, we use the GET command on HTTP protocol using the URL (resource) corresponding to that functionality:

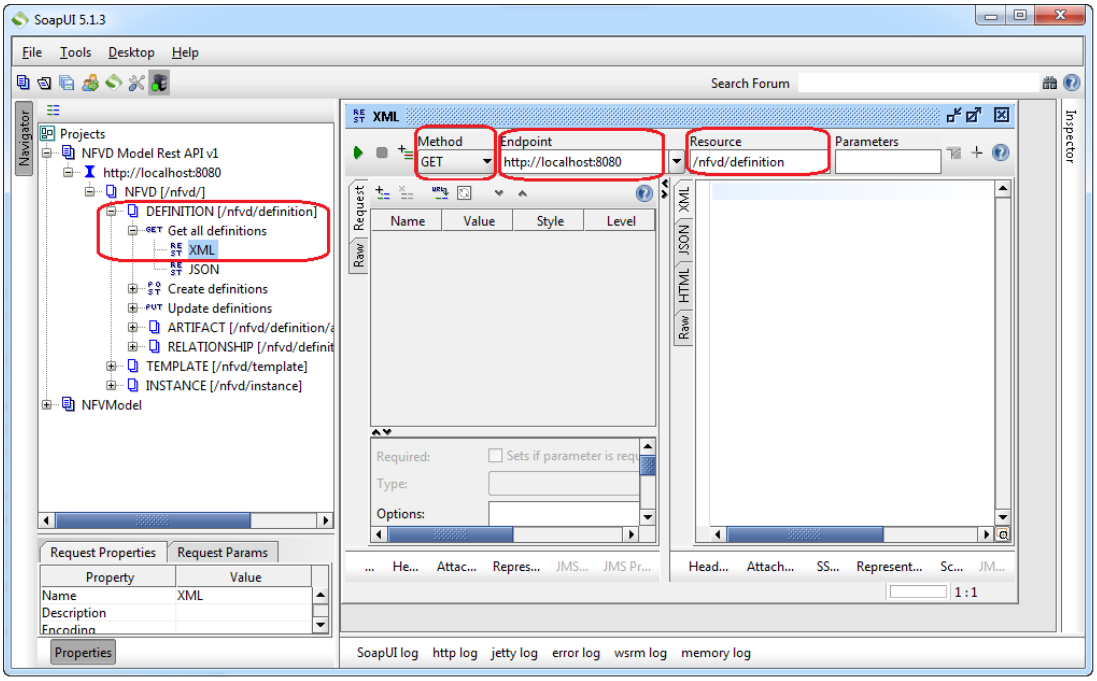


Figure 28 SoapUI: Get all definitions

You can send your requests (request body) both in XML format as JSON format.

The RAW request example sent in this case would be the next:

```
GET http://localhost:8080/nfvd/definition HTTP/1.1
Accept-Encoding: gzip,deflate
Accept: application/xml
Host: localhost:8080
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

If the request is correct, you will get a RAW Success response similar to this:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 11:18:40 GMT

<definitions xmlns="http://www.hp.com/nfvd">
   <artifact-definitions>
      <artifact-definition id="1009"
uri="/nfvd/definition/artifact/1009">
         <categories>
            <category>
               <attributes>
                  <attribute>
                     <label>Operational Status</label>
                     <mandatory>false</mandatory>
                     <order>1</order>
                     <type>TEXT</type>
                     <unit>TEXT</unit>
                  </attribute>
                  <attribute>
                     <label>LifeCycle_State_Date</label>
                     <mandatory>false</mandatory>
```

```
                      <order>5</order>
                      <type>Date</type>
                      <unit>Date</unit>
                   </attribute>
                </attributes>
                <label>STATUS</label>
                <order>2</order>
             </category>
……————
……————
          <family>VIM</family>
          <physical>false</physical>
       </artifact-definition>
……————————
……————————
     </artifact-definitions>
     <relationship-definitions>
       <relationship-definition id="755"
uri="/nfvd/definition/relationship/755">
          <child-artifact-definition>
             <category>PROCESSING_ACTION</category>
             <family>POLICY</family>
          </child-artifact-definition>
          <enabled>true</enabled>
          <parent-artifact-definition>
             <category>PROCESSING_TREE</category>
             <family>POLICY</family>
          </parent-artifact-definition>
          </available-status>
          <type>APPLY</type>
       </relationship-definition>
……————————
……————————
    </relationship-definitions>
</definitions>
```


All about the NFV Director REST API in presented in the following chapter *Northbound Interface.*

Chapter 3

Northbound Interface

The Northbound Interface is based on Service Activator Sosa Module and its main function is to manage requests using queues.

The Northbound architecture is represented in next figure:

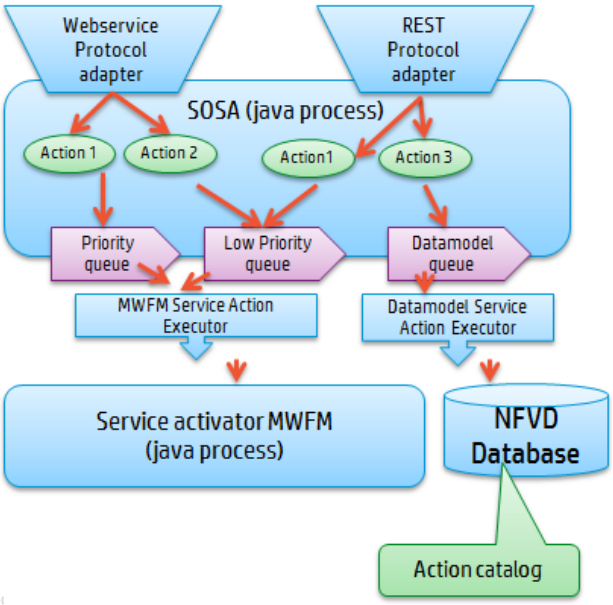


Figure 29 Northbound Architecture

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 (java processes).
- REST Protocol Adapter. It allows access to data model.

Requests can be listened from an out of the box web service interface. If the request come in other format or using other protocol, a new protocol adapter can be create to listen rest, ssh, telnet, snmp , etc ... new requests.

Once a request is received it can start an action (Service Action) or a group of actions (Service Order).

The catalog of actions and group of actions remains the same regardless the protocol adapter used so it is possible to call the same action using different protocols.

Each action goes to a specific queue and each queue has an executor that executes the pending actions (executors can be used in several queues).

Catalog of actions, queues and historical request data can be accessed and managed through GUI and are stored in database.

3.1 Data Model

HP NFV Director v3.0 provides a renewed REST API where database has completely changed from previous version and all the queries have been redesigned improving the system performance.

For interacting with NFV Director Data Model, there are a set of operations to manage all the content related to artifacts and relationships.

All the actions are executed by a REST protocol for which NFVD has implemented its corresponding adapter.

Operations are grouped by flavors (definitions, templates and instances) and their actions can be one of the following:

- Create (POST method)
- Get (GET method)
- Update (PUT method)
- Delete (DELETE method)

And possible objects to interact with are:

- Artifacts (any entity)
- Relationships

For more details about the REST API, refer to *HP NFV Director v3.0 Rest API - Reference guide*.

3.2 Automation

Once a request is received, it can start an action (Service Action) or a group of actions (Service Order).

Basic Service Actions involve Virtual Network Functions (VNFs) management and Monitors. Also, it can be related (among others) to Inventory (scale operations), Network Services and Orchestration.

Next chapter shows more detailed info about Service Actions.

3.3 SOSA as NFVD operator

Apart from Web user interface, NFVD operations can be executed using SOSA web services interface also by sending appropriate commands. In this section xml format is used for illustration.

Note

You can read more about SOSA in the *HPSA Extension Pack SOSA v6.0 - Developer Reference*.

3.3.1 Understanding SOAP requests

Identify the operation with its service action:

```
<dyn:name>
<dyn:type>
<dyn:action>
```

The three parameters identify uniquely the service action to be executed.

Request body:

```
<dyn:name>REQUEST</dyn:name>
<dyn:value>REQUEST-BODY</dyn:value>
```

It has to be set here as XML structure that SOSA can read and convert into an NFVModel object (artifact definition, instance, and template or artifact relationship).

```
<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 type="NFVD" name="DEFART" action="CREATE">
               <dyn:service>
                  <dyn:name>DEFART</dyn:name>
                  <dyn:type>NFVD</dyn:type>
                  <dyn:action>CREATE</dyn:action>
                  <dyn:characteristics>
                     <!--1 or more repetitions:-->
                     <dyn:characteristic>
                        <dyn:name>REQUEST</dyn:name>
                        <dyn:value><![CDATA[
                              <REQUEST>
                        ]]></dyn:value>
                     </dyn:characteristic>
                  </dyn:characteristics>
               </dyn:service>
            </dyn:services>
         </dyn:serviceRequest>
         <ngws:user>foo</ngws:user>
      </ngws:startDynamicOrderSync>
   <ngws:startDynamicOrderAsync><dyn:serviceRequest/><ngws:user/></ngws:startDynamicOrderAsync></soapenv:Body>
</soapenv:Envelope>
```

3.3.2 Operations

Next sections show info about all the available operations in NFV Director v3.0

You can find more info about these operations in *HP NFV Director v3.0 Rest API v1 - Reference Guide*.

3.3.2.1 Definitions

Next table shows all the operations related to Definitions.

	Type	Subtype	Name	Description	HTTP method	Resource
1	Definitions	None	Get all definitions	Create definitions (artifacts and relationships)	GET	/nfvd/definition
2	Definitions	None	Create definitions	Create definitions (artifacts and relationships)	POST	/nfvd/definition
3	Definitions	None	Update definitions (artifacts and relationships) in XML format	Update definitions (artifacts and relationships)	PUT	/nfvd/definition
4	Definitions	Artifacts	Get artifact definition by id	Get artifact definition by its id	GET	/nfvd/definition/artifact/{id}
5	Definitions	Artifacts	Get artifact definition detail by id	Get artifact definition detail (artifact and its relationships) by its id	GET	/nfvd/definition/artifact/{id}/detail
6	Definitions	Artifacts	Get artifact definition list by definition parameters	Get artifact definition list by definition parameters	GET	/nfvd/definition/artifact/
7	Definitions	Artifacts	Create a list of artifact definition	Create a list artifact definition	POST	/nfvd/definition/artifact/
8	Definitions	Artifacts	Update an artifact definition by id	Update an artifact definition identified by the given id	PUT	/nfvd/definition/artifact/{id}

9	Definitions	Artifacts	Update a list of artifact definition	Update a list of artifact definition	PUT	/nfvd/definition/artifact/
10	Definitions	Artifacts	Delete artifact definition by id	Delete artifact definition by its id	DELETE	/nfvd/definition/artifact/{id}
11	Definitions	Artifacts	Delete artifact definition by definition parameters	Delete artifact definition by its definition parameters	DELETE	/nfvd/definition/artifact/
12	Definitions	Relationships	Get relationship definition by id	Get relationship definition by its id	GET	/nfvd/definition/relationship/{id}
13	Definitions	Relationships	Get relationship definition list by definition parameters	Get relationship definition list by definition parameters	GET	/nfvd/definition/relationship/
14	Definitions	Relationships	Create a list of relationship definition	Create a list relationship definition	POST	/nfvd/definition/relationship/
15	Definitions	Relationships	Update a relationship definition by id	Update a relationship definition identified by the given id	PUT	/nfvd/definition/relationship/{id}
16	Definitions	Relationships	Update a list of relationship definition	Update a list of relationship definition	PUT	/nfvd/definition/relationship/
17	Definitions	Relationships	Delete relationship definition by id	Delete relationship definition by its id	DELETE	/nfvd/definition/relationship/{id}
18	Definitions	Relationships	Delete relationship definition by definition parameters	Delete artifact definition by its definition parameters	DELETE	/nfvd/definition/artifact/

Table 3 Definitions

3.3.2.2 Templates

Following table lists all the operations about Templates.

	Type	Subtype	Name	Description	HTTP method	Resource
19	Templates	None	Get template from date	Get template (artifacts and relationships) from date	GET	/nfvd/template/query/elements
20	Templates	None	Get template elements recursively	Get template (artifacts and relationships) recursively	GET	/nfvd/template/query/elements/recursively
21	Templates	None	Get trees ids	Get template trees ids	GET	/nfvd/template/tree
22	Templates	None	Get template by tree id	Get template (artifacts and relationships) by tree id	GET	/nfvd/template/tree/{id}
23	Templates	None	Download tree	Download tree	GET	/nfvd/template/download
24	Templates	None	Create template	Create artifact and relationship set	POST	/nfvd/template
25	Templates	None	Create template tree	Create a new template tree	POST	/nfvd/template/tree

26	Templates	None	Create tree UUID	Create a new template tree UUID	POST	/nfvd/template/tree/uuid
27	Templates	None	Upload tree	Uploads tree definitions Elements (Artifacts and Relationships) and Trees	POST	/nfvd/template/upload
28	Templates	None	Update template	Update artifact and relationship set	PUT	/nfvd/template
29	Templates	None	Update template tree	Updating template tree	PUT	/nfvd/template/tree/{id}
30	Templates	None	Delete template tree	Deleting Template tree (just tree, neither artifacts nor relationships content)	DELETE	/nfvd/template/tree/{id}
31	Templates	None	Delete tree and content	Deleting template tree and its related artifacts and relationships for id	DELETE	/nfvd/template/tree/{id}/cascade
32	Templates	None	Delete element from tree	Deleting element from tree id	DELETE	/nfvd/template/tree/{id}/{elementId}
33	Templates	Artifacts	Get artifact templates by parameters	Get artifact templates by parameters	GET	/nfvd/template/artifact
34	Templates	Artifacts	Get artifact template by id	Get artifact template by id	GET	/nfvd/template/artifact/{id}
35	Templates	Artifacts	Get artifact template by path	Get artifact template by path	GET	/nfvd/template/artifact/query/path
36	Templates	Artifacts	Get parents of an artifact	Get artifact templates parents of an artifact	GET	/nfvd/template/artifact/query/parents
37	Templates	Artifacts	Get parents of an artifact recursively	Get artifact templates parents of an artifact recursively	GET	/nfvd/template/artifact/query/parents/recursive
38	Templates	Artifacts	Get children of an artifact	Get artifacts children of an artifact	GET	/nfvd/template/artifact/query/children
39	Templates	Artifacts	Get children of an artifact recursively	Get artifacts children of an artifact recursively	GET	/nfvd/template/artifact/query/children/recursive
40	Templates	Artifacts	Get tree ids	Get tree ids in which the artifact template id belongs to	GET	/nfvd/template/artifact/{id}/tree
41	Templates	Artifacts	Get by misc parameters	Get artifact by misc parameters	GET	/nfvd/template/artifact/query/parameters
42	Templates	Artifacts	Get attribute by parameters	Get attribute by parameters	GET	/nfvd/template/artifact/{id}/{cat}/{att}

43	Templates	Artifacts	Create artifacts template	Creating ArtifactTemplate set	POST	/nfvd/template/artifact
44	Templates	Artifacts	Create artifacts template with new UUID	Create artifacts with new UUID	POST	/nfvd/template/artifact/uuid
45	Templates	Artifacts	Create artifact template attribute	Create a new attribute in a category of an artifact	POST	/nfvd/template/artifact/{id}/{cat}/{att}
46	Templates	Artifacts	Update artifacts template	Update artifact set	PUT	/nfvd/template/artifact
47	Templates	Artifacts	Update attribute of artifact template	Update attribute in a category of an artifact	PUT	/nfvd/template/artifact /{id}/{cat}/{att}
48	Templates	Artifacts	Delete artifact template by id	Delete artifact template by id	DELETE	/nfvd/template/artifact/{id}
49	Templates	Artifacts	Delete attribute of artifact template	Delete attribute in a category of artifact template	DELETE	/nfvd/template/artifact/{id}/{cat}/{att}
50	Templates	Relationships	Get relationships template	Get relationships template by parameters	GET	/nfvd/template/relationship
51	Templates	Relationships	Get relationship template by id	Get relationship template by id	GET	/nfvd/template/artifact/{id}
52	Templates	Relationships	Get tree ids	Get tree ids in which the relationship template id belongs to	GET	/nfvd/template/relationship/{id}/tree
53	Templates	Relationships	Get attribute by parameters	Get attribute in a category of a relationship	GET	/nfvd/template/relationship/{id}/{cat}/{att}
54	Templates	Relationships	Create relationship template	Creating relationship set	POST	/nfvd/template/relationship
55	Templates	Relationships	Create relationship template attribute	Create attribute in a category of a relationship	POST	/nfvd/template/relationship/{id}/{cat}/{att}
56	Templates	Relationships	Update relationship template	Update rlationship set	PUT	/nfvd/template/relationship
57	Templates	Relationships	Update attribute of relationship template	Update attribute in a category of a relationship	PUT	/nfvd/template/relationship/{id}/{cat}/{att}
58	Templates	Relationships	Delete relationship template by id	Delete relationship template by id	DELETE	/nfvd/template/relationship/{id}
59	Templates	Relationships	Delete relationship template by parameters	Delete relationship template by parameters	DELETE	/nfvd/template/relationship

60	Templates	Relationships	Delete attribute of relationship template	Delete attribute in a category of a relationship template	DELETE	/nfvd/template/relationhip/{id}/{cat}/{att}

Table 4 Templates

3.3.2.3 Instances

This table shows operations related to Instances.

	Type	Subtype	Name	Description	HTTP method	Resource
61	Instances	None	Get instances from date	Gets instances from a date.	GET	/nfvd/instance/query/elements
62	Instances	None	Gets instances recursively	Gets instance elements recursively	GET	/nfvd/instance/query/elements/recusive
63	Instances	None	Get instance tree from id	Gets instance elements tree from id	GET	/nfvd/instance/tree/{id}
64	Instances	None	Get instance uuid list	Returns a instance uuid list based on name and/or type	GET	/nfvd/instance/tree
65	Instances	None	Download tree	Download tree	GET	/nfvd/instance/download
66	Instances	None	Create a new Instance	Creates a new instance	POST	/nfvd/instance
67	Instances	None	Create new Instance tree.	Creates a new instance tree	POST	/nfvd/instance/tree
68	Instances	None	Create new Instance tree UUID	Creates a new instance tree with a new UUID	POST	/nfvd/instance/tree/uuid
69	Instances	None	Upload tree	Uploads tree definitions	POST	/nfvd/instance/upload
70	Instances	None	Update Instance	Updates an instance.	PUT	/nfvd/instance
71	Instances	None	Update an instance tree	Updates an instance tree.	PUT	/tree/{id}
72	Instances	None	Delete tree by id	Deletes an instance tree by id.	DELETE	/tree/{id}
73	Instances	None	Delete tree and its content by id	Deletes a instance tree and its content by id	DELETE	/tree/{id}/cascade
74	Instances	None	Delete a instance tree by tree id and element id	Deletes a instance/relationship tree by tree id and element id	DELETE	/tree/{id}/elementId
75	Instances	Artifacts	Get an artifact instance by path	Gets an artifact instance by path	GET	/instance/artifact/query/path

76	Instances	Artifacts	Get artifact's parents	Return a list of artifacts which are parents of an artifact with the given id, reached by the given relationships and matches the definition	GET	/instance/artifact/query/parents
77	Instances	Artifacts	Get artifact's children	Return a list of Artifact who are children of given id, reached by the given relationships and match the definition	GET	/instance/artifact/query/children
78	Instances	Artifacts	Get artifact's parents recursively	Return a list of artifacts which are parents of an artifact with the given id, reached by the given relationships and match the definition recursively	GET	/instance/artifact/query/parents/recursive
79	Instances	Artifacts	Get artifact's children recursively	Return a list of artifacts which are children of an artifact with the given id, reached by the given relationships and match the definition recursively	GET	/instance/artifact/query/children/recursive
80	Instances	Artifacts	Get instance artifacts by miscellaneous parameters	Return a list of instance artifacts that match miscellaneous parameters	GET	/instance/artifact/query/parameters
81	Instances	Artifacts	Get instance artifact by Id	Get instance artifact by Id	GET	/instance/artifact/{id}
82	Instances	Artifacts	Get instance artifact by parameters	Return instance artifact based on artifactDefinition, and exactMatching,	GET	/instance/artifact/query/parameters
83	Instances	Artifacts	Get tree ids	Get tree ids in which the artifact instance id belongs to.	GET	/instance/artifact/{id}/tree
84	Instances	Artifacts	Get an artifact instance attribute	Return an artifact instance attribute.	GET	/instance/artifact/{id}/{cat}/{attr}
85	Instances	Artifacts	Create Artifact Instance	Creates a new ArtifactInstance.	POST	/instance/artifact

86	Instances	Artifacts	Create ArtifactInsance with a new UUID	Creates a new ArtifactInstance with a new UUID	POST	/instance/artifact/uuid
87	Instances	Artifacts	Create new attribute value	Creates a new attribute.	POST	/instance/artifact/{id}/{cat}/{attr}
88	Instances	Artifacts	Update Artifact Instance	Update an artifact instance.	PUT	/instance/artifact
89	Instances	Artifacts	Update attribute value	Updates an existing Attribute.	PUT	/instance/artifact/{id}/{cat}/{attr}
90	Instances	Artifacts	Delete artifact instance by id	Delete an artifact instance by id.	DELETE	/instance/artifact/{id}
91	Instances	Artifacts	Delete attribute	Delete an attribute.	DELETE	/instance/artifact/{id/{cat}/{attr}
92	Instances	Relationships	Get relationship instance by parameters	Get an relationship instance that marches the given parameters	GET	/instance/relationship
93	Instances	Relationships	Get relationship instance by ID	Get relationship instance searching by id.	GET	/instance/relationship/{id}
94	Instances	Relationships	Get tree ids	Get tree ids in which the relationship instance id belongs to	GET	/instance/relationship/{id}/tree
95	Instances	Relationships	Get an relationship instance attribute	Return a relationship instance attribute.	GET	/instance/relationship/{id}/{cat}/{attr}
96	Instances	Relationships	Create a relationship instance	Create a new relationship instance..	POST	/instance/relationship/
97	Instances	Relationships	Create new attribute	Creates a new Attribute.	POST	/instance/relationship/{id}/{cat}/{attr}
98	Instances	Relationships	Update Instance Relationship	Updates an instance relationship.	PUT	/instance/relationship
99	Instances	Relationships	Update attribute	Updates an existing Attribute.	PUT	/instance/relationship/{id}/{cat}/{attr}
100	Instances	Relationships	Delete instance relationship by id	Deletes a relationship by id	DELETE	/instance/relationship/{id}
101	Instances	Relationships	Delete by parameters	Deletes an attribute.	DELETE	/instance/relationship
102	Instances	Relationships	Delete Attribute	Deletes an attribute.	DELETE	/instance/relationship/{id/{cat}/{attr}

Table 5 Instances

3.4 Northbound Interface – Automation

The complete list of the automation processes 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 6 Automation processes

3.4.1 VNFs actions

Next table represents all the actions related to VNFs:

	Service Order Name	Service Order Service	Service Order Operation
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

Table 7 VNFs actions

3.4.1.1 Start Virtual Machine (START_VM)

Start Virtual Machines takes the machines selected and puts them in running state, and it could be checked on DDBB status and VIM status.

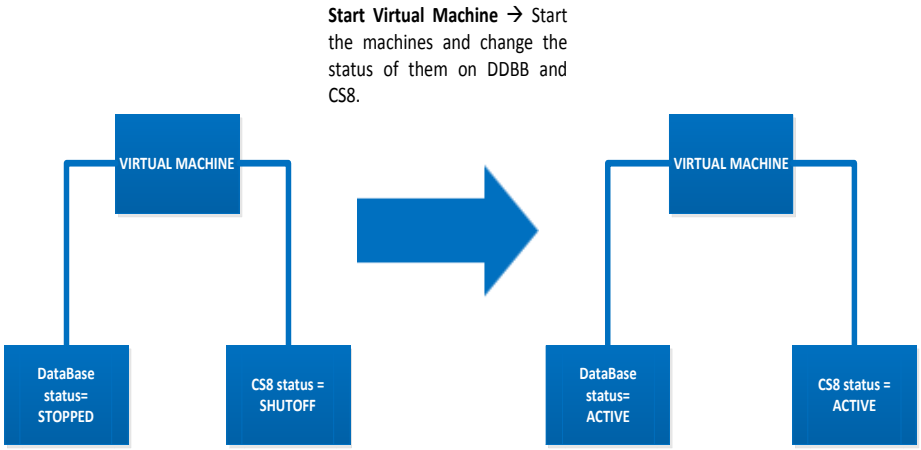


Figure 30 NBI: Start VM

The following shows a typical Soap call for a start virtual machine process:

```
<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>START VM</ngws:action>
         <!--Optional:-->
         <ngws:inputParams>
            <!--Zero or more repetitions:-->
            <ngws:param>

<ngws:name>INPUT ARTIFACTTREEID</ngws:name>
               <ngws:value>14093117003051</ngws:value>
```

```
            </ngws:param>
         </ngws:inputParams>
         <ngws:user></ngws:user>
         <!--Optional:-->
<ngws:userId></ngws:userId>
      </ngws:startServiceOrderAsync>
   </soapenv:Body>
</soapenv:Envelope>
```

3.4.1.2 Stop Virtual Machine (STOP_VM)

Stop Virtual Machines takes the machines selected and makes them stop, and it could be checked on DDBB status and CS8 status.

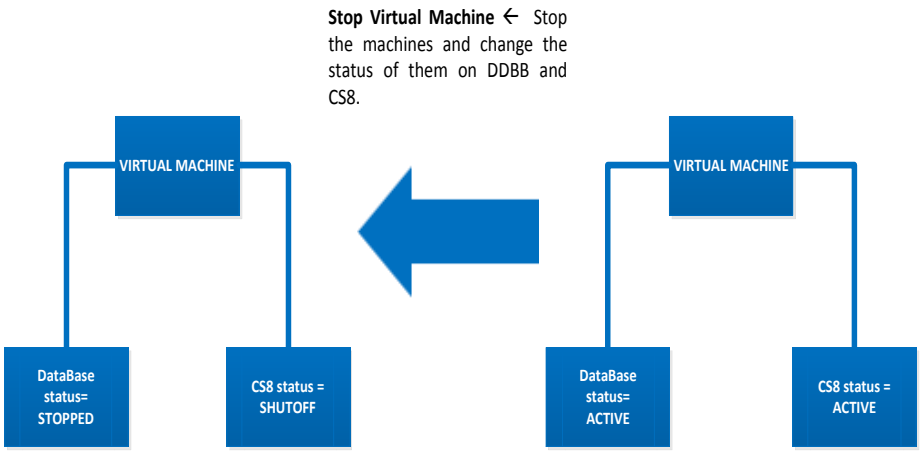


Figure 31 NBI: Stop VM

The following shows a typical Soap call for a stop virtual machine process:

```
<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>STOP VM</ngws:action>
         <!--Optional:-->
         <ngws:inputParams>
            <!--Zero or more repetitions:-->
            <ngws:param>

<ngws:name>INPUT ARTIFACTTREEID</ngws:name>
               <ngws:value>14093117003051</ngws:value>
            </ngws:param>
         </ngws:inputParams>
         <ngws:user></ngws:user>
         <!--Optional:-->
         <ngws:userId></ngws:userId>
      </ngws:startServiceOrderAsync>
   </soapenv:Body>
</soapenv:Envelope>
```

3.4.1.3 Scale Up: Enlarge attributes amount of VM (SCALE_UP)

Scale up process takes the policies nodes and increases the attributes amount as many instances of a VM as these policies determine.

The operation has an impact both in DB and on the OpenStack side, changing the VM flavor, according to the attributes set in DB.

The following shows a typical Soap call for a scale up process:

```
<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>SCALE UP</ngws:action>
         <ngws:inputParams>
            <!--Zero or more repetitions:-->
           <ngws:param>

<ngws:name>INPUT_INSTANCEARTIFACTID</ngws:name>
               <ngws:value>14087039824031</ngws:value>
            </ngws:param>
            <!--Optional:-->
            <ngws:param>
               <ngws:name>INPUT_SCALEALLTREE</ngws:name>
               <ngws:value>true</ngws:value>
            </ngws:param>
             <!--Optional:-->
            <ngws:param>
               <ngws:name>INPUT_FORCESTOP</ngws:name>
               <ngws:value>false</ngws:value>
            </ngws:param>
         </ngws:inputParams>
          <!--Optional:-->
         <ngws:user>?</ngws:user>
         <ngws:userId>?</ngws:userId>
      </ngws:startServiceOrderAsync>
   </soapenv:Body>
</soapenv:Envelope>
```

The “INPUT_SCALEALLTREE” and “INPUT_FORCESTOP” parameters are optional. The predefined values are “true” and “false” respectively.

3.4.1.4 Scale Down: Reduce attributes amount of VM (SCALE_DOWN)

Scale down process takes the policies nodes and decreases the attributes amount as many instances of a VM as these policies determine.

The operation has an impact both in DB and on the OpenStack side, changing the VM flavor, according to the attributes set in DB.

The following shows a typical Soap call for a scale down process:

```
<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>SCALE DOWN</ngws:action>
         <ngws:inputParams>
            <!--Zero or more repetitions:-->
           <ngws:param>

<ngws:name>INPUT_INSTANCEARTIFACTID</ngws:name>
               <ngws:value>14087039824031</ngws:value>
            </ngws:param>
            <!--Optional:-->
            <ngws:param>
               <ngws:name>INPUT_SCALEALLTREE</ngws:name>
               <ngws:value>true</ngws:value>
            </ngws:param>
             <!--Optional:-->
            <ngws:param>
               <ngws:name>INPUT_FORCESTOP</ngws:name>
               <ngws:value>false</ngws:value>
            </ngws:param>
         </ngws:inputParams>
          <!--Optional:-->
         <ngws:user>?</ngws:user>
         <ngws:userId>?</ngws:userId>
      </ngws:startServiceOrderAsync>
   </soapenv:Body>
</soapenv:Envelope>
```

The “INPUT_SCALEALLTREE” and “INPUT_FORCESTOP” parameters are optional. The predefined values are “true” and “false” respectively.

3.4.2 Inventory actions

Next table represents all the actions related to Inventory:

	Service Order Name	Service Order Service	Service Order Operation
2	NFVD	INVENTORY	CREATE_FROM_TEMPLATE
3	NFVD	INVENTORY	SCALE_DOWN
4	NFVD	INVENTORY	SCALE_IN

Table 8 Inventory actions

3.4.2.1 Scale-in: Delete an Existing VM (SCALE_IN)

Scale-in process takes the policies nodes and deletes as many instances of a VM as these policies determine.

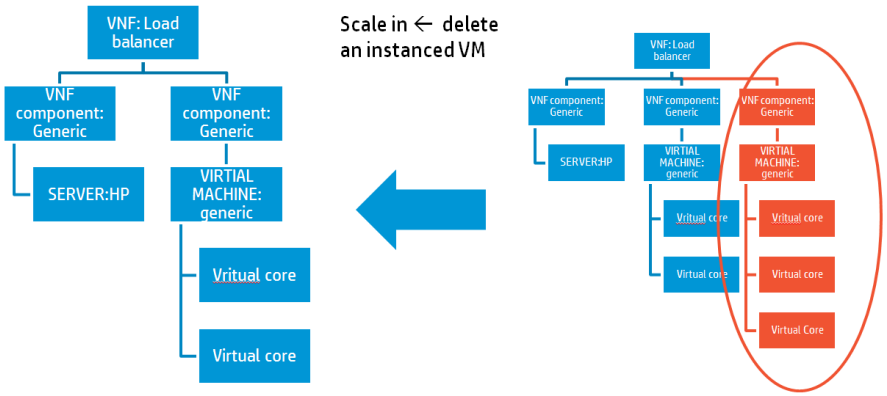


Figure 32 NBI: Scale-In

The following shows a typical Soap call for a scale–in process:

```
<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 type="NFVD" name="INVENTORY"
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>14032664806161</dyn:value>
                     </dyn:characteristic>
                  </dyn:characteristics>
               </dyn:service>
            </dyn:services>
         </dyn:serviceRequest>
         <ngws:user>foo</ngws:user>
      </ngws:startDynamicOrderSync>
<ngws:startDynamicOrderAsync><dyn:serviceRequest/><ngws:u
ser/></ngws:startDynamicOrderAsync>
</soapenv:Body></soapenv:Envelope>
```

3.4.2.2 Scale-out: Create a new VM (SCALE_OUT)

Scale-out process takes the policies nodes and creates as many instances of a VM as these policies determine.

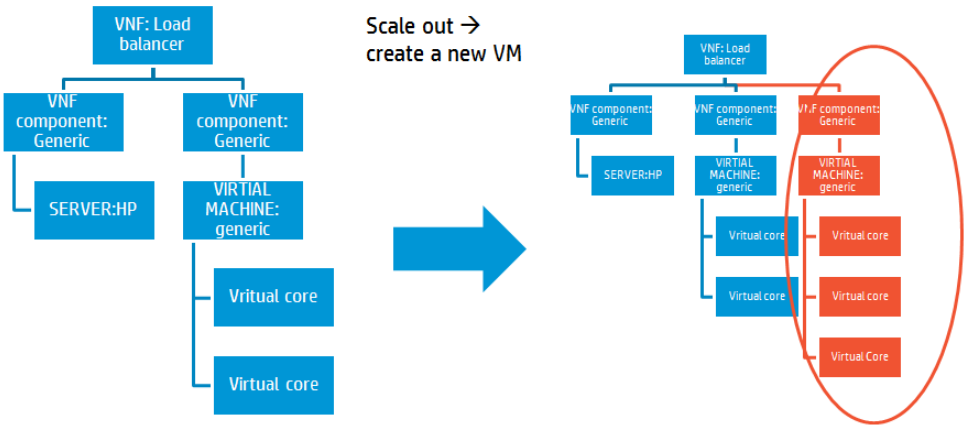


Figure 33 NBI: Scale-Out

The following is a typical Soap call for a scale–out process:

```
<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 type="NFVD" name="INVENTORY"
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>14032664806161</dyn:value>
                     </dyn:characteristic>
                  </dyn:characteristics>
               </dyn:service>
            </dyn:services>
         </dyn:serviceRequest>
         <ngws:user>foo</ngws:user>
      </ngws:startDynamicOrderSync>

<ngws:startDynamicOrderAsync><dyn:serviceRequest/><ngws:user/></ngws:st
artDynamicOrderAsync></soapenv:Body>
</soapenv:Envelope>
```

3.4.2.3 Scale-out: for Network Services

The scale out operation for Network services or VNF that contains the complete information of the Network that the different VMs will be connected is that. The idea is the VNF or NS will be scaled creating new VMs and these VMs will be attached to different networks (depends on VIM or tenant of the VM) and this operation will check if the network exist or the creation is needed and precede to do that.

The following is a typical Soap call for a scale–out process:

```
<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 type="NFVD" name="INVENTORY"
action="SCALE OUT">
               <dyn:service>
                  <dyn:name>NS|VNF</dyn:name>
                  <dyn:type>NFVD</dyn:type>
                  <dyn:action>SCALE_OUT_AND_CONNECT</dyn:action>
                  <dyn:characteristics>
                     <dyn:characteristic>
                        <dyn:name>ArtifactInstanceId</dyn:name>
                        <dyn:value>14032664806161</dyn:value>
                     </dyn:characteristic>
                  </dyn:characteristics>
               </dyn:service>
            </dyn:services>
         </dyn:serviceRequest>
         <ngws:user>foo</ngws:user>
      </ngws:startDynamicOrderSync>

<ngws:startDynamicOrderAsync><dyn:serviceRequest/><ngws:user/></ngws:st
artDynamicOrderAsync></soapenv:Body>
</soapenv:Envelope>
```

3.4.3 Monitoring actions

Next table represents all the actions related to Monitoring:

	Service Order Name	Service Order Service	Service Order Operation
5	NFVD	MONITOR	DEPLOY
6	NFVD	MONITOR	START
7	NFVD	MONITOR	STOP
8	NFVD	MONITOR	UNDEPLOY

Table 9 Monitoring actions

3.4.4 Network Service actions

Next table represents all the actions related to Network Services:

	Service Order Name	Service Order Service	Service Order Operation
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

Table 10 Network_Service actions

3.4.5 NS actions

Next table represents all the actions related to NS:

	Service Order Name	Service Order Service	Service Order Operation
14	NFVD	NS	CREATE_AND_CONNECT
15	NFVD	NS	SCALE_IN
16	NFVD	NS	SCALE_OUT_AND_CONNECT

Table 11 NS actions

3.4.6 Virtual Core actions

Next table represents all the actions related to Virtual Core:

	Service Order Name	Service Order Service	Service Order Operation
17	NFVD	VIRTUAL_CORE	SCALE_DOWN
18	NFVD	VIRTUAL_CORE	SCALE_UP

Table 12 Virtual Core actions

Next figure shows a graphical representation of Scale up action:

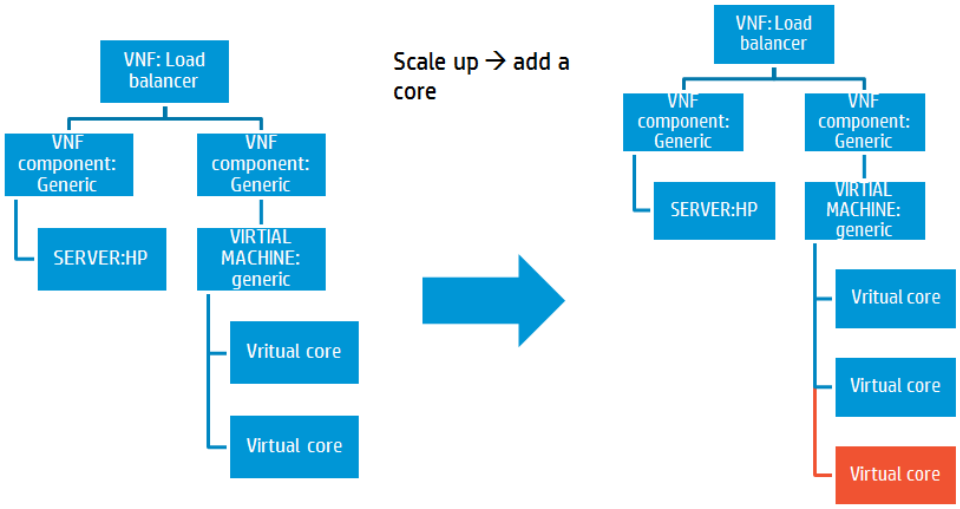


Figure 34 Scale up Virtual Core

3.4.7 Virtual Memory actions

Next table represents all the actions related to Virtual Memory:

	Service Order Name	Service Order Service	Service Order Operation
19	NFVD	VIRTUAL_MEMORY	SCALE_DOWN
20	NFVD	VIRTUAL_MEMORY	SCALE_UP

Table 13 Virtual Memory actions

3.4.8 VNF Manager actions

Next table represents all the actions related to VNF Manager:

	Service Order Name	Service Order Service	Service Order Operation

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 14 VNF Manager actions

3.4.9 Orchestration actions

3.4.9.1 VNF Orchestrator process (Create, assign and activate)

To create a new instance using a VNF template, assign resources, and deploy it over VIM, NFVD offer a complete operation inside Northbound Interface. This operation assumes that all elements inside template will be instantiated as new (The next section describes another complete orchestration with the re-use of existing networks instead of always creating a new one).

Parameters used:

- templateID: Template ID used to create VNF instance tree
- relationshipType: Type of relationship between Tenant and new VNF instance
- parentArtifactID: Instance ID of a Tenant that will contain a new VNF instance
- resourceTreeID: Instance ID of resource pool like a datacenter
- assignmentRelationshipID: Instance ID of assignment rules used to allocate resources

The following script displays a typical Soap call for complete VNF orchestration:

```
<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>resourceTreeID</ngws:name>
               <ngws:value>14019920442251</ngws:value>
            </ngws:param>
            <ngws:param>
               <ngws:name>assignmentRelationshipID</ngws:name>
               <ngws:value>14018960162001</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:inputParams>
         <ngws:user>?</ngws:user>
         <!--Optional:-->
         <ngws:userId>?</ngws:userId>
      </ngws:startServiceOrderAsync>
   </soapenv:Body>
</soapenv:Envelope>
```

3.4.9.2 VNF Orchestrator process with networking connection (Create, assign, connect networks, and activate)

If the VNF contains network and this network already exists on VIM, and you do not want to create a new network instance, NFVD exposes this orchestrator alternative.

There is only one difference with usual orchestrator process, instead of creating a new network often, subnetwork and IP address entities will be checked. If it exists, the VNF will use existing elements. If not, it will be created.

Parameters used:

- templateID: Template ID used to create VNF instance tree
- relationshipType: Type of relationship between Tenant and new VNF instance
- parentArtifactID: Instance ID of a Tenant that will contain a new VNF instance
- resourceTreeID: Instance ID of resource pool like a datacenter
- assignmentRelationshipID: Instance ID of assignment rules used to allocate resources

The following script displays a typical Soap call for complete VNF orchestration:

```
<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_AND_CONNECT</ngws:action>
         <!--Optional:-->
         <ngws:inputParams>
            <!--Zero or more repetitions:-->
            <ngws:param>
               <ngws:name>templateID</ngws:name>
               <ngws:value>vnf</ngws:value>
            </ngws:param>
            <ngws:param>
               <ngws:name>resourceTreeID</ngws:name>
               <ngws:value>14019920442251</ngws:value>
            </ngws:param>
```

```
            <ngws:param>
               <ngws:name>assignmentRelationshipID</ngws:name>
               <ngws:value>14018960162001</ngws:value>
            </ngws:param>
            <ngws:param>
               <ngws:name>parentArtifactId</ngws:name>
               <ngws:value>14133823926521</ngws:value>
            </ngws:param>
            <ngws:param>
               <ngws:name>relationshipType</ngws:name>
               <ngws:value>CONTAINS</ngws:value>
            </ngws:param>
         </ngws:inputParams>
         <ngws:user>hpsa</ngws:user>
      </ngws:startServiceOrderAsync>
   </soapenv:Body>
</soapenv:Envelope>
```

3.4.9.3 NS Orchestrator process

The operations over Network Services is very similar to creating a VNF with connect networks.

Parameters used:

- templateID: Template ID used to create VNF instance tree
- tenantID: Instance ID of a Tenant that will contain the VNFs instance of the NS
- resourceTreeID: Instance ID of resource pool like a datacenter
- assignmentRelationshipID: Instance ID of assignment rules used to allocate resources

The following script displays a typical Soap call for complete VNF orchestration:

```
<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>NS</ngws:name>
         <ngws:action>CREATE AND CONNECT</ngws:action>
         <!--Optional:-->
         <ngws:inputParams>
            <!--Zero or more repetitions:-->
            <ngws:param>
               <ngws:name>templateID</ngws:name>
               <ngws:value>TC1 NS</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:param>
               <ngws:name>tenantID</ngws:name>
```

```
                <ngws:value>14133823926521</ngws:value>
            </ngws:param>
        </ngws:inputParams>
        <ngws:user>hpsa</ngws:user>
    </ngws:startServiceOrderAsync>
  </soapenv:Body>
</soapenv:Envelope>
```

3.5 Alarms and notifications interface

Notifications will be generated from assurance gateway and these notifications are 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, topology, or both due to auto-scaling/update/upgrade/termination, switching to/from standby and so on.

In such cases, NFVD 3.0 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, Analytic tools and so on.

The consumer must use ActiveMQ connection factory and connect to a 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 the kind of messages that it is interested in.

The consumers can refer to the customfield in alarm to create such JMS Message Selectors.

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

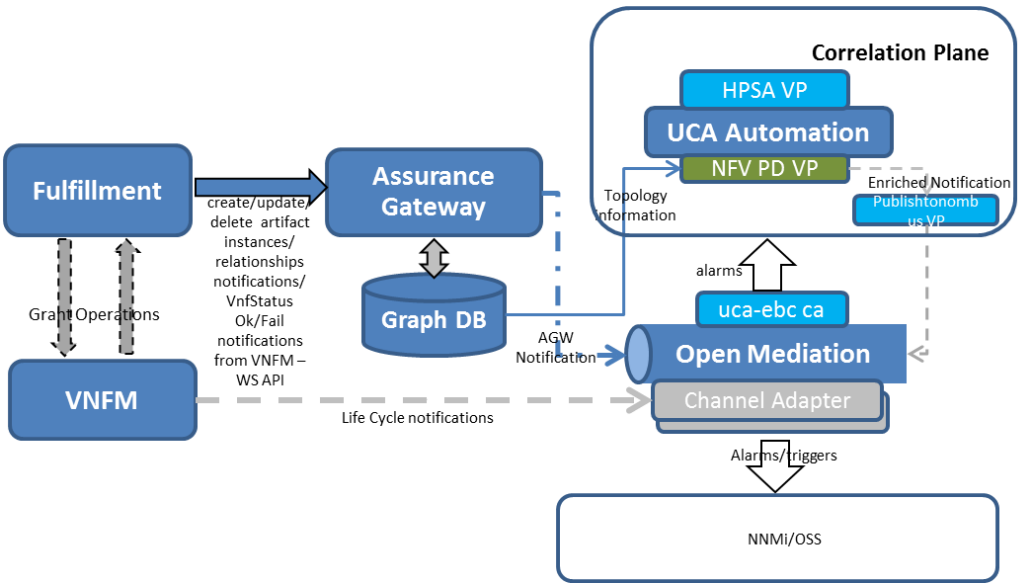


Figure 35 Notification flow diagram

Following are the different use cases for notification flow:

1. In case NFVD is an embedded VNFM, all VNFs including VDU components life cycle events such as create/update/delete operations, which include change in lifecycle state, and operational status to assurance, will generate enriched alarms respectively and publish to the OM Topic.

- Status field of an artifact will be referred for lifecycle status.

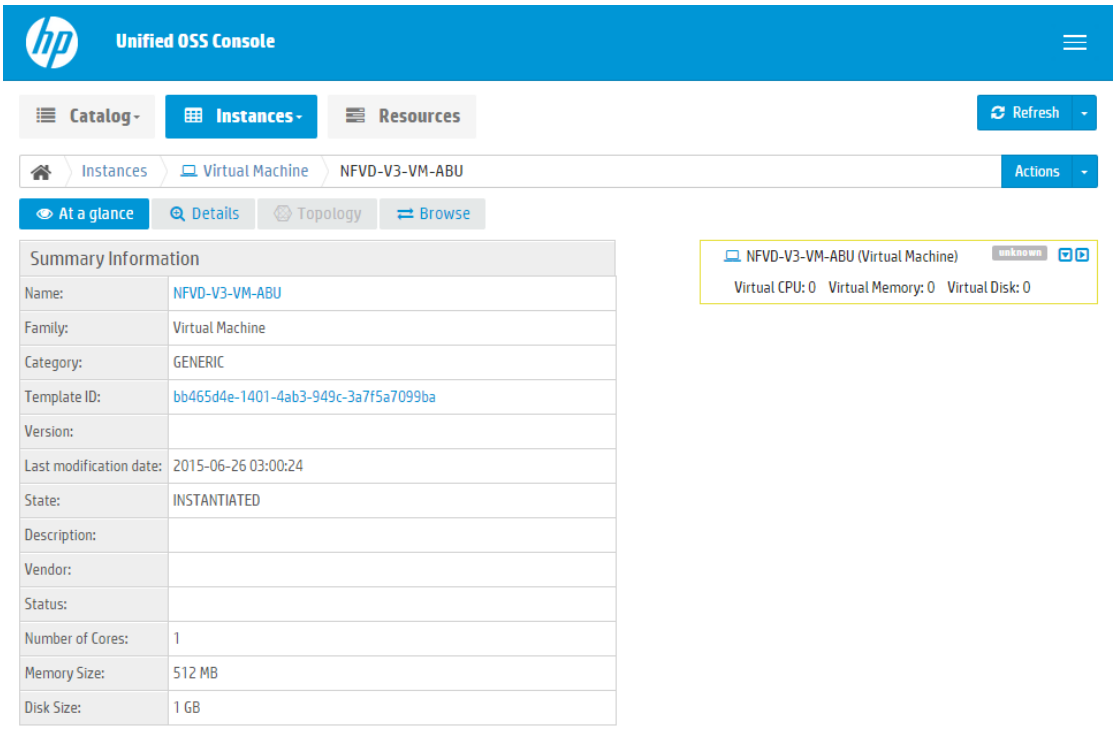


Figure 36 Artifact Status

2. In case of external VNFM, VNFM will request for grant permission from fulfillment for create or scale operations and will also post VNF status `Ok`, along with VNF status Fail notifications. Those notifications will be processed and sent to assurance, which will generate enriched alarms respectively and publish to the OM Topic.
3. In case, VNFM sends life cycle notifications as a trap or alarm, assurance can listen to those traps/alarms, provided, there is a channel adapter to convert the VNFM alarm to X733 alarm format, and in alarm, alarmName is LifeCycleStateChangeNotification, artifact ID field is available, the problem detection value pack will enrich the alarm and publish enriched alarm back to OM Bus for OSS.

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
originatingManagedEntityStructure	Full FQDN of the instance, starting from Network service
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	Status of the alarm with respect to problem raised by the alarm.
operatorState	Status of the alarm with respect to the operator
problemState	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: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
customField:source	Alarm generation source Internal: In case of state propagation alarm AGW: In case of life cycle and operational status alarm from VNFM.
customField:NOMType	Variable which can be used as a selector for fetching alarms, presently UCA-EBC CA requires OM Type.
customField:alarmSubtype	States the process 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	Life Cycle Notification event details received from VNFM

Table 15 Alarm field description

3.6 Access to NFV Director: Soap UI

As we described in previous chapter, SoapUI is a tool that allows you to play with NFV Director Data Model and Automation.

You can access to NFV Director Data Model using this tool and also execute all the available operations on the Automation.

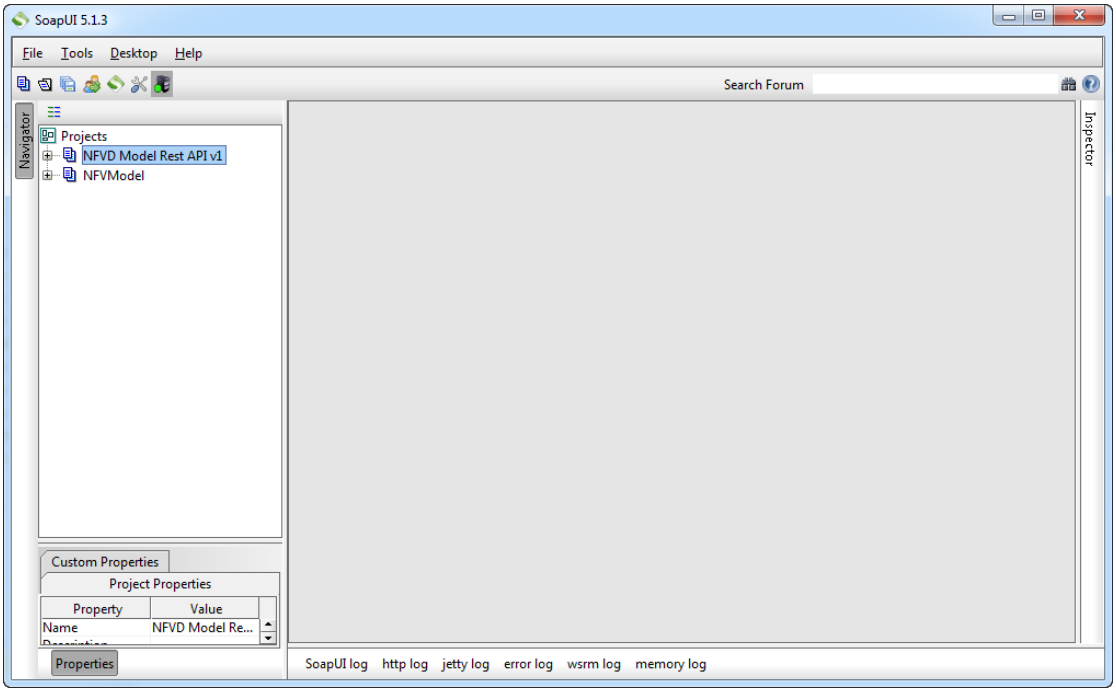


Figure 37 SoapUI

We can send instructions to HPSA via SOSA with an intuitive user interface and also, we can send commands via REST API.

Note

Read chapter *Northbound Interface* for more detailed info about NFV Director API.

The first step is to import an example project provided in the virtual machine.

This project is located at /opt/OV/ServiceActivator/solutions/NFVModel/docs/testing/NFVModel-SOSA-API-soapui-project.xml

Left click File > Import Project and search the location above.

3.6.1 Data Model

The REST API to access NFV Director Data Model is organized in three main groups corresponding to different artifacts present in NFV Director:

- Definition
- Template
- Instances

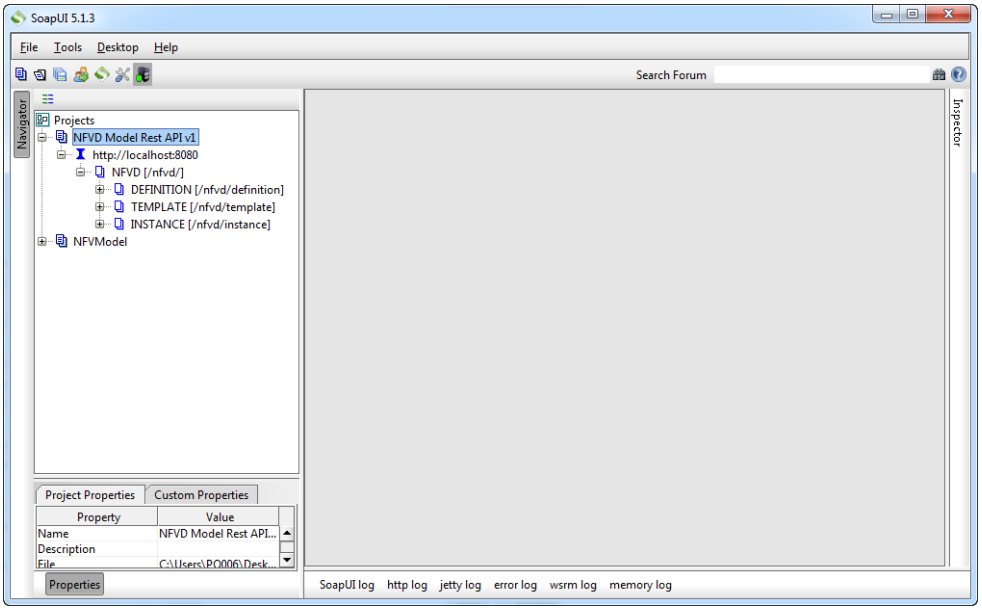


Figure 38 SoapUI: NFVD Model Rest API

Every action in NFV Director REST API is associated to corresponding REST protocol action. For example, to get all the definitions, we use the GET command on HTTP protocol using the URL corresponding to that functionality:

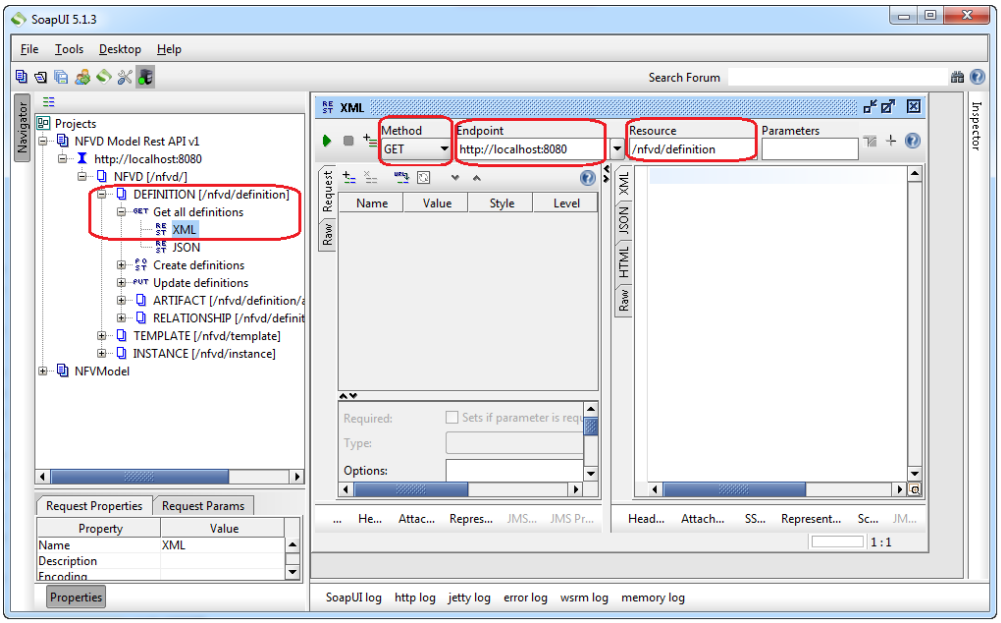


Figure 39 SoapUI: Get all definitions

You can send your requests (request body) both in XML format as JSON format.

The RAW request example sent in this case would be the next:

```
GET http://localhost:8080/nfvd/definition HTTP/1.1
Accept-Encoding: gzip,deflate
Accept: application/xml
Host: localhost:8080
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

If you request is correct, you will get a RAW Success response similar to this:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 11:18:40 GMT

<definitions xmlns="http://www.hp.com/nfvd">
   <artifact-definitions>
      <artifact-definition id="1009"
uri="/nfvd/definition/artifact/1009">
         <categories>
            <category>
               <attributes>
                  <attribute>
                     <label>Operational_Status</label>
                     <mandatory>false</mandatory>
                     <order>1</order>
                     <type>TEXT</type>
                     <unit>TEXT</unit>
                  </attribute>
                  <attribute>
                     <label>LifeCycle State Date</label>
                     <mandatory>false</mandatory>
                     <order>5</order>
                     <type>Date</type>
                     <unit>Date</unit>
                  </attribute>
               </attributes>
               <label>STATUS</label>
               <order>2</order>
            </category>
……————————
……————————
         <family>VIM</family>
         <physical>false</physical>
      </artifact-definition>
……————————————
……————————————
    </artifact-definitions>
    <relationship-definitions>
      <relationship-definition id="755"
uri="/nfvd/definition/relationship/755">
         <child-artifact-definition>
            <category>PROCESSING_ACTION</category>
            <family>POLICY</family>
         </child-artifact-definition>
         <enabled>true</enabled>
         <parent-artifact-definition>
            <category>PROCESSING_TREE</category>
            <family>POLICY</family>
         </parent-artifact-definition>
         </available-status>
         <type>APPLY</type>
      </relationship-definition>
……————————————
……————————————
   </relationship-definitions>
</definitions>
```

You have all the information you need for the NFV Director *REST API in the HP NFV Director v3.0 Rest API v1 - Reference guide.*

3.6.2 Automation

All the operations listed in previous section can be launched from SoapUI.

The first step is to import an example project provided in the virtual machine.

This project is located at /opt/OV/ServiceActivator/solutions/NFVAuto/docs/testing/NFVD-soapui-project.xml.

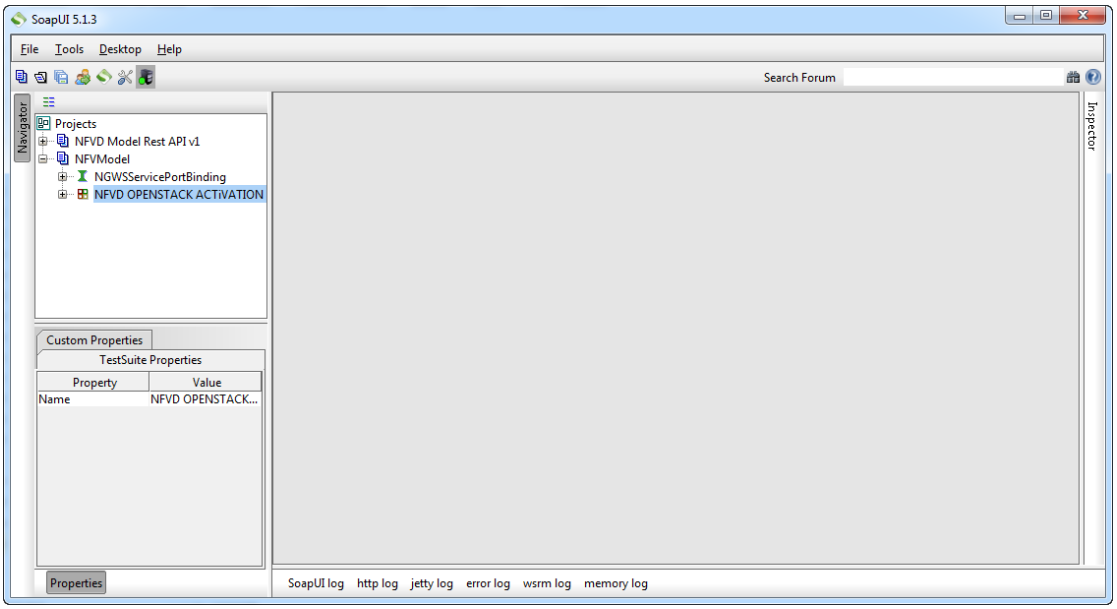


Figure 40 SoapUI: NFVD Automation

You can see many TestRunners created. There is one per possible operation to be done.

For example, to activate an VNF you will have to:

- open the Whole Process from the Test Case «OK». (Right click on «WHOLE PROCESS» > Open Editor / Double click on «WHOLE PROCESS»)

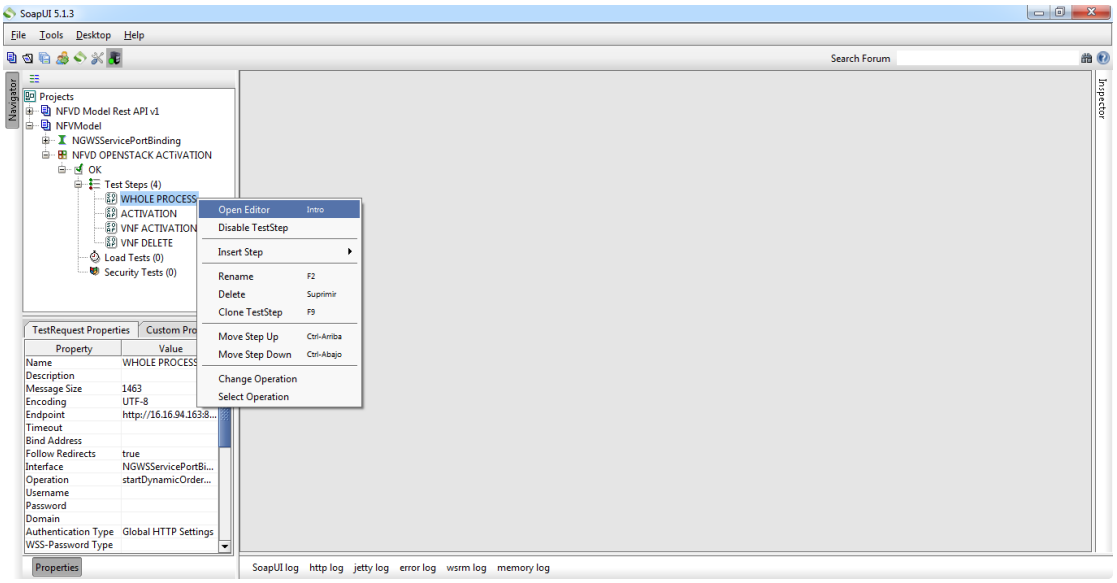


Figure 41 SoapUI: Activate VNF example

- click «Submit» button to run a default example

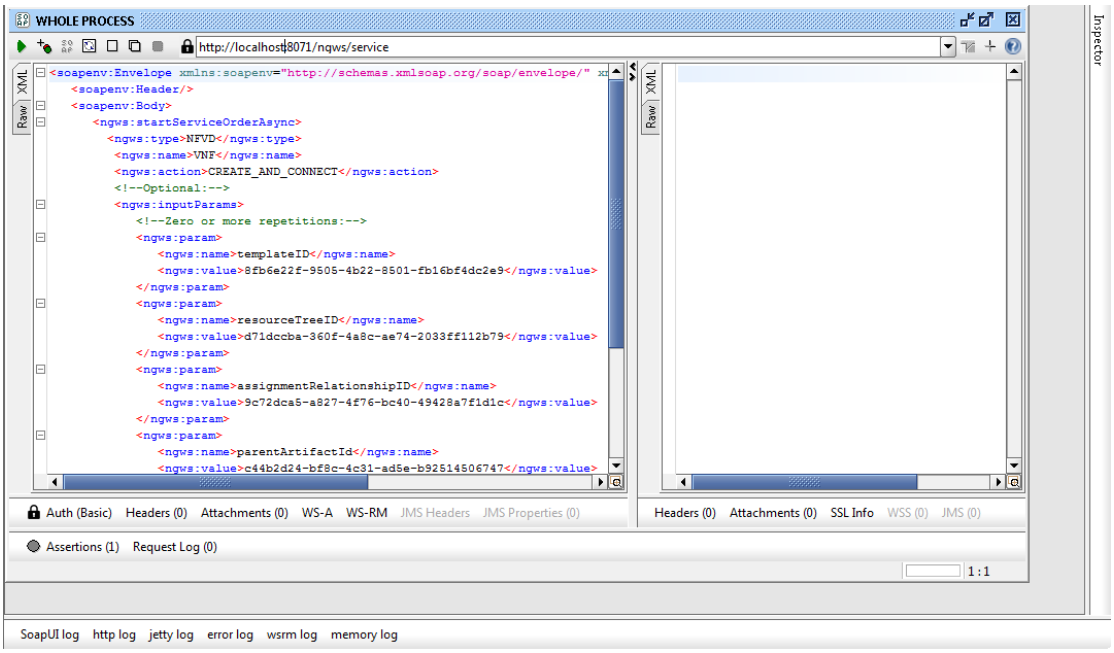


Figure 42 SoapUI: Activate VNF example (II)

Chapter 4

Southbound Interface

The high level view of HP NFV Director is showed in the following figure:

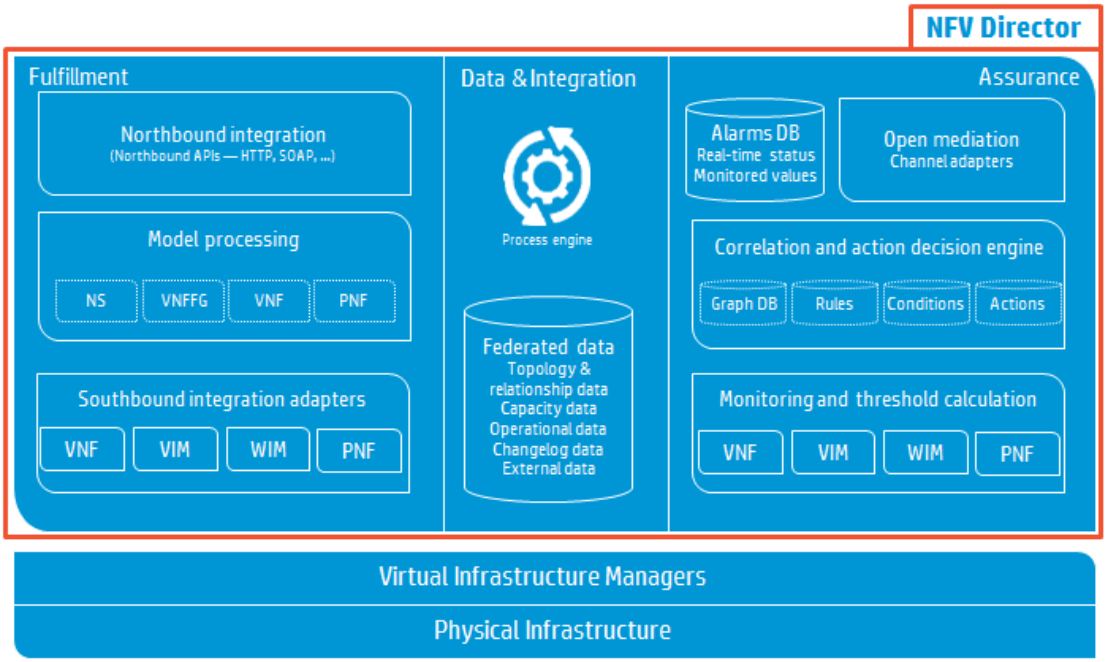


Figure 43 HP NFV Director Detailed Modules

The Southbound interface is the nearest layer to Virtual Infrastructure Managers in Fulfillment module.

It is composed by an Element Abstraction layer in which are situated an Error and Transaction Handler and a Template Manager.

This Element Abstraction is connected to different plug-ins (generic or vendor-specific) for full multi-vendor support (typical VIMs are, for example: Vmware, Openstack, Helion, CS8, ...).

Plug-ins may be as thick or thin as necessary to fill the gaps.

The following figure shows the Southbound interface in more detail.

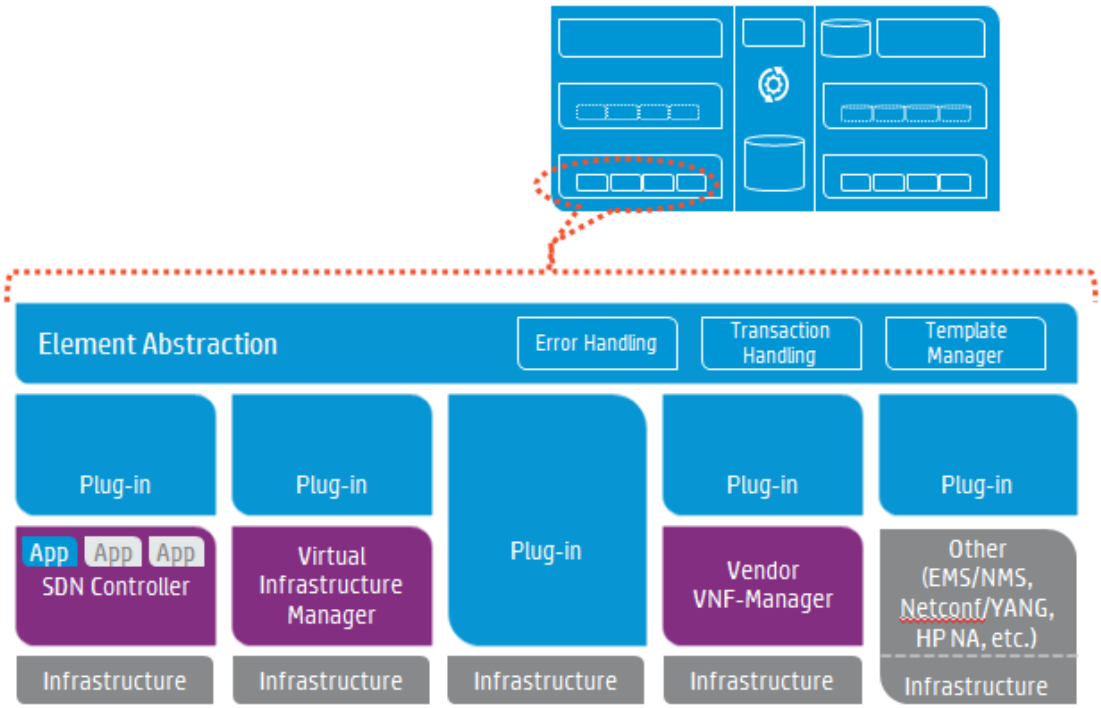


Figure 44 Southbound Interface - Detailed

NFV Director supports any VIM in the market through its telco grade proven plugin architecture.

Although not all the operations neither all the adapters are provided out of the box, extra operations and VIM types can be acquired as NFV Director plug-ins

The next table resumes the actual status of current VIMs.

VIM TYPE	VERSION	COMPUTE	NETWORKING	STORAGE	MONITORING	IMAGES
Openstack Compliant VIM	Previous to Havana	PLUGIN BASED	PLUGIN BASED	PLUGIN BASED	Not supported by the VIM	PLUGIN BASED
Openstack Compliant VIM	Havana	SUPPORTED	SUPPORTED	PLUGIN BASED	Not supported by the VIM	PLUGIN BASED
Openstack Compliant VIM	Icehouse	SUPPORTED	SUPPORTED	PLUGIN BASED	SUPPORTED	PLUGIN BASED
Openstack Compliant VIM	Juno	SUPPORTED	SUPPORTED	PLUGIN BASED	SUPPORTED	PLUGIN BASED
Non Openstack	any	PLUGIN BASED	PLUGIN BASED	PLUGIN BASED	PLUGIN BASED	PLUGIN BASED
vmware hypervisor	any	PLUGIN BASED	PLUGIN BASED	PLUGIN BASED	SUPPORTED	PLUGIN BASED
kvm hypervisor	any	PLUGIN BASED	PLUGIN BASED	PLUGIN BASED	SUPPORTED	PLUGIN BASED
dockers containers	any	PLUGIN BASED	PLUGIN BASED	PLUGIN BASED	PLUGIN BASED	Not supported by the Hypervisor

Table 16 VIMs status

Next sections show the command lists which are currently supported by NFV Director v3.0.

Titles in every section in this chapter follow the next nomenclature:

2.X VendorID # OSVersionGroupID # ElementTypeGroupID

For example, in section

4.2 HP # HP_Juno_v2 # HP_NEUTRON

That title corresponds to:

- VendorID: *HP*
- OSVersionID: *HP_Juno_v2*
- ElementTypeGroupID: *HP_NEUTRON*

4.1 GENERIC # VNFM # NFVO_TO_VNFM_v1

The following commands have all of them the next common data:

- ELEMENTTYPEGROUPID: VNFM
- OSVERSIONGROUPID: NFVO_TO_VNFM_v1

4.1.1 CHANGE_FLAVOR command

- Description: Change Flavor Request

4.1.1.1 JSON

```
"#if($JSON_REQUEST==""""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!=""""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR""
 #if($GRANT_APPROVAL_DESCRIPTOR!=""""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!=""""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.1.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}"
```

4.1.2 CREATE_VNF command

- Description: Create Operation for VNF

4.1.2.1 JSON

```
"#if($JSON_REQUEST==""""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
```

```
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!=""""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR""
 #if($GRANT_APPROVAL_DESCRIPTOR!=""""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!=""""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.2.2 REST

```
"http.operation=POST
 http.url.sufix=/v1/vnfdescriptor/${vnfdescriptorid}"
```

4.1.3 CREATE_VNF_DESCRIPTOR command

- Description: Create Operation for VNF Descriptor

4.1.3.1 JSON

```
"#if($JSON_REQUEST==""""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!=""""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR""
 #if($GRANT_APPROVAL_DESCRIPTOR!=""""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!=""""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.3.2 REST

```
"http.operation=POST
 http.url.sufix=/v1/vnfdescriptor"
```

4.1.4 DELETE_VNF command

- Description: Delete Operation for VNF

4.1.4.1 REST

```
"http.operation=DELETE
```

```
 http.url.sufix=/v1/vnf/${vnfid}"
```

4.1.5 GET_JOB_STATUS command

- Description: Get Job Status

4.1.5.1 REST

```
"http.operation=GET
 http.url.sufix=/v1/jobs/${jobid}"
```

4.1.6 GET_LIST_ORCHESTATOR_REGISTERED command

- Description: Get List Orchestator Registered

4.1.6.1 REST

```
"http.operation=GET
 http.url.sufix=/v1/vnfm/register"
```

4.1.7 GET_VNF_DESCRIPTOR_DETAILS command

- Description: Get VNF Descriptor Details

4.1.7.1 REST

```
"http.operation=GET
 http.url.sufix=/v1/vnfdescriptor/${vnfdescriptorID}"
```

4.1.8 GET_VNF_DESCRIPTOR_LIST command

- Description: Get VNF Descriptor List

4.1.8.1 REST

```
"http.operation=GET
 http.url.sufix=/v1/vnfdescriptor"
```

4.1.9 GET_VNF_DETAILS command

- Description: Get VNF Details

4.1.9.1 REST

```
"http.operation=GET
 http.url.sufix=/v1/vnf/${vnfid}"
```

4.1.10 GET_VNF_LIST command

- Description: Get VNF List

4.1.10.1 REST

```
"http.operation=GET
```

```
 http.url.sufix=/v1/vnf"
```

4.1.11 REGISTER_ORCHESTATOR command

- Description: Template for registration

4.1.11.1 JSON

```
"#if($JSON_REQUEST==""""")
 {
 ""nfvodomain"":""$NFVO_DOMAIN"",
 ""nfvoip"":""127.0.0.1"",
 ""Protocol"":""https"",
 ""Port"":""8766"",
 ""Credentials"":{
 ""username"":""$USERNAME"",
 ""password"":""$PASSWORD""
 }
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.11.2 REST

```
"http.operation=POST
 http.url.sufix=/v1/vnfm/register"
```

4.1.12 SCALE_DOWN_VDU command

- Description: Scale Down Operation for VDU

4.1.12.1 JSON

```
"#if($JSON_REQUEST==""""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.12.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/vdu/${vduid}/scale_down"
```

4.1.13 SCALE_DOWN_VM command

- Description: Scale Down Operation for VM

4.1.13.1 JSON

```
"#if($JSON_REQUEST==""""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.13.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/vdu/${vduid}/vm/${vmid}/scale_down"
```

4.1.14 SCALE_DOWN_VNF command

- Description: Scale Down Operation for VNF

4.1.14.1 JSON

```
"#if($JSON_REQUEST==""""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.14.2 REST

```
"http.operation=PUT
```

```
 http.url.sufix=/v1/vnf/${vnfid}/scale_down"
```

4.1.15 SCALE_IN_VDU command

- Description: Scale In Operation for VDU

4.1.15.1 JSON

```
"#if($JSON REQUEST=="""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF DETAILS DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT APPROVAL DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.15.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/vdu/${vduid}/scale_out"
```

4.1.16 SCALE_IN_VM command

- Description: Scale In Operation for VM

4.1.16.1 JSON

```
"#if($JSON REQUEST=="""")
 {
 ""vnfdescriptorid"":""$VNF DESCRIPTOR ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF DETAILS DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT APPROVAL DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.16.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/vdu/${vduid}/vm/${vmid}/scale_in"
```

4.1.17 SCALE_IN_VNFcommand

- Description: Scale In Operation for VNF

4.1.17.1 JSON

```
"#if($JSON_REQUEST==""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.17.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/scale_in"
```

4.1.18 SCALE_OUT_VDU command

- Description: Scale Out Operation for VDU

4.1.18.1 JSON

```
"#if($JSON_REQUEST==""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
```

```
 #end"
```

4.1.18.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/vdu/${vduid}/scale_out"
```

4.1.19 SCALE_OUT_VM command

- Description: Scale Out Operation for VM

4.1.19.1 JSON

```
"#if($JSON_REQUEST==""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.19.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/vdu/${vduid}/vm/${vmid}/scale_out"
```

4.1.20 SCALE_OUT_VNF command

- Description: Scale Out Operation for VNF

4.1.20.1 JSON

```
"#if($JSON_REQUEST==""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
```

```
 #else
 $JSON_REQUEST
 #end"
```

4.1.20.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/scale_out"
```

4.1.21 SCALE_UP_VDU command

- Description: Scale Up Operation for VDU

4.1.21.1 JSON

```
"#if($JSON_REQUEST==""""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.21.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/vdu/${vduid}/scale_up"
```

4.1.22 SCALE_UP_VM command

- Description: Scale Up Operation for VM

4.1.22.1 JSON

```
"#if($JSON_REQUEST==""""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!="""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!="""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
```

```
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.22.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/vdu/${vduid}/vm/${vmid}/scale_up"
```

4.1.23 SCALE_UP_VNF command

- Description: Scale Up Operation for VNF

4.1.23.1 JSON

```
"#if($JSON_REQUEST==""""")
 {
 ""vnfdescriptorid"":""$VNF_DESCRIPTOR_ID"",
 ""flavor"":""$FLAVOR"",
 ""vnfdescriptorversion"":""$VNF_DESCRIPTOR_VERSION"",
 #if($VNF_DETAILS_DESCRIPTOR!=""""")
 ""vnfDetailsDescriptor"":""$VNF_DETAILS_DESCRIPTOR"",
 #if($GRANT_APPROVAL_DESCRIPTOR!=""""")
 "",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!=""""")
 ""grantApprovalDescriptor"":""$GRANT_APPROVAL_DESCRIPTOR""
 #end
 }
 #else
 $JSON_REQUEST
 #end"
```

4.1.23.2 REST

```
"http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/scale_up"
```

4.1.24 UNREGISTER_ORCHESTATOR command

- Description: Template for unregistration

4.1.24.1 REST

```
"http.operation=DELETE
 http.url.sufix=/v1/vnfm/register/${vnfoid}"
```

4.2 HP # HP_Juno_v2 # HP_NEUTRON

4.2.1 ATTACH_VIP_SERVER command

- Description: Attach a Virtual IP

4.2.1.1 JSON

```
"{
 ""port"": {
 ""allowed_address_pairs"":[{""ip_address"":""${IP_ADDRESS}""}]
 }
 }"
```

4.2.1.2 REST

```
"http.operation=PUT
 http.url.sufix=/v2.0/ports/${PORT_ID_CREATED}
 openstack.endpointtype=network"
```

4.2.2 CREATE_NETWORK command

- Description: Create a Network HP Interface

4.2.2.1 JSON

```
"{
 ""network"":
 {
 ""name"": ""${NETWORK_NAME}""
 #if($NETWORK_ADMIN_STATE_UP)
 ,
 ""admin_state_up"": $NETWORK_ADMIN_STATE_UP
 #end
 #if($EXTERNAL)
 ,
 ""router:external"":$EXTERNAL
 #end

#if(($NETWORK_PROVIDER_PHYSICAL)&&($NETWORK_PROVIDER_TYPE)&&($NETWORK
PROVIDER_SEGMENTATION_ID))
 ,
 ""segments"":[
 {
 ""provider:segmentation_id"":$NETWORK_PROVIDER_SEGMENTATION_ID,
 ""provider:physical_network"":""$NETWORK_PROVIDER_PHYSICAL"",
 ""provider:network_type"":""$NETWORK_PROVIDER_TYPE""
 }
 ]
 #end
 #if($SHARED)
 ,
 ""shared"":$SHARED
 #end
 }
 }"
```

4.2.2.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/networks
 openstack.endpointtype=network"
```

4.2.3 CREATE_PORT command

- Description: Create port in a network HP Interface

4.2.3.1 JSON

```
"{
 ""port"":{
 ""network_id"":""${NETWORK_ID}"",
 ""name"":""${NAME}"",
 ""admin_state_up"":true,
 ""fixed_ips"":[
 {
 ""subnet_id"":""${SUBNETWORK_ID}""
 #if($IP_ADDRESS)
 ,
 ""ip_address"":""${IP_ADDRESS}""
 #end
 }
 ]
 #if($ARRAY_SG)
 ,
 ""security_groups"" : [${ARRAY_SG}]
 #end
 }
 }"
```

4.2.3.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/ports
 openstack.endpointtype=network"
```

CREATE_SG

4.2.4 CREATE_SG command

- Description: Create a SecurityGroup OpenStack Interface

4.2.4.1 JSON

```
"{
 ""security_group"":{
 ""name"":""${SG_NAME}"",
 ""description"":""${SG_DESCRIPTION}""
 }
 }"
```

4.2.4.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/security-groups
 openstack.endpointtype=network"
```

4.2.5 CREATE_SG_RULE command

- Description: Create a SecurityGroupRule OpenStack Interface

4.2.5.1 JSON

```
"{
 ""security_group_rule"":{
 ""direction"":""${DIRECTION}"",
 ""port_range_min"":""${PORT_RANGE_MIN}"",
 ""ethertype"":""${ETHERTYPE}"",
 ""port_range_max"":""${PORT_RANGE_MAX}"",
 ""protocol"":""${PROTOCOL}"",
 ""security_group_id"" : ""${SG_ID}""
 }
 }"
```

4.2.5.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/security-group-rules
 openstack.endpointtype=network"
```

4.2.6 CREATE_SUBNET command

- Description: Create a Subnet HP Interface

4.2.6.1 JSON

```
"{
 ""subnet"":{
 ""name"":""${SUBNET_NAME}"",
 ""network_id"":""${SUBNET_NETWORK_ID}"",
 #if($ENABLE_DHCP)
 ""enable_dhcp"":$ENABLE_DHCP,
 #end
 #if($GATEWAY_IP)
 ""gateway_ip"":""$GATEWAY_IP"",
 #end
 #if(($ALLOCATION_POOLS_START_IP)&&($ALLOCATION_POOLS_END_IP))
 ""allocation_pools"":[
 {
 ""end"":""$ALLOCATION_POOLS_END_IP"",
 ""start"":""$ALLOCATION_POOLS_START_IP""
 }
 ],
 #end
 ""ip_version"":${SUBNET_IP_VERSION},
 ""cidr"":""${SUBNET_CIDR}""
 }
 }"
```

4.2.6.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/subnets
 openstack.endpointtype=network
 "
```

4.2.7 CREATE_VIP command

- Description: Create a Virtual IP

4.2.7.1 JSON

```
"{
 ""port"": {
 ""network_id"": ""${NETWORK_ID}"",
 ""name"": ""${PORT_NAME}"",
 #if($SECURITY_GROUP_ID!="""")
 ""security_groups"":[""${SECURITY_GROUP_ID}""],
 #end
 ""admin_state_up"": true
 }
 }"
```

4.2.7.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/ports
 openstack.endpointtype=network"
```

4.2.8 DELETE_NETWORK command

- Description: Delete a Network HP Interface

4.2.8.1 REST

```
"http.operation=DELETE
 http.url.sufix=/v2.0/networks/${NETWORK_ID}
 openstack.endpointtype=network
 "
```

4.2.9 DELETE_SUBNET command

- Description: Delete a Subnet OpenStack Interface

4.2.9.1 REST

```
"http.operation=DELETE
 http.url.sufix=/v2.0/subnets/${SUBNET_ID}
 openstack.endpointtype=network
 "
```

4.2.10 QUERY_FLOATING_IP command

- Description: Query a Floating IP HP Interface

4.2.10.1 JSON

```
"{
 ""floatingip"": {
 ""floating_network_id"": ""${VAR_SUBNET_EXTERNAL_ID}"",
 ""port_id"": ""${PORT_ID}""
 }
 }"
```

4.2.10.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/floatingips
 openstack.endpointtype=network"
```

4.2.11 QUERY_NETWORK command

- Description: Query a Network HP Interface

4.2.11.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/networks/${NETWORK_ID}
 openstack.endpointtype=network
 "
```

4.2.12 QUERY_NETWORK_BY_PARAMS command

- Description: Query a Network by parameters HP Interface

4.2.12.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/networks
 openstack.endpointtype=network"
```

4.2.13 QUERY_PORT command

- Description: Query a Port for Floating IP HP Interface

4.2.13.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/ports
 openstack.endpointtype=network"
```

4.2.14 QUERY_SUBNET command

- Description: Query a Subnet HP Interface

4.2.14.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/subnets/${SUBNET_ID}
 openstack.endpointtype=network
 "
```

4.2.15 UPDATE_PORT_SECURITY_GROUP command

- Description: Update Port HP Interface

4.2.15.1 JSON

```
"{
 ""port"" :
 {
 ""security_groups"" : [${SG_ID}]
```

```
 }
 }"
```

4.2.15.2 REST

```
"http.operation=PUT
 http.url.sufix=/v2.0/ports/${PORT_ID}
 openstack.endpointtype=network"
```

4.2.16 UPDATE_SUBNET command

- Description: Update a Subnet HP Interface

4.2.16.1 JSON

```
"{
 ""subnet"":{
 ""name"":""${SUBNET_NAME}"",
 ""network_id"":""${SUBNET_NETWORK_ID}"",
 ""ip_version"": ${SUBNET_IP_VERSION},
 ""cidr"":""${SUBNET_CIDR}"",
 ""enable_dhcp"":""${SUBNET_ENABLE_DHCP}"",
 ""gateway_ip"":""${SUBNET_GATEWAY}""
 }
 }"
```

4.2.16.2 REST

```
"http.operation=PUT
 http.url.sufix=/v2.0/subnets/${SUBNET_ID}
 openstack.endpointtype=network
 "
```

4.3 HP # HP_Juno_v2 # HP_NOVA

4.3.1 CREATE_FLAVOR command

- Description: Create a Flavor HP Interface

4.3.1.1 JSON

```
"{
 ""flavor"": {
 ""name"": ""${FLAVOR_NAME}"",
 ""ram"": ${FLAVOR_RAM},
 ""vcpus"": ${FLAVOR_VCPUS},
 ""disk"": ${FLAVOR_DISK},
 ""id"": ""${FLAVOR_ID}"",
 ""os-flavor-access:is_public"": false
 }
 }"
```

4.3.1.2 REST

```
"http.operation=POST
 #http.url.sufix=/v2/${tenant_id}/servers
```

```
 http.url.sufix=/flavors
 openstack.endpointtype=compute
 "
```

4.3.2 CREATE_SERVER command

- Description: Create a Server HP Interface

4.3.2.1 JSON

```
"{
 ""server"":{
 ""name"":""${SERVER_NAME}"",
 ""imageRef"":""${IMAGE}"",
 ""flavorRef"":""${FLAVOR}"",
 ""max_count"":1,
 ""min_count"":1,
 #if(($HOST!=""""&&($AVAILABILITY_ZONE!=""""))
 ""availability_zone"":""$AVAILABILITY_ZONE:$HOST"",
 #end
 #if(($HOST==""""&&($AVAILABILITY_ZONE!=""""))
 ""availability_zone"":""$AVAILABILITY_ZONE"",
 #end
 ""networks"":[
 #set($INDEX=0)
 #foreach ($NETWORK_ID in $SERVER_NETWORK_ID)
 #if ($INDEX!=0)
 ,
 #end
 {""uuid"":""$NETWORK_ID""
 #set($INDEX2=0)
 #foreach ($PORTIP in $FIXED_IP_ARRAY)
 #if(($INDEX==$INDEX2)&&($PORTIP!=""DHCP"")&&($PORTIP!=""""))
 ,""port"":""$PORTIP""
 #end
 #set($INDEX2=$INDEX2 + 1)
 #end
 }
 #set($INDEX=$INDEX + 1)
 #end
 ],
 ""security_groups"":[
 {
 ""name"":""default""
 }
 ]
 }
 }"
```

4.3.2.2 REST

```
"http.operation=POST
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/servers
 openstack.endpointtype=compute"
```

4.3.3 DELETE_FLAVOR command

- Description: Delete a Flavor HP Interface

4.3.3.1 REST

```
"http.operation=DELETE
 http.url.sufix=/flavors/${FLAVOR_ID}
 openstack.endpointtype=compute"
```

4.3.4 DELETE_SERVER command

- Description: Delete a Server HP Interface

4.3.4.1 REST

```
"http.operation=DELETE
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/servers/${SERVER_ID}
 openstack.endpointtype=compute"
```

4.3.5 QUERY_FLAVOR command

- Description: Query a Flavor HP Interface

4.3.5.1 REST

```
"http.operation=GET
 http.url.sufix=/flavors/${FLAVOR_ID}
 openstack.endpointtype=compute
 "
```

QUERY_FLAVOR_BY_PARAMS

4.3.6 QUERY_FLAVOR_BY_PARAMS command

- Description: Query a Flavor by parameters HP Interface

4.3.6.1 REST

```
"http.operation=GET
 http.url.sufix=/flavors
 openstack.endpointtype=compute"
```

4.3.7 QUERY_FROM_SERVER command

- Description: Query FROM Server HP Interface

4.3.7.1 REST

```
"http.operation=GET
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/servers
 openstack.endpointtype=compute"
```

4.3.8 QUERY_IMAGE command

- Description: Query an Image HP Interface

4.3.8.1 REST

```
"http.operation=GET
 http.url.sufix=/images/${IMAGE_ID}
 openstack.endpointtype=compute
 "
```

4.3.9 QUERY_IMAGE_BY_PARAMS command

- Description: Query an Image by parameters HP Interface

4.3.9.1 REST

```
"http.operation=GET
 http.url.sufix=/images
 openstack.endpointtype=compute"
```

4.3.10 QUERY_SERVER command

- Description: Query a Server HP Interface

4.3.10.1 REST

```
"http.operation=GET
 #http.url.sufix=/v2/${tenant_id}/servers/${SERVER_ID}
 http.url.sufix=/servers/${SERVER_ID}
 openstack.endpointtype=compute"
```

4.3.11 SERVER_ACTIONS command

- Description: Template for server actions

4.3.11.1 REST

```
"{
 #if( ${VAR_OPERATION}==""RESIZE"")
 ""resize"": {""flavorRef"": ""${FLAVOR_ID}""}
 #end
 #if( ${VAR_OPERATION}==""CONFIRMRESIZE"")
 ""confirmResize"": null
 #end
 #if( ${VAR_OPERATION}==""REVERTRESIZE"")
 ""revertResize"": null
 #end
 #if( ${VAR_OPERATION}==""CHANGEPASSWORD"")
 ""changePassword"": {""adminPass"": ${PASSWORD}}
 #end
 #if( ${VAR_OPERATION}==""REBOOT"")
 ""reboot"": {""type"": ${TYPE}}
 #end
 #if( ${VAR_OPERATION}==""ADD_FLOATING_IP"")
 ""addFloatingIp"": {
 ""address"": ""${VAR_FLOATINGIPADDRESS}""
 }
 #end
 }"
```

4.3.11.2 REST

```
"http.operation=POST
 #http.url.sufix=/v2/${tenant_id}/servers/${SERVER_ID}
 http.url.sufix=/servers/${SERVER_ID}/action
 openstack.endpointtype=compute"
```

4.3.12 START_SERVER command

- Description: Starts a stopped Server and change its status to RUNNING

4.3.12.1 REST

```
"{
 ""os-start"": null
 }"
```

4.3.12.2 REST

```
"http.operation=POST
 http.url.sufix=/servers/${SERVER_ID}/action
 openstack.endpointtype=compute"
```

4.3.13 STOP_SERVER command

- Description: Stops a running Server and change its status to STOPPED

4.3.13.1 REST

```
"{
 ""os-stop"": null
 }"
```

4.3.13.2 REST

```
"http.operation=POST
 http.url.sufix=/servers/${SERVER_ID}/action
 openstack.endpointtype=compute"
```

4.3.14 UPDATE_SERVER command

- Description: Update a Server HP Interface

4.3.14.1 REST

```
"http.operation=PUT
 #http.url.sufix=/v2/${tenant_id}/servers/${SERVER_ID}
 http.url.sufix=/servers/${SERVER_ID}
 openstack.endpointtype=compute"
```

4.3.14.2 REST

```
"{
 ""server"" :
 {
```

```
 ""name"" : ""${SERVER NAME}""
 }
 }"
```

4.4 OpenStack # OS_Havana_v2 # OS_NEUTRON

4.4.1 ATTACH_VIP_SERVER command

- Description: Attach a Virtual IP

4.4.1.1 JSON

```
"{
 ""port"": {
 ""allowed_address_pairs"":[{""ip_address"":""${IP_ADDRESS}""}]
 }
 }"
```

4.4.1.2 REST

```
"http.operation=PUT
 http.url.sufix=/v2.0/ports/${PORT ID CREATED}
 openstack.endpointtype=network"
```

4.4.2 CREATE_NETWORK command

- Description: Create a Network OpenStack Interface

4.4.2.1 JSON

```
"{
 ""network"":
 {
 ""name"": ""${NETWORK_NAME}""
 #if($NETWORK ADMIN STATE UP)
 ,
 ""admin_state_up"": $NETWORK_ADMIN_STATE_UP
 #end

#if(($NETWORK PROVIDER PHYSICAL)&&($NETWORK PROVIDER TYPE)&&($NETWORK
PROVIDER SEGMENTATION ID))
 ,
 ""segments"":[
 {
 ""provider:segmentation_id"":$NETWORK_PROVIDER_SEGMENTATION_ID,
 ""provider:physical network"":""$NETWORK PROVIDER PHYSICAL"",
 ""provider:network_type"":""$NETWORK_PROVIDER_TYPE""
 }
 ]
 #end
 #if($SHARED)
 ,
 ""shared"":$SHARED
 #end
 }
 }"
```

4.4.2.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/networks
 openstack.endpointtype=network"
```

4.4.3 CREATE_SG command

- Description: Create a SecurityGroup OpenStack Interface

4.4.3.1 JSON

```
"{
 ""security_group"":{
 ""name"":""${SG_NAME}"",
 ""description"":""${SG_DESCRIPTION}""
 }
 }"
```

4.4.3.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/security-groups
 openstack.endpointtype=network"
```

4.4.4 CREATE_SG_RULE command

- Description: Create a SecurityGroupRule OpenStack Interface

4.4.4.1 JSON

```
"{
 ""security_group_rule"":{
 ""direction"":""${DIRECTION}"",
 ""port_range_min"":""${PORT_RANGE_MIN}"",
 ""ethertype"":""${ETHERTYPE}"",
 ""port_range_max"":""${PORT_RANGE_MAX}"",
 ""protocol"":""${PROTOCOL}"",
 ""security_group_id"" : ""${SG_ID}""
 }
 }"
```

4.4.4.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/security-group-rules
 openstack.endpointtype=network"
```

4.4.5 CREATE_SUBNET command

- Description: Create a Subnet OpenStack Interface

4.4.5.1 JSON

```
"{
 ""subnet"":{
```

```
 ""name"":""${SUBNET_NAME}"",
 ""network_id"":""${SUBNET_NETWORK_ID}"",
 #if($ENABLE_DHCP)
 ""enable_dhcp"":$ENABLE_DHCP,
 #end
 #if($GATEWAY_IP)
 ""gateway_ip"":""$GATEWAY_IP"",
 #end
 #if(($ALLOCATION_POOLS_START_IP)&&($ALLOCATION_POOLS_END_IP))
 ""allocation_pools"":[
 {
 ""end"":""$ALLOCATION_POOLS_END_IP"",
 ""start"":""$ALLOCATION_POOLS_START_IP""
 }
 ],
 #end
 ""ip_version"":${SUBNET_IP_VERSION},
 ""cidr"":""${SUBNET_CIDR}""
 }
 }"
```

4.4.5.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/subnets
 openstack.endpointtype=network
 "
```

4.4.6 CREATE_VIP command

- Description: Create a Virtual IP

4.4.6.1 JSON

```
"{
 ""port"": {
 ""network_id"": ""${NETWORK_ID}"",
 ""name"": ""${PORT_NAME}"",
 #if($SECURITY_GROUP_ID!="""")
 ""security_groups"":[""${SECURITY_GROUP_ID}""],
 #end
 ""admin_state_up"": true
 }
 }"
```

4.4.6.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/ports
 openstack.endpointtype=network"
```

4.4.7 DELETE_NETWORK command

- Description: Delete a Network OpenStack Interface

4.4.7.1 REST

```
"http.operation=DELETE
```

```
 http.url.sufix=/v2.0/networks/${NETWORK ID}
 openstack.endpointtype=network
 "
```

4.4.8 DELETE_SUBNET command

- Description: Delete a Subnet OpenStack Interface

4.4.8.1 REST

```
"http.operation=DELETE
 http.url.sufix=/v2.0/subnets/${SUBNET ID}
 openstack.endpointtype=network
 "
```

4.4.9 QUERY_FLOATING_IP command

- Description: Query a Floating IP OpenStack Interface

4.4.9.1 JSON

```
"{
 ""floatingip"": {
 ""floating network id"": ""${VAR SUBNET EXTERNAL ID}"",
 ""port_id"": ""${PORT_ID}""
 }
 }"
```

4.4.9.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/floatingips
 OpenStack.endpointtype=network"
```

4.4.10 QUERY_NETWORK command

- Description: Query a Network OpenStack Interface

4.4.10.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/networks/${NETWORK_ID}
 openstack.endpointtype=network
 "
```

4.4.11 QUERY_NETWORK_BY_PARAMS command

- Description: Query a Network by parameters OpenStack Interface

4.4.11.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/networks
 openstack.endpointtype=network"
```

4.4.12 QUERY_PORT command

- Description: Query a Port for Floating IP OpenStack Interface

4.4.12.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/ports
 OpenStack.endpointtype=network"
```

4.4.13 QUERY_SUBNET command

- Description: Query a Subnet OpenStack Interface

4.4.13.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/subnets/${SUBNET_ID}
 openstack.endpointtype=network
 "
```

4.4.14 UPDATE_PORT_SECURITY_GROUP command

- Description: Update Port OpenStack Interface

4.4.14.1 JSON

```
"{
 ""port"" :
 {
 ""security_groups"" : [${SG_ID}]
 }
 }"
```

4.4.14.2 REST

```
"http.operation=PUT
 http.url.sufix=/v2.0/ports/${PORT_ID}
 openstack.endpointtype=network"
```

4.4.15 UPDATE_SUBNET command

- Description: Update a Subnet OpenStack Interface

4.4.15.1 JSON

```
"{
 ""subnet"":{
 ""name"":""${SUBNET_NAME}"",
 ""network_id"":""${SUBNET_NETWORK_ID}"",
 ""ip_version"": ${SUBNET_IP_VERSION},
 ""cidr"":""${SUBNET_CIDR}"",
 ""enable_dhcp"":""${SUBNET_ENABLE_DHCP}"",
 ""gateway_ip"":""${SUBNET_GATEWAY}""
 }
 }"
```

4.4.15.2 REST

```
"http.operation=PUT
 http.url.sufix=/v2.0/subnets/${SUBNET_ID}
 openstack.endpointtype=network
 "
```

4.5 OpenStack # OS_Havana_v2 # OS_NOVA

4.5.1 CREATE_FLAVOR command

- Description: Create a Flavor OpenStack Interface

4.5.1.1 JSON

```
"{
 ""flavor"": {
 ""name"": ""${FLAVOR_NAME}"",
 ""ram"": ${FLAVOR_RAM},
 ""vcpus"": ${FLAVOR_VCPUS},
 ""disk"": ${FLAVOR_DISK},
 ""id"": ""${FLAVOR_ID}"",
 ""os-flavor-access:is_public"": false
 }
 }"
```

4.5.1.2 REST

```
"http.operation=POST
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/flavors
 openstack.endpointtype=compute
 "
```

4.5.2 CREATE_SERVER command

- Description: Create a Server OpenStack Interface

4.5.2.1 JSON

```
"{
 ""server"":{
 ""name"":""${SERVER_NAME}"",
 ""imageRef"":""${IMAGE}"",
 ""flavorRef"":""${FLAVOR}"",
 ""max_count"":1,
 ""min_count"":1,
 #if(($HOST!="""")&&($AVAILABILITY_ZONE!=""""))
 ""availability_zone"":""$AVAILABILITY_ZONE:$HOST"",
 #end
 #if(($HOST==""""&&($AVAILABILITY_ZONE!=""""))
 ""availability_zone"":""$AVAILABILITY_ZONE"",
 #end
 ""networks"":[
 #set($INDEX=0)
 #foreach ($NETWORK_ID in $SERVER_NETWORK_ID)
 #if ($INDEX!=0)
```

```
,
#end
{""uuid"":""$NETWORK_ID""
#set($INDEX2=0)
#foreach ($PORTIP in $FIXED_IP_ARRAY)
#if(($INDEX==$INDEX2)&&($PORTIP!=""DHCP"")&&($PORTIP!=""""))
,""fixed ip"":""$PORTIP""
#end
#set($INDEX2=$INDEX2 + 1)
#end
}
#set($INDEX=$INDEX + 1)
#end
],
""security_groups"":[
{
""name"":""default""
}
]
}
}"
```

4.5.2.2 REST

```
"http.operation=POST
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/servers
 openstack.endpointtype=compute"
```

4.5.3 DELETE_FLAVOR command

- Description: Delete a Flavor OpenStack Interface

4.5.3.1 REST

```
"http.operation=DELETE
 http.url.sufix=/flavors/${FLAVOR_ID}
 openstack.endpointtype=compute"
```

4.5.4 DELETE_SERVER command

- Description: Delete a Server OpenStack Interface

4.5.4.1 REST

```
"http.operation=DELETE
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/servers/${SERVER_ID}
 openstack.endpointtype=compute"
```

4.5.5 QUERY_FLAVOR command

- Description: Query a Flavor OpenStack Interface

4.5.5.1 REST

```
"http.operation=GET
 http.url.sufix=/flavors/${FLAVOR_ID}
```

```
 openstack.endpointtype=compute
 "
```

4.5.6 QUERY_FLAVOR_BY_PARAMS command

- Description: Query a Flavor by parameters OpenStack Interface

4.5.6.1 REST

```
"http.operation=GET
 http.url.sufix=/flavors
 openstack.endpointtype=compute"
```

4.5.7 QUERY_FROM_SERVER command

- Description: Query FROM Server OpenStack Interface

4.5.7.1 REST

```
"http.operation=GET
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/servers
 openstack.endpointtype=compute""""
```

4.5.8 QUERY_IMAGE command

- Description: Query an Image OpenStack Interface

4.5.8.1 REST

```
"http.operation=GET
 http.url.sufix=/images/${IMAGE_ID}
 openstack.endpointtype=compute
 "
```

4.5.9 QUERY_IMAGE_BY_PARAMS command

- Description: Query an Image by parameters OpenStack Interface

4.5.9.1 REST

```
"http.operation=GET
 http.url.sufix=/images
 openstack.endpointtype=compute"
```

4.5.10 QUERY_SERVER command

- Description: Query a Server OpenStack Interface

4.5.10.1 REST

```
"http.operation=GET
 #http.url.sufix=/v2/${tenant_id}/servers/${SERVER_ID}
 http.url.sufix=/servers/${SERVER_ID}
 openstack.endpointtype=compute"
```

4.5.11 SERVER_ACTIONS command

- Description: Template for server actions

4.5.11.1 JSON

```
"{
 #if( ${VAR_OPERATION}==""RESIZE"")
 ""resize"": {""flavorRef"": ""${FLAVOR_ID}""}
 #end
 #if( ${VAR_OPERATION}==""CONFIRMRESIZE"")
 ""confirmResize"": null
 #end
 #if( ${VAR_OPERATION}==""REVERTRESIZE"")
 ""revertResize"": null
 #end
 #if( ${VAR_OPERATION}==""CHANGEPASSWORD"")
 ""changePassword"": {""adminPass"": ${PASSWORD}}
 #end
 #if( ${VAR_OPERATION}==""REBOOT"")
 ""reboot"": {""type"": ${TYPE}}
 #end
 #if( ${VAR_OPERATION}==""ADD_FLOATING_IP"")
 ""addFloatingIp"": {
 ""address"": ""${VAR_FLOATINGIPADDRESS}""
 }
 #end
 }"
```

4.5.11.2 REST

```
"http.operation=POST
 #http.url.sufix=/v2/${tenant_id}/servers/${SERVER_ID}
 http.url.sufix=/servers/${SERVER_ID}/action
 openstack.endpointtype=compute"
```

4.5.12 START_SERVER command

- Description: Starts a stopped Server and change its status to RUNNING

4.5.12.1 JSON

```
"{
 ""os-start"": null
 }"
```

4.5.12.2 REST

```
"http.operation=POST
 http.url.sufix=/servers/${SERVER_ID}/action
 openstack.endpointtype=compute"
```

4.5.13 STOP_SERVER command

- Description: Stops a running Server and change its status to STOPPED

4.5.13.1 JSON

```
"{
 ""os-start"": null
 }"
```

4.5.13.2 REST

```
"http.operation=POST
 http.url.sufix=/servers/${SERVER_ID}/action
 openstack.endpointtype=compute"
```

4.5.14 UPDATE_SERVER command

- Description: Update a Server OpenStack Interface

4.5.14.1 JSON

```
"{
 ""server"" :
 {
 ""name"" : ""${SERVER_NAME}""
 }
 }"
```

4.5.14.2 REST

```
"http.operation=PUT
 #http.url.sufix=/v2/${tenant_id}/servers/${SERVER_ID}
 http.url.sufix=/servers/${SERVER_ID}
 openstack.endpointtype=compute"
```

4.6 OpenStack # OS_Icehoust_v2 # OS_NEUTRON

4.6.1 ATTACH_VIP_SERVER command

- Description: Attach a Virtual IP

4.6.1.1 JSON

```
"{
 ""port"": {
 ""allowed_address_pairs"":[{""ip_address"":""${IP_ADDRESS}""}]
 }
 }"
```

4.6.1.2 REST

```
"http.operation=PUT
 http.url.sufix=/v2.0/ports/${PORT_ID_CREATED}
 openstack.endpointtype=network"
```

4.6.2 CREATE_NETWORK command

- Description: Create a Network OpenStack Interface

4.6.2.1 JSON

```
"{
 ""network"":
 {
 ""name"": ""${NETWORK_NAME}""
 #if($NETWORK_ADMIN_STATE_UP)
 ,
 ""admin_state_up"": $NETWORK_ADMIN_STATE_UP
 #end
 #if($EXTERNAL)
 ,
 ""router:external"":$EXTERNAL
 #end

#if(($NETWORK_PROVIDER_PHYSICAL)&&($NETWORK_PROVIDER_TYPE)&&($NETWORK_
PROVIDER_SEGMENTATION_ID))
 ,
 ""segments"":[
 {
 ""provider:segmentation_id"":$NETWORK_PROVIDER_SEGMENTATION_ID,
 ""provider:physical_network"":""$NETWORK_PROVIDER_PHYSICAL"",
 ""provider:network_type"":""$NETWORK_PROVIDER_TYPE""
 }
 ]
 #end
 #if($SHARED)
 ,
 ""shared"":$SHARED
 #end
 }
 }"
```

4.6.2.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/networks
 openstack.endpointtype=network"
```

4.6.3 CREATE_SG command

- Description: Create a SecurityGroup OpenStack Interface

4.6.3.1 JSON

```
"{
 ""security_group"":{
 ""name"":""${SG_NAME}"",
 ""description"":""${SG_DESCRIPTION}""
 }
 }"
```

4.6.3.2 REST

```
"http.operation=POST
```

```
 http.url.sufix=/v2.0/security-groups
 openstack.endpointtype=network"
```

4.6.4 CREATE_SG_RULE command

- Description: Create a SecurityGroupRule OpenStack Interface

4.6.4.1 JSON

```
"{
 ""security_group_rule"":{
 ""direction"":""${DIRECTION}"",
 ""port_range_min"":""${PORT_RANGE_MIN}"",
 ""ethertype"":""${ETHERTYPE}"",
 ""port_range_max"":""${PORT_RANGE_MAX}"",
 ""protocol"":""${PROTOCOL}"",
 ""security_group_id"" : ""${SG_ID}""
 }
 }"
```

4.6.4.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/security-group-rules
 openstack.endpointtype=network"
```

4.6.5 CREATE_SUBNET command

- Description: Create a Subnet OpenStack Interface

4.6.5.1 JSON

```
"{
 ""subnet"":{
 ""name"":""${SUBNET_NAME}"",
 ""network_id"":""${SUBNET_NETWORK_ID}"",
 #if($ENABLE_DHCP)
 ""enable_dhcp"":$ENABLE_DHCP,
 #end
 #if($GATEWAY_IP)
 ""gateway_ip"":""$GATEWAY_IP"",
 #end
 #if(($ALLOCATION_POOLS_START_IP)&&($ALLOCATION_POOLS_END_IP))
 ""allocation_pools"":[
 {
 ""end"":""$ALLOCATION_POOLS_END_IP"",
 ""start"":""$ALLOCATION_POOLS_START_IP""
 }
 ],
 #end
 ""ip_version"":${SUBNET_IP_VERSION},
 ""cidr"":""${SUBNET_CIDR}""
 }
 }"
```

4.6.5.2 REST

```
"http.operation=POST
```

```
 http.url.sufix=/v2.0/subnets
 openstack.endpointtype=network
 "
```

4.6.6 CREATE_VIP command

- Description: Create a Virtual IP

4.6.6.1 JSON

```
"{
 ""port"": {
 ""network_id"": ""${NETWORK_ID}"",
 ""name"": ""${PORT_NAME}"",
 #if($SECURITY_GROUP_ID!="""")
 ""security_groups"":[""${SECURITY_GROUP_ID}""],
 #end
 ""admin_state_up"": true
 }
 }"
```

4.6.6.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/ports
 openstack.endpointtype=network"
```

4.6.7 DELETE_NETWORK command

- Description: Delete a Network OpenStack Interface

4.6.7.1 REST

```
"http.operation=DELETE
 http.url.sufix=/v2.0/networks/${NETWORK_ID}
 openstack.endpointtype=network
 "
```

4.6.8 DELETE_SUBNET command

- Description: Delete a Subnet OpenStack Interface

4.6.8.1 REST

```
"http.operation=DELETE
 http.url.sufix=/v2.0/subnets/${SUBNET_ID}
 openstack.endpointtype=network
 "
```

4.6.9 QUERY_FLOATING_IP command

- Description: Query a Floating IP OpenStack Interface

4.6.9.1 JSON

```
"{
```

```
 ""floatingip"": {
 ""floating_network_id"": ""${VAR_SUBNET_EXTERNAL_ID}"",
 ""port_id"": ""${PORT_ID}""
 }
 }"
```

4.6.9.2 REST

```
"http.operation=POST
 http.url.sufix=/v2.0/floatingips
 OpenStack.endpointtype=network"
```

4.6.10 QUERY_NETWORK command

- Description: Query a Network OpenStack Interface

4.6.10.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/networks/${NETWORK_ID}
 openstack.endpointtype=network
 "
```

4.6.11 QUERY_NETWORK_BY_PARAMS command

- Description: Query a Network by parameters OpenStack Interface

4.6.11.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/networks
 openstack.endpointtype=network "
```

4.6.12 QUERY_PORT command

- Description: Query a Port for Floating IP OpenStack Interface

4.6.12.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/ports
 OpenStack.endpointtype=network"
```

4.6.13 QUERY_SUBNET command

- Description: Query a Subnet OpenStack Interface

4.6.13.1 REST

```
"http.operation=GET
 http.url.sufix=/v2.0/subnets/${SUBNET_ID}
 openstack.endpointtype=network
 "
```

4.6.14 UPDATE_PORT_SECURITY_GROUP command

- Description: Update Port OpenStack Interface

4.6.14.1 JSON

```
"{
 ""port"" :
 {
 ""security_groups"" : [${SG_ID}]
 }
 }"
```

4.6.14.2 REST

```
"http.operation=PUT
 http.url.sufix=/v2.0/ports/${PORT_ID}
 openstack.endpointtype=network"
```

4.6.15 UPDATE_SUBNET command

- Description: Update a Subnet OpenStack Interface

4.6.15.1 JSON

```
"{
 ""subnet"":{
 ""name"":""${SUBNET_NAME}"",
 ""network_id"":""${SUBNET_NETWORK_ID}"",
 ""ip_version"": ${SUBNET_IP_VERSION},
 ""cidr"":""${SUBNET_CIDR}"",
 ""enable_dhcp"":""${SUBNET_ENABLE_DHCP}"",
 ""gateway_ip"":""${SUBNET_GATEWAY}""
 }
 }"
```

4.6.15.2 REST

```
"http.operation=PUT
 http.url.sufix=/v2.0/subnets/${SUBNET_ID}
 openstack.endpointtype=network
 "
```

4.7 OpenStack # OS_Icehouse_v2 # OS_NOVA

4.7.1 CREATE_FLAVOR command

- Description: Create a Flavor OpenStack Interface

4.7.1.1 JSON

```
"{
 ""flavor"": {
 ""name"": ""${FLAVOR_NAME}"",
 ""ram"": ${FLAVOR_RAM},
 ""vcpus"": ${FLAVOR_VCPUS},
```

```
 ""disk"": ${FLAVOR_DISK},
 ""id"": ""${FLAVOR_ID}"",
 ""os-flavor-access:is_public"": false
 }
 }"
```

4.7.1.2 REST

```
"http.operation=POST
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/flavors
 openstack.endpointtype=compute
 "
```

4.7.2 CREATE_SERVER command

- Description: Create a Server OpenStack Interface

4.7.2.1 JSON

```
"{
 ""server"":{
 ""name"":""${SERVER_NAME}"",
 ""imageRef"":""${IMAGE}"",
 ""flavorRef"":""${FLAVOR}"",
 ""max_count"":1,
 ""min_count"":1,
 #if(($HOST!=""""&&($AVAILABILITY_ZONE!=""""))
 ""availability_zone"":""$AVAILABILITY_ZONE:$HOST"",
 #end
 #if(($HOST==""""&&($AVAILABILITY_ZONE!=""""))
 ""availability_zone"":""$AVAILABILITY_ZONE"",
 #end
 ""networks"":[
 #set($INDEX=0)
 #foreach ($NETWORK_ID in $SERVER_NETWORK_ID)
 #if ($INDEX!=0)
 ,
 #end
 {""uuid"":""$NETWORK_ID""
 #set($INDEX2=0)
 #foreach ($PORTIP in $FIXED_IP_ARRAY)
 #if(($INDEX==$INDEX2)&&($PORTIP!=""DHCP"")&&($PORTIP!=""""))
 ,""fixed_ip"":""$PORTIP""
 #end
 #set($INDEX2=$INDEX2 + 1)
 #end
 }
 #set($INDEX=$INDEX + 1)
 #end
 ],
 ""security_groups"":[
 {
 ""name"":""default""
 }
 ]
 }
 }"
```

4.7.2.2 REST

```
"http.operation=POST
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/servers
 openstack.endpointtype=compute"
```

4.7.3 DELETE_FLAVOR command

- Description: Delete a Flavor OpenStack Interface

4.7.3.1 REST

```
"http.operation=DELETE
 http.url.sufix=/flavors/${FLAVOR_ID}
 openstack.endpointtype=compute"
```

4.7.4 DELETE_SERVER command

- Description: Delete a Server OpenStack Interface

4.7.4.1 REST

```
"http.operation=DELETE
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/servers/${SERVER_ID}
 openstack.endpointtype=compute"
```

4.7.5 QUERY_FLAVOR command

- Description: Query a Flavor OpenStack Interface

4.7.5.1 REST

```
"http.operation=GET
 http.url.sufix=/flavors/${FLAVOR_ID}
 openstack.endpointtype=compute
 "
```

4.7.6 QUERY_FLAVOR_BY_PARAMS command

- Description: Query a Flavor by parameters OpenStack Interface

4.7.6.1 REST

```
"http.operation=GET
 http.url.sufix=/flavors
 openstack.endpointtype=compute"
```

4.7.7 QUERY_FROM_SERVER command

- Description: Query FROM Server OpenStack Interface

4.7.7.1 REST

```
"http.operation=GET
 #http.url.sufix=/v2/${tenant_id}/servers
 http.url.sufix=/servers
 openstack.endpointtype=compute"
```

4.7.8 QUERY_IMAGE command

- Description: Query an Image OpenStack Interface

4.7.8.1 REST

```
"http.operation=GET
 http.url.sufix=/images/${IMAGE_ID}
 openstack.endpointtype=compute
 "
```

4.7.9 QUERY_IMAGE_BY_PARAMS command

- Description: Query an Image by parameters OpenStack Interface

4.7.9.1 REST

```
"http.operation=GET
 http.url.sufix=/images
 openstack.endpointtype=compute"
```

4.7.10 QUERY_SERVER command

- Description: Query a Server OpenStack Interface

4.7.10.1 REST

```
"http.operation=GET
 #http.url.sufix=/v2/${tenant_id}/servers/${SERVER_ID}
 http.url.sufix=/servers/${SERVER_ID}
 openstack.endpointtype=compute"
```

4.7.11 SERVER_ACTIONS command

- Description: Template for server actions

4.7.11.1 JSON

```
"{
 #if( ${VAR_OPERATION}==""RESIZE"")
 ""resize"": {""flavorRef"": ""${FLAVOR_ID}""}
 #end
 #if( ${VAR_OPERATION}==""CONFIRMRESIZE"")
 ""confirmResize"": null
 #end
 #if( ${VAR_OPERATION}==""REVERTRESIZE"")
 ""revertResize"": null
 #end
 #if( ${VAR_OPERATION}==""CHANGEPASSWORD"")
 ""changePassword"": {""adminPass"": ${PASSWORD}}
 #end
 #if( ${VAR_OPERATION}==""REBOOT"")
```

```
 ""reboot"": {""type"": ${TYPE}}
 #end
 #if( ${VAR_OPERATION}==""ADD_FLOATING_IP"")
 ""addFloatingIp"": {
 ""address"": ""${VAR_FLOATINGIPADDRESS}""
 }
 #end
 }"
```

4.7.11.2 REST

```
"http.operation=POST
 #http.url.sufix=/v2/${tenant_id}/servers/${SERVER_ID}
 http.url.sufix=/servers/${SERVER_ID}/action
 openstack.endpointtype=compute"
```

4.7.12 START_SERVER command

- Description: Starts a stopped Server and change its status to RUNNING

4.7.12.1 JSON

```
"{
 ""os-start"": null
 }"
```

4.7.12.2 REST

```
"http.operation=POST
 http.url.sufix=/servers/${SERVER_ID}/action
 openstack.endpointtype=compute"
```

4.7.13 STOP_SERVER command

- Description: Stops a running Server and change its status to STOPPED

4.7.13.1 JSON

```
"{
 ""os-start"": null
 }"
```

4.7.13.2 REST

```
"http.operation=POST
 http.url.sufix=/servers/${SERVER_ID}/action
 openstack.endpointtype=compute"
```

4.7.14 UPDATE_SERVER command

- Description: Update a Server OpenStack Interface

4.7.14.1 JSON

```
"{
 ""server"" :
 {
 ""name"" : ""${SERVER NAME}""
 }
 }"
```

4.7.14.2 REST

```
"http.operation=PUT
 #http.url.sufix=/v2/${tenant_id}/servers/${SERVER_ID}
 http.url.sufix=/servers/${SERVER_ID}
 openstack.endpointtype=compute"
```

4.8 OpenStack plug-in

The OpenStack is the tool for final activation. The resource that is used to set up a virtual machine, modifies a network, or queries a system image as other operations.

It provides the complete process for creating or modifying any operation from the beginning.

4.8.1 OpenStack CS8 user interface

HP Cloud System has implemented a Web environment to make user's interaction easier with the system.

Although NFV Director does not interact with this tool, it helps the user to check if NFV Director OpenStack operations are performed without problems, as the NFVD operations are made using CS8.

Note

Read Appendix B at present document to read about some CS8 user interface

4.8.2 OpenStack plug-in operations

The following operations are possible.

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

Next table shows which version you need when you run operations from CS8:

- For COMPUTE Artifacts:

VIM Type	Manufacturer	Element Type	Version
CS8	OpenStack	OS_NOVA	OS_Havana_v2

Table 17 Version for COMPUTE Artifacts

- For NETWORK Artifacts:

VIM Type	Manufacturer	Element Type	Version
CS8	OpenStack	OS_NEUTRON	OS_Havana_v2

Table 18 Version for NETWORK Artifacts

All operations supported by NFV Director in CS8 VIMs type are shown in next table:

	Vendor ID	ElementType	Version	Command	Description
1	OpenStack	OS_NEUTRON	OS_Havana_v2	ATTACH_VIP_SERVER	Attach a Virtual IP
2	OpenStack	OS_NOVA	OS_Havana_v2	CREATE_FLAVOR	Create a Flavor OpenStack Interface
3	OpenStack	OS_NEUTRON	OS_Havana_v2	CREATE_NETWORK	Create a Network OpenStack Interface
4	OpenStack	OS_NOVA	OS_Havana_v2	CREATE_SERVER	Create a Server OpenStack Interface
5	OpenStack	OS_NEUTRON	OS_Havana_v2	CREATE_SG	Create a SecurityGroup OpenStack Interface
6	OpenStack	OS_NEUTRON	OS_Havana_v2	CREATE_SG_RULE	Create a SecurityGroupRule OpenStack Interface
7	OpenStack	OS_NEUTRON	OS_Havana_v2	CREATE_SUBNET	Create a Subnet OpenStack Interface
8	OpenStack	OS_NEUTRON	OS_Havana_v2	CREATE_VIP	Create a Virtual IP
9	OpenStack	OS_NOVA	OS_Havana_v2	DELETE_FLAVOR	Delete a Flavor OpenStack Interface
10	OpenStack	OS_NEUTRON	OS_Havana_v2	DELETE_NETWORK	Delete a Network OpenStack Interface
11	OpenStack	OS_NOVA	OS_Havana_v2	DELETE_SERVER	Delete a Server OpenStack Interface
12	OpenStack	OS_NEUTRON	OS_Havana_v2	DELETE_SUBNET	Delete a Subnet OpenStack Interface
13	OpenStack	OS_NOVA	OS_Havana_v2	QUERY_FLAVOR	Query a Flavor OpenStack Interface
14	OpenStack	OS_NOVA	OS_Havana_v2	QUERY_FLAVOR_BY_PARAMS	Query a Flavor by params OpenStack Interface
15	OpenStack	OS_NEUTRON	OS_Havana_v2	QUERY_FLOATING_IP	Query a Floating IP OpenStack Interface
16	OpenStack	OS_NOVA	OS_Havana_v2	QUERY_FROM_SERVER	Query FROM Server OpenStack Interface
17	OpenStack	OS_NOVA	OS_Havana_v2	QUERY_IMAGE	Query an Image OpenStack Interface
18	OpenStack	OS_NOVA	OS_Havana_v2	QUERY_IMAGE_BY_PARAMS	Query an Image by params OpenStack Interface
19	OpenStack	OS_NEUTRON	OS_Havana_v2	QUERY_NETWORK	Query a Network OpenStack Interface
20	OpenStack	OS_NEUTRON	OS_Havana_v2	QUERY_NETWORK_BY_PARAMS	Query a Network by params Openstack Interface
21	OpenStack	OS_NEUTRON	OS_Havana_v2	QUERY_PORT	Query a Port for Floating IP OpenStack Interface
22	OpenStack	OS_NOVA	OS_Havana_v2	QUERY_SERVER	Query a Server OpenStack Interface
23	OpenStack	OS_NEUTRON	OS_Havana_v2	QUERY_SUBNET	Query a Subnet OpenStack Interface
24	OpenStack	OS_NOVA	OS_Havana_v2	SERVER_ACTIONS	Template for server actions
25	OpenStack	OS_NOVA	OS_Havana_v2	START_SERVER	Starts a stopped Server and change its status to

					RUNNING
26	OpenStack	OS_NOVA	OS_Havana_v2	STOP_SERVER	Stops a running Server and change its status to STOPPED
27	OpenStack	OS_NEUTRON	OS_Havana_v2	UPDATE_PORT_SECURITY_GROUP	Update Port OpenStack Interface
28	OpenStack	OS_NOVA	OS_Havana_v2	UPDATE_SERVER	Update a Server OpenStack Interface
29	OpenStack	OS_NEUTRON	OS_Havana_v2	UPDATE_SUBNET	Update a Subnet OpenStack Interface

Table 19 Operations in CS8 supported by NFV Director

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.

4.8.3 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 / listed and edited through the HPSA Solution Container: **Administrator**> **ECP** > **Activation Commands Template** > **Template**.

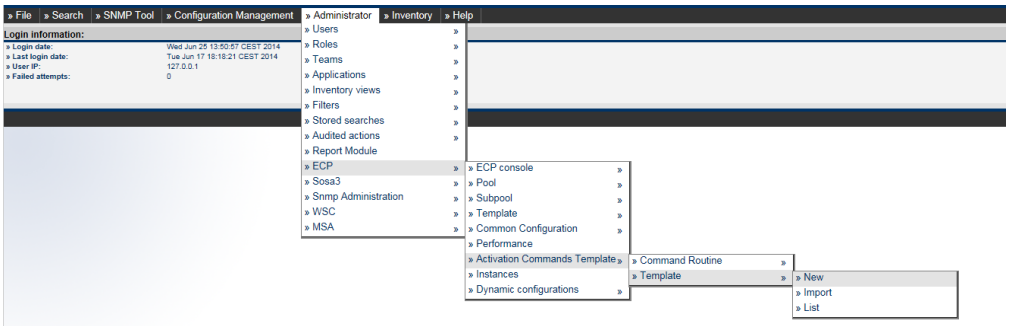


Figure 45 HPSA Solution Container ECP command template

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 46 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

DO—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.

4.8.4 OpenStack Workflows

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

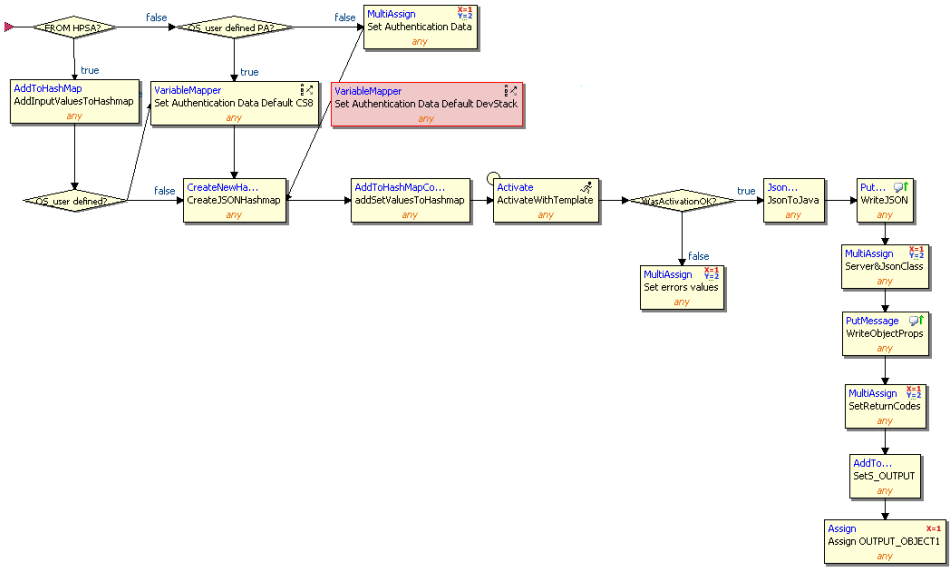


Figure 47 Example: Create Server Workflow

4.8.5 Workflows supported by NFV Director

HP NFV Director v3.0 introduces in this version new concepts about execution tasks. Similar to assignment rules that defines what can be assigned, now is possible to define execution

rules that will define WHAT will be executed, the order and the parallelization of those tasks.

Also, when using execution rules it is possible to track down how many task have been executed, which ones where executed in parallel and the time consumed for each one.

Next sections list all the Workflows in NFV Director v3.0.

	WorkFlow Name	Description
1	WF_NFVD_ACTIVATE	Activate Virtual Machine
2	WF_NFVD_ACTIVATE_CS8	Activate CS8
3	WF_NFVD_ACTIVATE_HELION	Activate HELION
4	WF_NFVD_ACTIVATE_ICEHOUSE	Activate IceHouse
5	WF_NFVD_ACTIVATE_OPENSTACK	Activate OPENSTACK
6	WF_NFVD_ASSIGN_RESOURCES	Assign Resources
7	WF_NFVD_ASSIGN_RESOURCES_OLD	Assign Resources Old
8	WF_NFVD_ASSIGNMENT	Assignment resources
9	WF_NFVD_ASSIGNMENT_OLD	Assignment resources old
10	WF_NFVD_ASSURANCE_MONITOR	Assurance Monitor
11	WF_NFVD_CREATE_CONNECT_NET_FROM_TEMPLATE	Create a tree of instance artifacts and relationship since a template tree
12	WF_NFVD_CREATE_EXECUTION_TASKS	Create a tree of execution tasks from a definition task tree and ARTIFACT_ID
13	WF_NFVD_CREATE_INSTANCES_FROM_TEMPLATE	Create a tree of instance artifacts and relationship since a template tree
14	WF_NFVD_CREATE_POLICY_INSTANCES	Creation All policies instances Child from a template tree
15	WF_NFVD_CREATE_WITHOUT_CONNECT_NET_FROM_TEMPLATE	Create a tree of instance artifacts and relationship since a template tree
16	WF_NFVD_DEACTIVATE	Deactivate Virtual Machine
17	WF_NFVD_DEACTIVATE_NETWORK_CS8	Deactivate Network CS8
18	WF_NFVD_DEACTIVATE_NETWORK_HELION	Deactivate Network HELION
19	WF_NFVD_DEACTIVATE_NETWORK_ICEHOUSE	Deactivate Network ICEHOUSE
20	WF_NFVD_DEACTIVATE_NETWORK_OPENSTACK	Deactivate Network OPENSTACK
21	WF_NFVD_DEACTIVATE_SUBNET_CS8	Deactivate Subnetwork CS8
22	WF_NFVD_DEACTIVATE_SUBNET_HELION	Deactivate Subnetwork HELION
23	WF_NFVD_DEACTIVATE_SUBNET_ICEHOUSE	Deactivate Subnetwork ICEHOUSE
24	WF_NFVD_DEACTIVATE_SUBNET_OPENSTACK	Deactivate Subnetwork OPENSTACK
25	WF_NFVD_DEACTIVATE_VM_CS8	Deactivate CS8
26	WF_NFVD_DEACTIVATE_VM_HELION	Deactivate HELION
27	WF_NFVD_DEACTIVATE_VM_ICEHOUSE	Deactivate ICEHOUSE
28	WF_NFVD_DEACTIVATE_VM_OPENSTACK	Deactivate OPENSTACK
29	WF_NFVD_DELETE_INSTANCE	Delete instance
30	WF_NFVD_DELETE_TEMPLATE	Delete template
31	WF_NFVD_EXECUTE_TASKS	Task separation
32	WF_NFVD_EXECUTE_TASKS_DO	Check if the query was realized ok
33	WF_NFVD_EXECUTE_TASKS_UNDO	Rollbacks the execution of tasks if any error happens
34	WF_NFVD_INSTANCE_ASSIGNMENT	Result must be empty. Oter response will be interpreter as an error
35	WF_NFVD_INSTANCE_VALIDATION	Result must be empty. Oter response will be interpreter as an error

36	WF_NFVD_NS_ORCHESTRATOR	Network Services orchestration
37	WF_NFVD_NS_PRECONNECT	Network Services preconnect
38	WF_NFVD_POSTPROCESS_ACTION	Postprocess action
39	WF_NFVD_PREPROCESS_ACTION	Preprocess action
40	WF_NFVD_PROCESSING_MONITORS	Processing Monitors
41	WF_NFVD_SCALE_CHANGE_FLAVOR_CS8	Scale change flavor on CS8
42	WF_NFVD_SCALE_CHANGE_FLAVOR_HELION	Scale change flavor on HELION
43	WF_NFVD_SCALE_CHANGE_FLAVOR_ICEHOUSE	Scale change flavor on ICEHOUSE
44	WF_NFVD_SCALE_CHANGE_FLAVOR_OPENSTACK	Scale change flavor on OPENSTACK
45	WF_NFVD_SCALE_IN	Scale IN
46	WF_NFVD_SCALE_OUT	Scale OUT
47	WF_NFVD_SCALE_OUT_AND_CONNECT	Scale OUT and connect
48	WF_NFVD_SCALE_OUT_FROM_TREE	Scale OUT from tree
49	WF_NFVD_SCALE_UPDOWN	Scale UP/DOWN
50	WF_NFVD_SCALE_UPDOWN_ACTIVATION	Scale UP/DOWN Activation
51	WF_NFVD_SCALE_UPDOWN_INVENTORY	Scale UP/DOWN Inventory
52	WF_NFVD_START_SO	Start SO
53	WF_NFVD_START_VM	Start Virtual Machine
54	WF_NFDV_STOP_VM	Stop Virtual Machine
55	WF_NFVD_VALIDATE_RELATIONSHIPS_NS	Validate relationships for Network Services
56	WF_OPENSTACK_REST_GET	Sent get OpenStack request
57	WF_OPENSTACK_REST_POST	Sent post OpenStack request
58	WF_TS_ACTIVATE_NETWORK	Activate Network Task
59	WF_TS_ACTIVATE_NETWORK_HELION	Activate Network Task on Helion
60	WF_TS_ACTIVATE_SUBNETWORK	Activate Subnetwork Task
61	WF_TS_ACTIVATE_SUBNETWORK_HELION	Activate Subnetwork Task on Helion
62	WF_TS_ACTIVATE_VM	Activate Virtual Machine Task
63	WF_TS_ACTIVATE_VM_HELION	Activate Virtual Machine Task on Helion
64	WF_TS_DEACTIVATE_NETWORK	Deactivate Network Task
65	WF_TS_DEACTIVATE_NETWORK_HELION	Deactivate Network Task on Helion
66	WF_TS_DEACTIVATE_SUBNETWORK	Deactivate Subnetwork Task
67	WF_TS_DEACTIVATE_SUBNETWORK_HELION	Deactivate Subnetwork Task on Helion
68	WF_TS_DEACTIVATE_VM	Deactivate Virtual Machine Task
69	WF_TS_DEACTIVATE_VM_HELION	Deactivate Virtual Machine Task on Helion
70	WF_TS_MONITOR_DEPLOY	Deploy Monitor Task
71	WF_TS_MONITOR_START	Start Monitor Task
72	WF_TS_MONITOR_STOP	Stop Monitor Task
73	WF_TS_MONITOR_UNDEPLOY	Undeploy Monitor Task
74	WF_NFVD_VNFM_CREATE_TEMPLATES_FROM_DESCRIPTOR	Create templates from descriptor
75	WF_NFVD_VNFM_DELETE_VNF_MANAGER	Delete VNF manager
76	WF_NFVD_VNFM_DELETE_VNF_MANAGER_GENERIC	Delete VNF manager generic
77	WF_NFVD_VNFM_DO_NOTHING	Do nothing
78	WF_NFVD_VNFM_GET_JSON_GRANT	Get JSON grant
79	WF_NFVD_VNFM_GET_MANAGER_VNF_DETAILS	Get manager VNF details
80	WF_NFVD_VNFM_GET_MANAGER_DETAILS_GENERIC	Get manager details generic
81	WF_NFVD_VNFM_JOB_STATUS_MANAGER	Job status manager

82	WF_NFVD_VNFM_JOB_STATUS_MANAGER_GENERIC	Job status manager generic
83	WF_NFVD_VNFM_REGISTER_MANAGER	Register manager
84	WF_NFVD_VNFM_RELATE_VNF_TO_TENANT	Relate VNF to Tenant
85	WF_NFVD_VNFM_SELECT_TEMPLATE_FROM_DESCRIPTOR	Select template from descriptor
86	WF_NFVD_VNFM_SELECT_VNF_IMPL_MANAGER	Select VNF implement manager
87	WF_NFVD_VNFM_UPDATE_STATUS	Update status
88	WF_NFVD_VNFM_UPDATE_STATUS_GENERIC	Update status generic
89	WF_NFVD_VNFM_VNFDESC_MANAGER	VNF description manager
90	WF_NFVD_VNFM_VNFDESC_MANAGER_GENERIC	VNF description manager generic
91	WF_NFVD_VNFM_CHANGE_FLAVOR	Change Flavor
92	WF_NFVD_VNFM_CREATE_VNF	Create VNF
93	WF_NFVD_VNFM_CREATE_VNF_DESCRIPTOR	Create VNF descriptor
94	WF_NFVD_VNFM_DELETE_VNF	Delete VNF
95	WF_NFVD_VNFM_GET_JOB_STATUS	Get Job status
96	WF_NFVD_VNFM_GET_LIST_ORCHESTATOR_REGISTERED	Get list orchestator registered
97	WF_NFVD_VNFM_GET_VNF_DESCRIPTOR_DETAILS	Get VNF Descriptor details
98	WF_NFVD_VNFM_GET_VNF_DESCRIPTOR_LIST	Get VNF Descriptor list
99	WF_NFVD_VNFM_GET_VNF_DETAILS	Get VNF Details
100	WF_NFVD_VNFM_GET_VNF_LIST	Get VNF List
101	WF_NFVD_VNFM_REGISTER_ORCHESTATOR	Register Orchestator
102	WF_NFVD_VNFM_SCALE	Scale
103	WF_NFVD_VNFM_UNREGISTER_ORCHESTATOR	Unregister Orchestator
104	WF_NFVD_OPENSTACK_Assign_SG	Openstack Interface: Assign Security Group
105	WF_NFVD_OPENSTACK_Assign_Virtual_IP	Openstack Interface: Create Server
106	WF_NFVD_OPENSTACK_Create_Flavor	Openstack Interface: Create Flavor
107	WF_NFVD_OPENSTACK_Create_Floating_IP	Openstack Interface: Activate Floating IP
108	WF_NFVD_OPENSTACK_Create_Network	Openstack Interface: Create Network
109	WF_NFVD_OPENSTACK_Create_Port	Openstack Interface: Create Network
110	WF_NFVD_OPENSTACK_Create_Server	Openstack Interface: Create Server
111	WF_NFVD_OPENSTACK_Create_SG	Openstack Interface: Create Network
112	WF_NFVD_OPENSTACK_Create_SG_Rule	Openstack Interface: Create Network
113	WF_NFVD_OPENSTACK_Create_Subnet	Openstack Interface: Create Subnet
114	WF_NFVD_OPENSTACK_Create_Virtual_IP	Openstack Interface: Create Server
115	WF_NFVD_OPENSTACK_Delete_Flavor	Openstack Interface: Delete Flavor
116	WF_NFVD_OPENSTACK_Delete_Network	Openstack Interface: Delete Network
117	WF_NFVD_OPENSTACK_Delete_Server	Openstack Interface: Delete Server
118	WF_NFVD_OPENSTACK_Delete_Subnet	Openstack Interface: Delete Network
119	WF_NFVD_OPENSTACK_Query_Flavor	Openstack Interface: Query Flavor
120	WF_NFVD_OPENSTACK_Query_Image	Openstack Interface: Query Image
121	WF_NFVD_OPENSTACK_Query_Network	Openstack Interface:

		Query Network
122	WF_NFVD_OPENSTACK_Query_Server	Openstack Interface: Query Server
123	WF_NFVD_OPENSTACK_Query_Subnet	Openstack Interface: Query Subnet
124	WF_NFVD_OPENSTACK_Server_Actions	Openstack Interface: Delete Server
125	WF_NFVD_OPENSTACK_Start_Server	Openstack Interface: Starts a stopped server and set its status to ACTIVE
126	WF_NFVD_OPENSTACK_Stop_Server	Openstack Interface: Stops a running server and set its status to STOPPED
127	WF_NFVD_OPENSTACK_Update_Network	Openstack Interface: Update Network
128	WF_NFVD_OPENSTACK_Update_Server	Openstack Interface: Update Server
129	WF_NFVD_OPENSTACK_Update_Subnet	Openstack Interface: Update Subnet

Table 20 Workflows supported

Read *HP NFV Director v3.0 WorkFlows - Reference guide* for more detailed info about NFV Director v3.0 WorkFlows.

4.9 CS8 – REST Interface

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

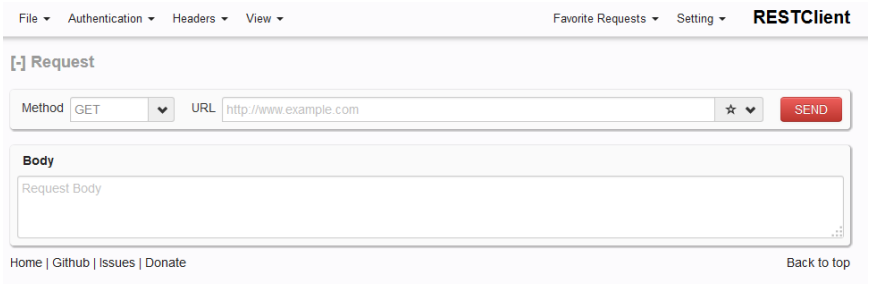


Figure 48 CloudSystem Firefox Rest Client

You should indicate proper headers.

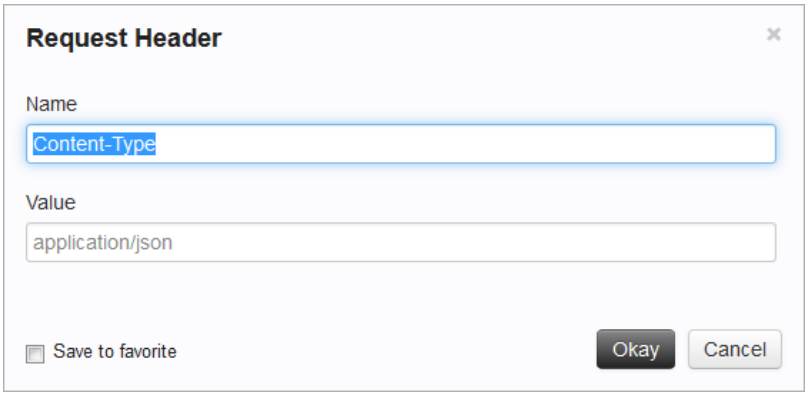


Figure 49 CloudSystem Rest Client Request Header

4.9.1 Operations

4.9.1.1 Create Server

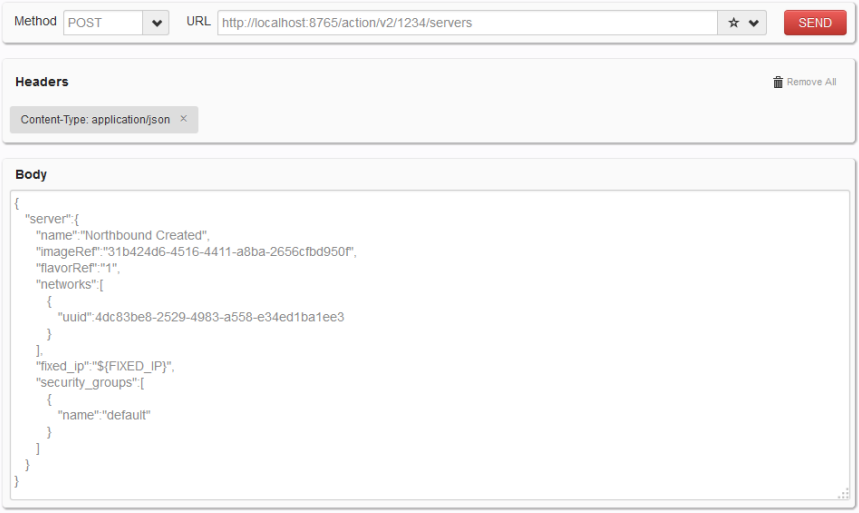


Figure 50 CloudSystem Create Server operation

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

4.9.1.2 Edit Server

Figure 51 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`

4.9.1.3 Get Server by Id

Figure 52 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`

4.9.1.4 Delete Server

Figure 53 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`

4.9.1.5 Start Server

Figure 54 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`

4.9.1.6 Stop Server

Figure 55 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`

4.9.1.7 Query Image

Figure 56 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`

4.9.1.8 Create Flavor

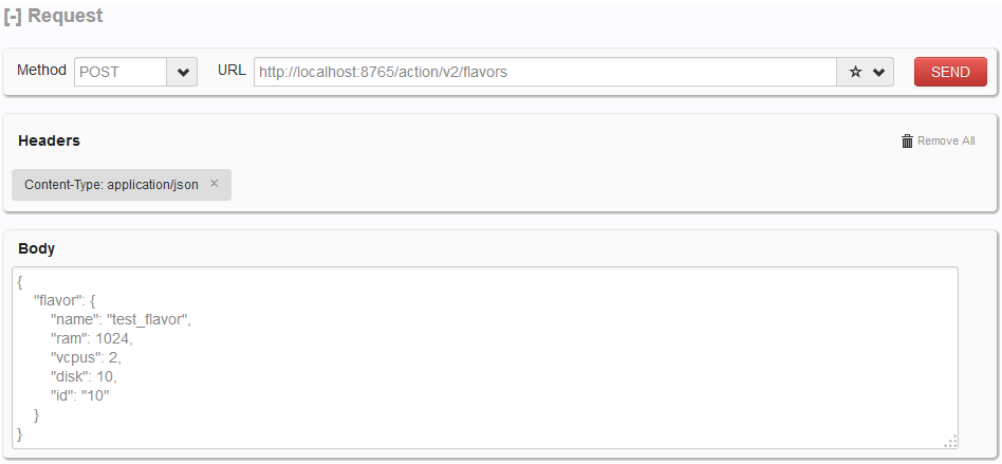


Figure 57 CloudSystem Create Flavour operation

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

4.9.1.9 Get Flavor

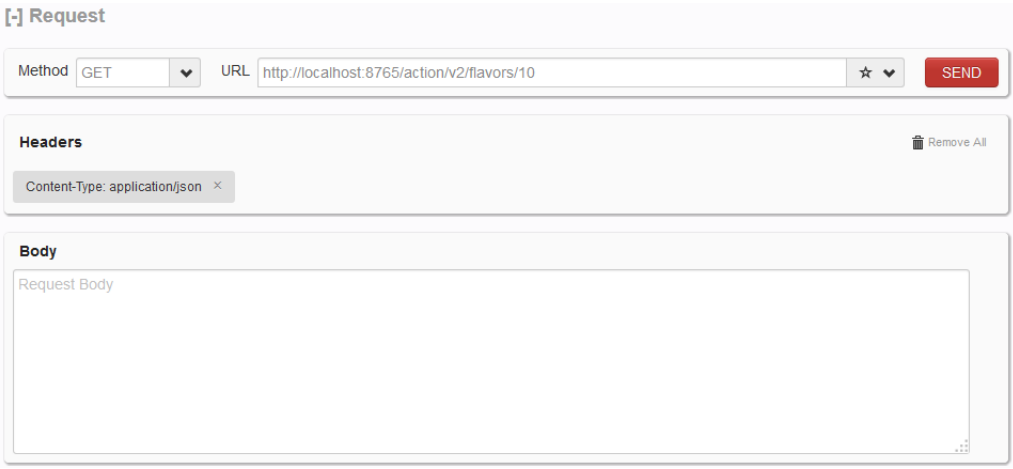


Figure 58 CloudSystem Query Flavour operation

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

4.9.1.10 Get Flavor by Parameters

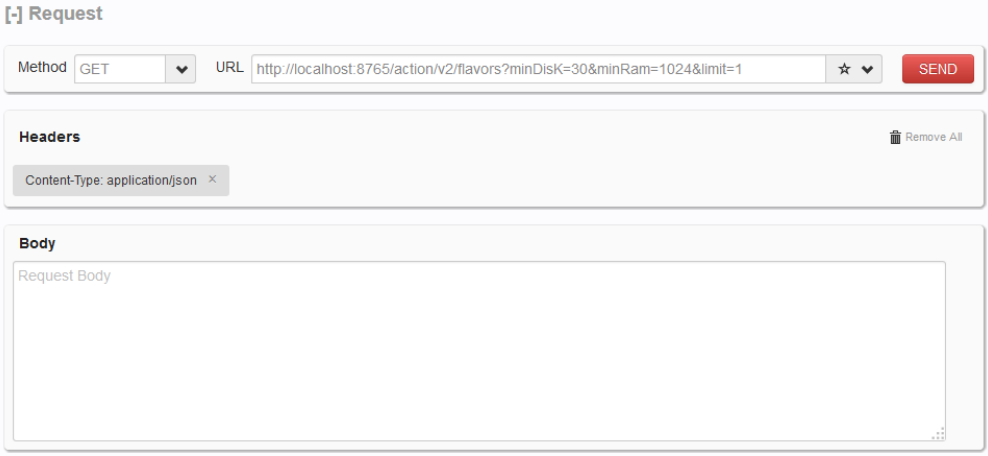


Figure 59 CloudSystem Query Flavour by Parameters operation

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

4.9.1.11 Delete Flavor

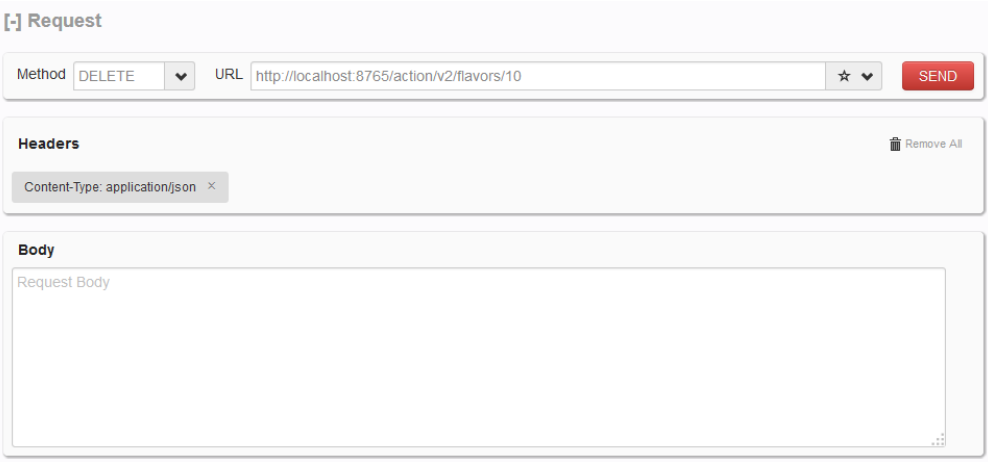


Figure 60 CloudSystem Delete Flavour operation

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

4.9.1.12 Create Network

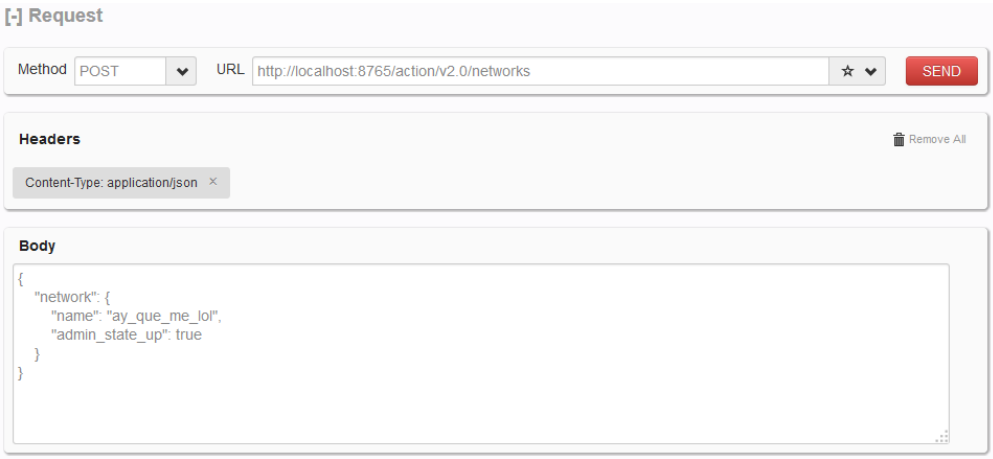


Figure 61 CloudSystem Create Network operation

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

4.9.1.13 Edit Network

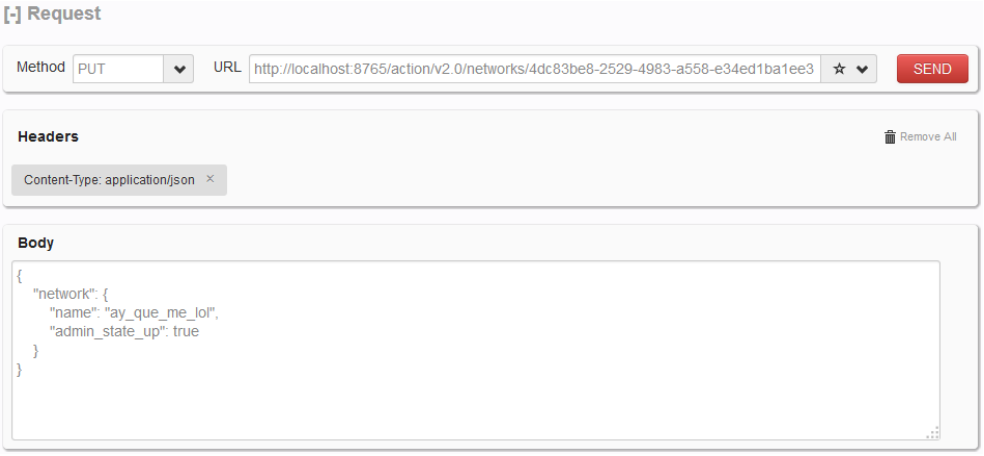


Figure 62 CloudSystem Edit Network operation

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

4.9.1.14 Get Network by ID

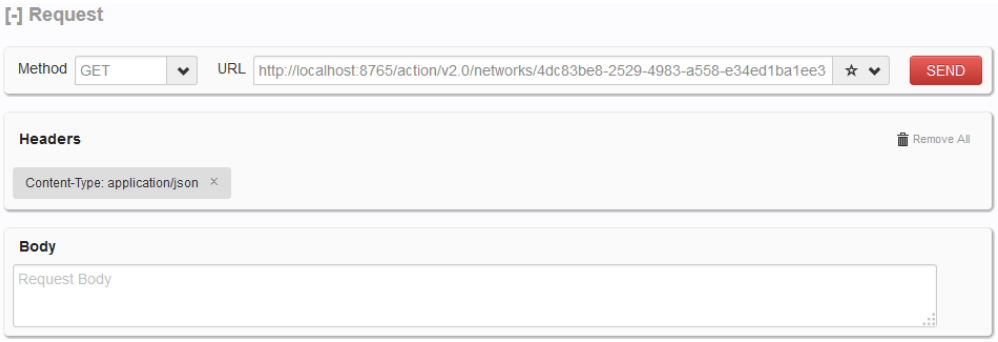


Figure 63 CloudSystem Query Network by ID operation

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

4.9.1.15 Get Network by Parameters

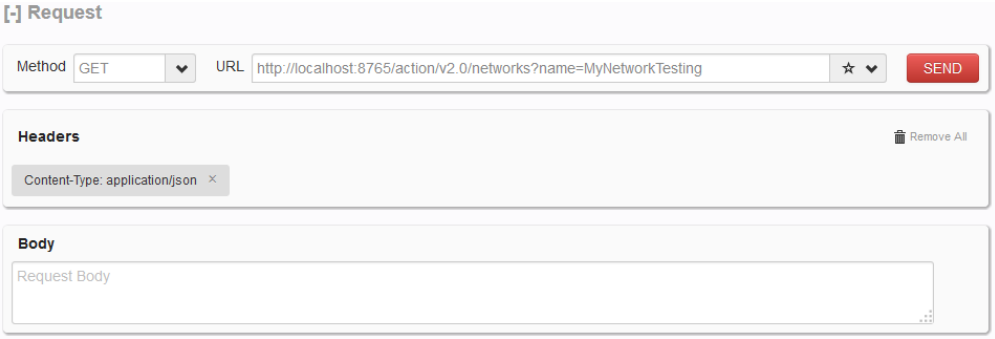


Figure 64 CloudSystem Query Network by Parameters operation

URL:
`http://localhost:8765/action/v2.0/networks?name=MyNetworkTesting`
`http:$host:$Protocol_Adapter_port/action/$version_api/networks?$param1=$value1`

4.9.1.16 Delete Network

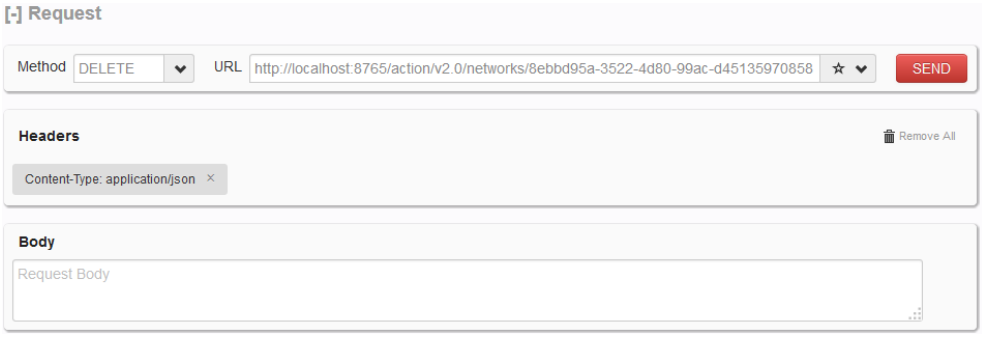


Figure 65 CloudSystem Delete Network operation

URL: `http://localhost:8765/action/v2.0/networks/4dc83be8-2529-4983-a558-e34ed1ba1ee3`

http:$host:$Protocol_Adapter_port/action/$version_api/networks/$network_id

4.9.1.17 Create Subnet

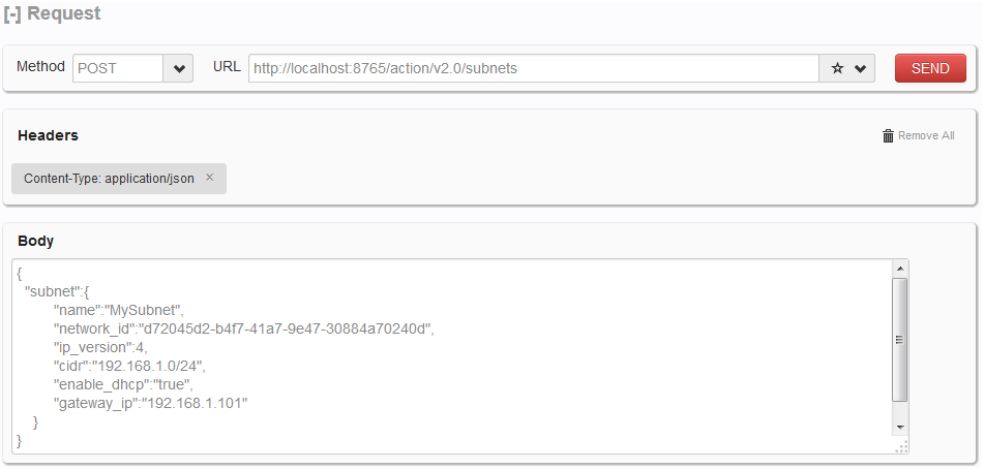


Figure 66 CloudSystem Create Subnet operation

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

4.9.1.18 Edit Subnet

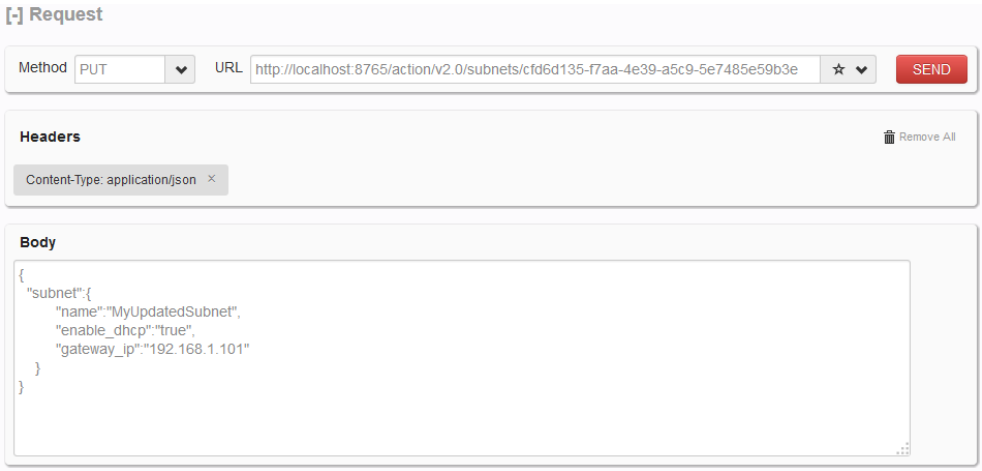


Figure 67 CloudSystem Edit Subnet operation

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

4.9.1.19 Get Subnet

Figure 68 CloudSystem Query Subnet operation

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

4.10 VIMs configuration

NFV Director needs to know the appropriate URL and credentials for the VIM and the appropriate tenants so they can be configured at the plugin level, workflow level and monitoring level.

If the credentials in CS8 do not match the operations NFVD triggers some errors will be raised at runtime execution.

Out of the box there are no Workflows to end2end create Networks although the atomic operation is provided in the OpenStack southbound plugin so they should be created upfront on the VIM or a Workflow developed to create them when needed.

If NFV director is going to choose the server where the VMs are going to be created then OpenStack regions and availability zones need to be created to the level of server so that the NFV director can specify the server using the region or availability zone.

Chapter 5

Creating VNF

One of the main objectives in NFV Director is to create VNFs and deploy them in one or more Virtual Infrastructure Managers.

The orchestrated creation operation performs a complete sequence of actions to have a VNF fully deployed in NFV Director and VIMs.

The next figure summarizes all the needed steps:

Figure 69 Creating VNF process

Thus, actions to be taken to create and deploy VNFs are:

4. Create instances.
5. Assign instances to physical resources.
6. Launch the activation operation.
7. Deploy VNF.

First process in VNF creation is to create instances.

The easiest way for creating instances is using templates.

You can also upload instances using VNF Visual Designer.

5.1 Creating instances

The VNF template tree should be well-formed with the right relationships between child elements and parents.

When creating instances, the following two additional operations are involved:

- Assignment
- Validation

The artifact template tree can include three types of policies, each one with its special functionality:

- POLICY:ENTITY_ASSIGN
- POLICY:VALUE_VALIDATION
- POLICY:ENTIY_RANGE

The first additional operation sets the attributes of an instance indicated by the ENTITY_ASSIGN template. The second additional operation validates those attribute values against the policy VALUE_VALIDATION. If the validation is incorrect, the instances cannot be created.

One thing to keep in mind is that when a VNF is created from a template, it stores the template ID (in an internal field) that is used. If the creations were triggered from the end to end, the creation stores the assignment tree and the resource tree that were used as well (these IDs are stored in a special category). The assignment tree and the resource tree used are stored only for the VNF artifact.

WARNING: When an instance is being activated, only the VMs are created. These VMs are attached to a network that already exists on the VIM. The VMs that are in instantiated status are the ones that are activated. Deactivating process is similar, taking only VMs which statuses are not instantiated.

5.2 Generating a template

The VNF template should be well-formed with the policies for the additional operations to be successful. This section discusses the structure of the template.

The following illustration explains the tree with the relationships between child components and parents.

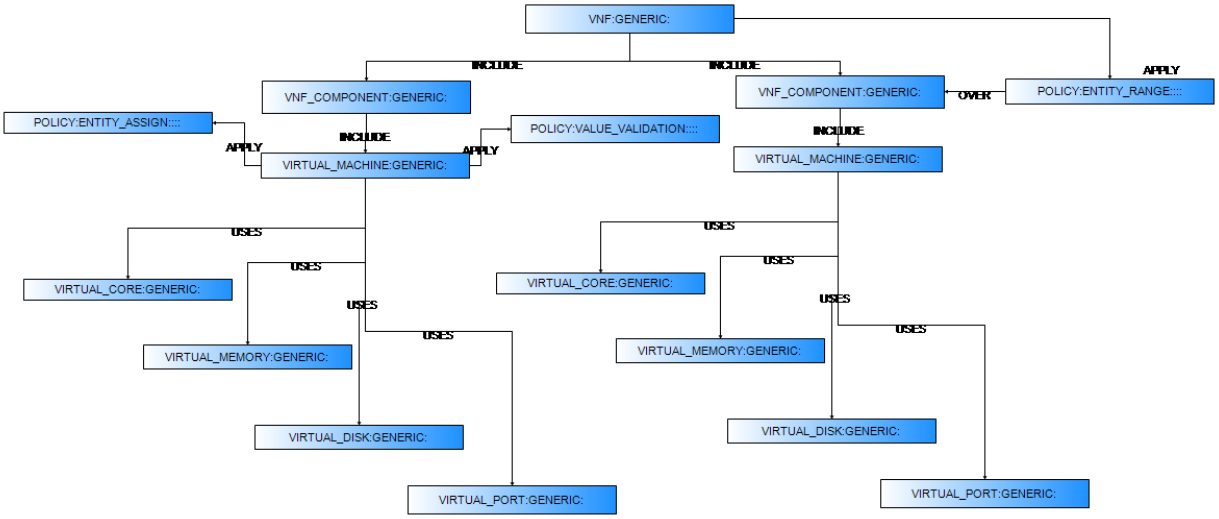


Figure 70 VNF Template example

5.3 Policies

This section discusses the policies and how to set the attributes for running the additional operations when creating the VNF.

Three kinds of policies are available.

- Assignment
- Validation
- Range

5.3.1 Assignment

Assignment is the additional operation linked with the POLICY:ENTITY_ASSIGN. In this task, you can set the attribute values before creating the instance.

You can assign the policy in three ways:

- Java method
- Script method
- Workflow method

To select the assigning method, configure the POLICY:ENTITY_ASSIGN properly. This policy has a category named **ASSIGN** with three attributes:

- TYPE
- EXECUTION
- ATTRIBUTE

The assignment mode is decided on the basis of values you enter for the attributes.

Set the policy for each one of the three modes.

5.3.1.1 Java method assignment

TYPE = JAVA

EXECUTION = Complete path of the Java method.

Example:

If you want to assign the attributes **Speed** and **Amount** belonged to INFO category, using the method, assign to the following JavaMethodUtils class. The assignment must be self-programmed in Java:

```
com.hp.ov.activator.nfv.nodes.JavaMethodUtils.assign(INFO.Speed
,INFO.Amount)
```

Note

Write the attributes to be assigned between the brackets conforming to the Java method name. The format for the attributes is `Category.Attribute`.

5.3.1.2 Script assignment

TYPE = SCRIPT

EXECUTION = Complete path of the Script.

For example, `com.hp.ov.activator.nfv.name_script.sh`

ATTRIBUTE: `Category.Attribute`

For example, if the policy applies to a VIRTUAL_CORE and you want to assign the attribute Speed, and the attribute belongs to the INFO category, enter the following:

```
ATTRIBUTE: INFO.Speed.
```

The script must return an integer that can be assigned to the attribute. The script must also be self-programmed.

5.3.1.3 Workflow assignment

TYPE = WORKFLOW

EXECUTION = Name of the workflow to be launched.

ATTRIBUTE = Category.Attribute

The workflow must be self-built.

5.3.2 Validation

The other additional operation provides the possibility to validate the attribute values. Similar to the assignment task, you can validate in 4 ways:

- Range validation
- Java method
- Script method
- Regular Expression

The POLICY:VALUE_VALIDATION should be configured properly. The **destiny** value, which is mandatory variable, should be set in the attribute **ARTIFACT_CATEGORY_ATTRIBUTE_TARGET**. To set it, the format of the attribute should be `Category.Attribute`. For VIRTUAL_MACHINE, a sample format is `INFO.Speed`.

After setting the attribute values, validate this attribute according to the policy mode specified in next type. To select the validation mode, set the appropriate fields in the next order.

Each policy validates and if any validation fails, creation fails and only a database rollback is performed.

5.3.2.1 Range validation

Range validation method takes the value of the artifact destiny. It checks if the value is greater than or equal to the policy validation MIN_VALUE and also whether the value is lower than or equal to the policy validation MAX_VALUE. You should define the values in integers (1, 2, 3, and so on).

For example:

MIN_VALUE=1

MAX_VALUE=3

In this example, in VIRTUAL_MACHINE, if the value of `INFO.Speed` is not greater than or equal to 1 and if it is not lower than or equal to 3, the validation fails.

5.3.2.2 Java method validation

This method takes the value of the artifact **destiny** to validate the content of the attribute Class of the policy. The content must be the fully qualified name of the class and the name of the method to execute using the corresponding parameters.

```
FullyQualifiedClassName.methodName (Param1, Param2, …)
```

For example:

```
com.hp.ov.activator.nfv.nodes.JavaMethodUtils.imprime
(INFO.Speed,INFO.Amount)
```

The validation must be self-programmed in Java.

5.3.2.3 Script validation

This method uses an external script. The input contains the full path to the script and the attribute to be validated that is located in the instance. The script returns an integer with number 1 to check the correct execution. Modify the last script line with the following code:

```
VAR_SCRIPT_RESULT=1"
```

This code line sends a correct value to a case packet variable for checking the correct execution of the script.

For example, `com.hp.ov.activator.nfv.name_script.sh`

Validation must be self-programmed into the script.

5.3.2.4 Regular Expression

This method checks the content of the value located in the **destiny** variable with a pattern defined in the attribute REGULAR_EXPRESSION.

For example, if you want to check whether the attribute value contains a character, enter the character in the REGULAR_EXPRESSION field.

This mode of validation happens only when you complete another one of the modes of validation. The order to check the modes is the following:

8. Range validation
9. Java method
10. Script method
11. Regular Expression

5.3.3 Range

You can add another policy (POLICY:ENTITY_RANGE) to the **Create Instance from Template** operation. This policy allows the opportunity to create more than one instance in a single time of concrete artifacts.

5.3.3.1 Configuring the policy

The DEFAULT_SCALE_OUT indicates the number of instances to create. The MAX attribute controls the maximum number of instances that the NFV Director can have.

For example, if the policy is applied over a VNF_COMPONENT with two instances, the DEFALUT_SCALE_OUT = 2, and MAX = 3, the policy can create only one instance as the maximum is 3.

5.4 Virtual Machines and VNF lifecycle

This section explains the Virtual Machines lifecycle. The different status on Virtual machines can be checked on status field on NFVD interface and CS8 interface.

5.4.1.1 Start Virtual Machine

The initial status is:

VNF→ENABLE

VIRTUAL MACHINE → STOPPED/SHUTOFF

MONITOR → STOPPED

First step:

VNF→LOCKED

VIRTUAL MACHINE → STOPPED/SHUTOFF

MONITOR → STOPPED

Second step:

VNF→LOCKED

VIRTUAL MACHINE → ACTIVE/ACTIVE

MONITOR → STOPPED

Third step:

VNF→LOCKED

VIRTUAL MACHINE → ACTIVE/ACTIVE

MONITOR → STARTED

Final status is:

VNF→ENABLE

VIRTUAL MACHINE → ACTIVE/ACTIVE

MONITOR → STARTED

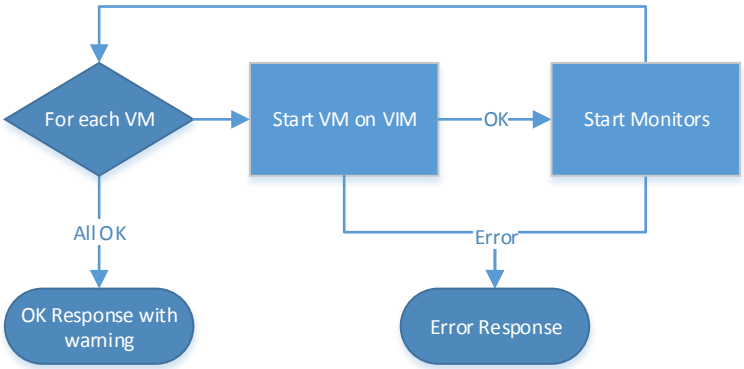


Figure 71 Procedure for Start virtual machine

5.4.1.2 Stop Virtual Machine

The initial status is:

VNF→ENABLE

VIRTUAL MACHINE → ACTIVE/ACTIVE

MONITOR → STARTED

First step:

VNF→LOCKED

VIRTUAL MACHINE → ACTIVE/ACTIVE

MONITOR → STARTED

Second step:

VNF→LOCKED

VIRTUAL MACHINE → ACTIVE/ACTIVE

MONITOR → STOPPED

Third step:

VNF→LOCKED

VIRTUAL MACHINE → STOPPED/SHUTOFF

MONITOR → STOPPED

Final status is:

VNF→ENABLE

VIRTUAL MACHINE → STOPPED/SHUTOFF

MONITOR → STOPPED

5.4.1.3 Scale Up/Down

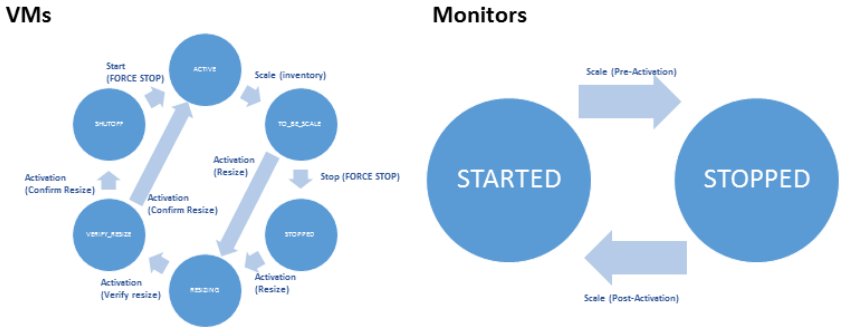


Figure 72 Virtual machine lifecycle

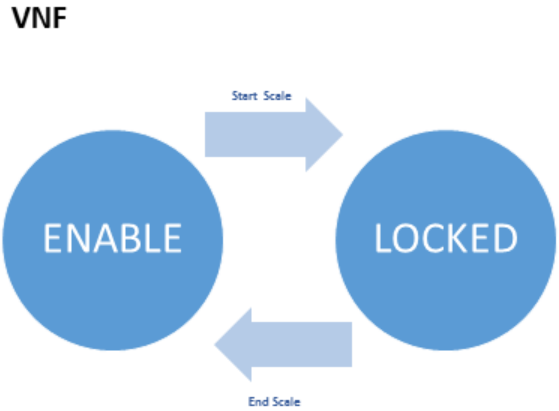


Figure 73 VNF lifecycle

5.5 Examples

This section explains two examples of the procedure to create VNF.

5.5.1 Example with Range Policy

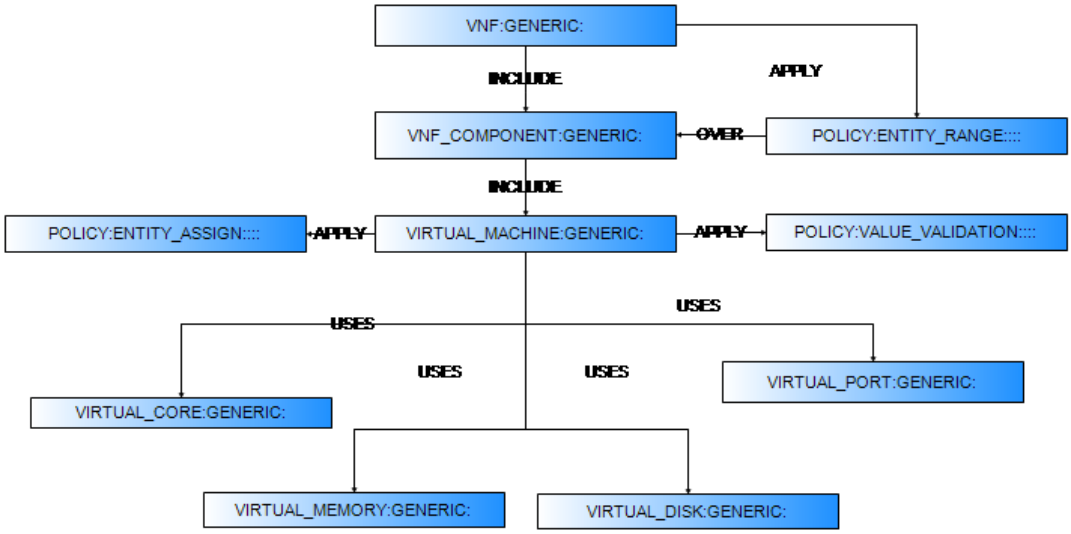


Figure 74 Template example with range policy

5.5.1.1 Instances

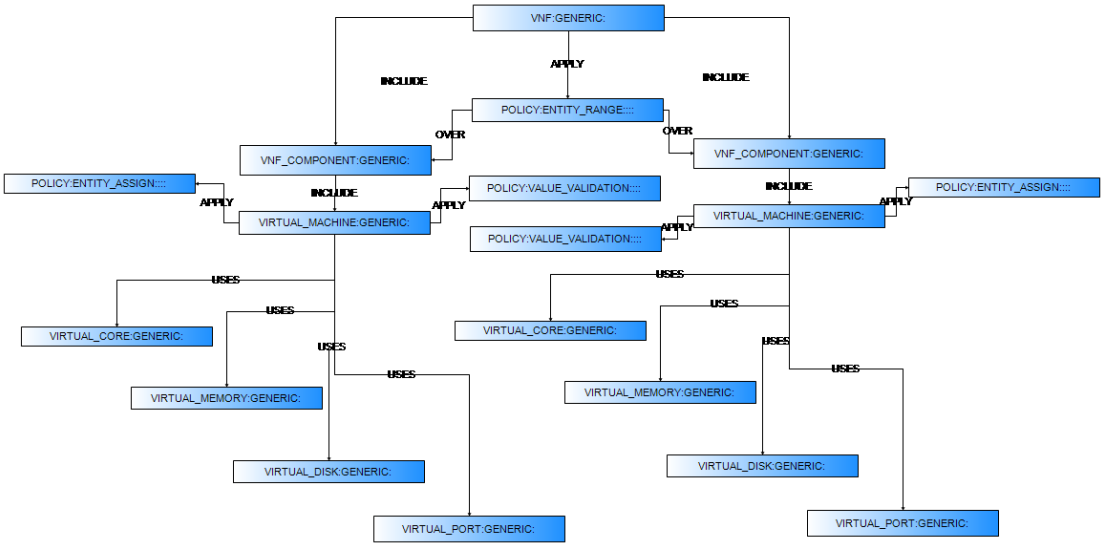


Figure 75 Instance result of template example with range policy

Doing the operation in the VNF and with the parameters set this way:

DEFAULT= 1, MAX= 5,

A VNF is generated with 2 VNF_COMPONENT as child components of the VNF.

If the VNF_COMPONENT does not have the POLICY:ENTITY_RANGE, the same instances tree is generated that is mentioned in the templates.

Note

You can create an instance only if the validation is correct.

5.5.2 Example with Assign Policy

5.5.2.1 Instances

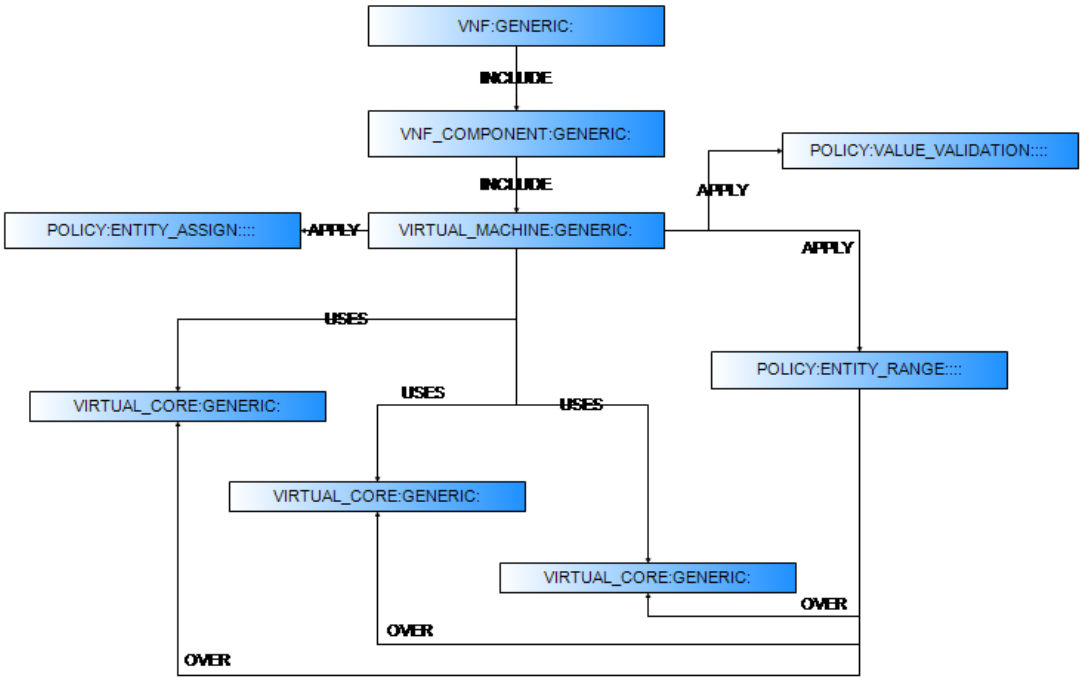
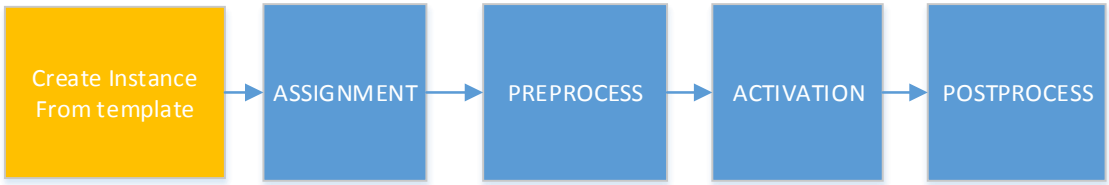


Figure 76 Instance result of Template example with assign policy

In this case, the policy creates two VIRTUAL_CORE instead of 4, because the MAX is 3.

5.6 Inside Orchestration

This is the first step in getting a view over the complete orchestration process.



Starting from a complete VNF template, the first process is to create an analogue VNF instance tree, where the policies described above are applied.

Chapter 6

VNF Resource assignment

6.1 Assignment Process

The Assignment process is the second major process involved in the global creation process. This process creates relationship between elements of instance tree created during instantiation and a pool of resources, which is another artifact tree. To create relationship between elements, you should define the elements the artifact instance consumes and the resources to which these instances can be allocated.

The instances tree and resources tree are artifact instances, which can be defined as:

- Artifacts instance tree—an instance tree created from artifact tree template.
- Resources tree—an instance tree used as resources to be consumed by the artifact instance tree.

6.2 Policies

The following sections describe the policies involved in resource assignment process.

6.2.1 Assignment relationship

Assignment relationship is defined as a hierarchical tree of relationship to assign instances to resources. The following illustration provides an overview.

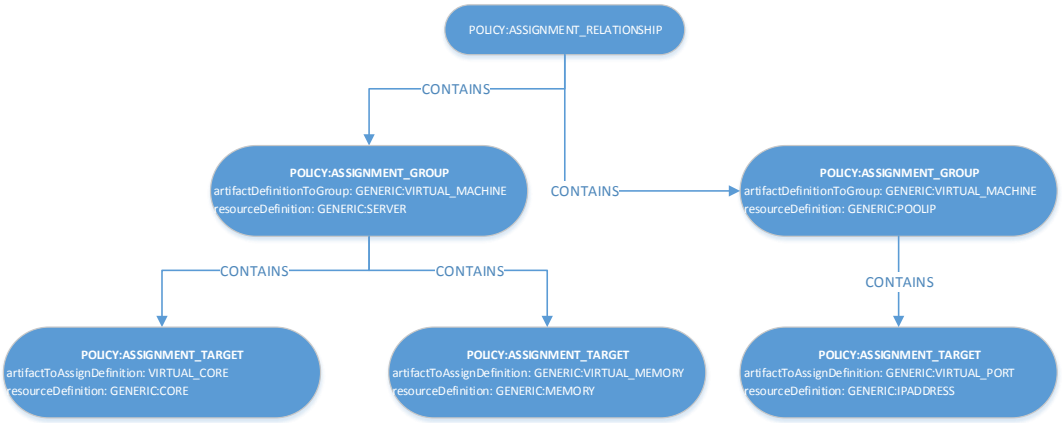


Figure 77 Policy Assignment Relationship hierarchical tree

6.2.1.1 ASSIGNMENT_RELATIONSHIP

This artifact is only a reference to grouping different kinds of relationships that you want to create. This artifact is the parent of elements that contain and define the relationship that you want.

It can have only one type of child: ASSIGNMENT_GROUP.

6.2.1.2 ASSIGNMENT_GROUP

This artifact represents a logical association between equivalent elements from instances and resources. For example, a virtual machine might be allocated inside a physical server. This representation does not involve a creation of a final relationship in the database and is meant only as a reference.

The representation is used to reduce and group final resource targets for low level instances. For example, when you define an assignment group between virtual machines and physical servers, you are defining that all elements of a virtual machine including vCore or memory (detailed and defined in ASSIGNMENT_TARGET policy) must have a direct and real relationship stored in the DB with resources under physical server artifact (core, memory, and so on).

This policy has an implicit validation. The resource candidate must have enough capacity to hold the instance. In other words, the final target resource candidate amount (cores, memory) must be greater than the artifact instance amount (vCores, memory) that you want to allocate on it.

Attributes under the Group category are the following:

- artifactDefinitionToGroup: Definition of the instances that must be related and in the NFVD Format (<Family>:<Category>:<Group>:<Type>:<SubType>:<Version>).
- resourceDefinition: Definition of the resources where the instances are related and in the NFVD Format (<Family>:<Category>:<Group>:<Type>:<SubType>:<Version>).
- relationshipTypeCountList: comma separated list of types of relationship that counts as consumer for resources. Typically `ALLOCATED`.

It can have only one type of child: ASSIGNMENT_TARGET.

6.2.1.3 ASSIGNMENT_TARGET

This policy identifies the relationship that the assignment process creates in the database. The policy defines the final elements that are related between, for example, vCores and physical cores or virtual memory and physical memory.

The assignment process gets all instances from the instances tree defined by the artifactDefinitionToGroup attribute in the Assignment group policy and a valid resource from the resources tree defined by the resourceDefinition attribute in the Assignment group policy.

Then it creates each relationship defined in the assignment targets.

Attributes under TARGET category are the following:

- artifactDefinitionToGroup: Definition of the instances that must be created for the relationship and in the NFVD Format (<Family>:<Category>:<Group>:<Type>:<SubType>:<Version>).
- resourceDefinition: Definition of the resources where the instances are related and in the NFVD Format (<Family>:<Category>:<Group>:<Type>:<SubType>:<Version>).
- Relationshiptype: Type of relationship that will be created between resource and artifact.

6.2.2 Over_subscription

All instances count as 1 as amount, unless it has an attribute name `INFO.Amount` that indicates a different amount.

Some resources around virtualization can have over-subscription to allocate more instances that the amount permits. To do so, you can define an over-subscription policy attached on that resource.

Attributes under OVER_SUBSCRIPTION category are the following:

- Rate—Ratio of over-subscription. The final amount is calculated as INFO.Amount per Rate. For example, a core with amount 8 and over-subscription rate as 2, has a final amount of 16.
- RELATIONSHIP_TYPE—not used on this release. It indicates the relationship of over-subscription.
- MIN—not used on this release.
- MAX—not used on this release.

An over-subscription policy can be attached over any artifact that is defined. If this policy is included in a template, you should create a relationship, because a parent artifact template is instantiated.

6.2.3 Affinity

Sometimes you may have to allocate some instances over the same resource (two virtual machines in the same location) or these instances need to share another resource (some ports over same network). In these scenarios, you should define an Affinity policy that applies over some instances and share the same final resource.

Affinity policy works like an Assignment relationship policy, where you designate which particular elements to get together. For those instances, the same rules apply as that of the assignment policies.

Assignment policies are general rules to create relationship. For all artifacts defined for group policy, their target relationship policies are applied. Otherwise, you can define different affinity policies to group concrete artifacts and apply only on the subset of its elements.

For example, a general policy is available to assign virtual cores and memory over physical core and memory and virtual ports to IP address on a server. You can define an affinity policy because you need some of the virtual machines to be created over the same location (only applies to core and memory, but not the relationship between ports and IP address).

Attributes under AFFINITY category are the following:

- Type
    - MUST—If the resource is not a valid resource to allocate all artifacts, the process returns an error.
    - SHOULD—If the resource is not valid, the assignment ignores the affinity policy.
- AFFINITY_TARGET—Definition of the resources which the affinity instances are related and in the NFVD Format (<Family>:<Category>:<Group>:<Type>:<SubType>:<Version>).
- FINDMETHOD—not used in this release. This attribute contains a search method to narrow the entire list of resources.
- LEVEL—not used in this release.

6.2.3.1 Relationship

If this policy is included in a template, you should create a relationship of the APPLY type, because a parent artifact template should be instantiated. You should have defined it earlier.

- Must have at least one relationship of the type APPLY over another artifact (group of affinity). It must be defined earlier.
- Must have at least one relationship of type CONTAINS over a policy ASSIGNMENT_GROUP.

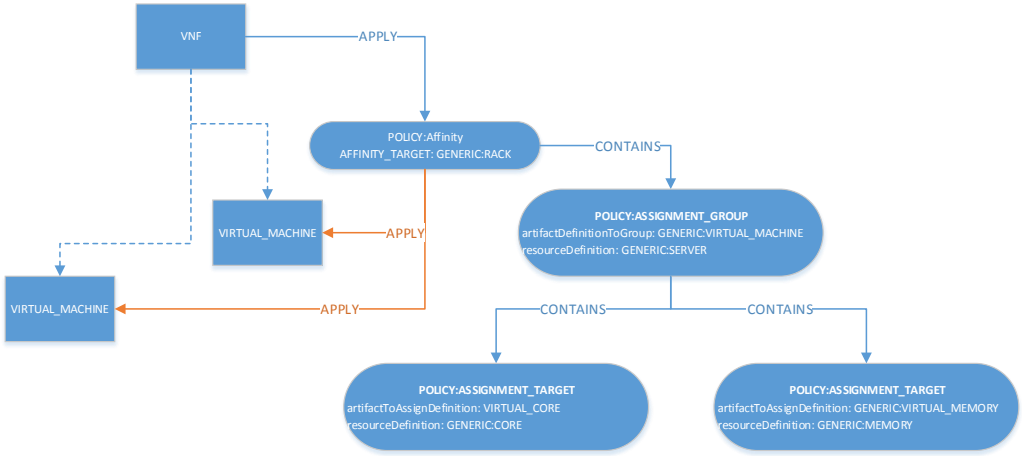


Figure 78 Policy: Apply Relationship

6.3 Process Description

This section provides an overview of the assignment process, how to apply the policies listed in the previous sections, and how final relationships are created.

This process needs the following tree input parameters:

- artifactTreeID—Parent artifact ID of instances tree that needs to be allocated.
- resourceTreeID—Parent artifact ID of resources tree which can allocate instances.
- assignmentRelationshipID—Assignment relationship ID.

Use the following procedure to assign the affinity policies:

12. Assign the artifacts that have affinity policies associated.

    The process queries all affinity policies and their groups and target of assignment.

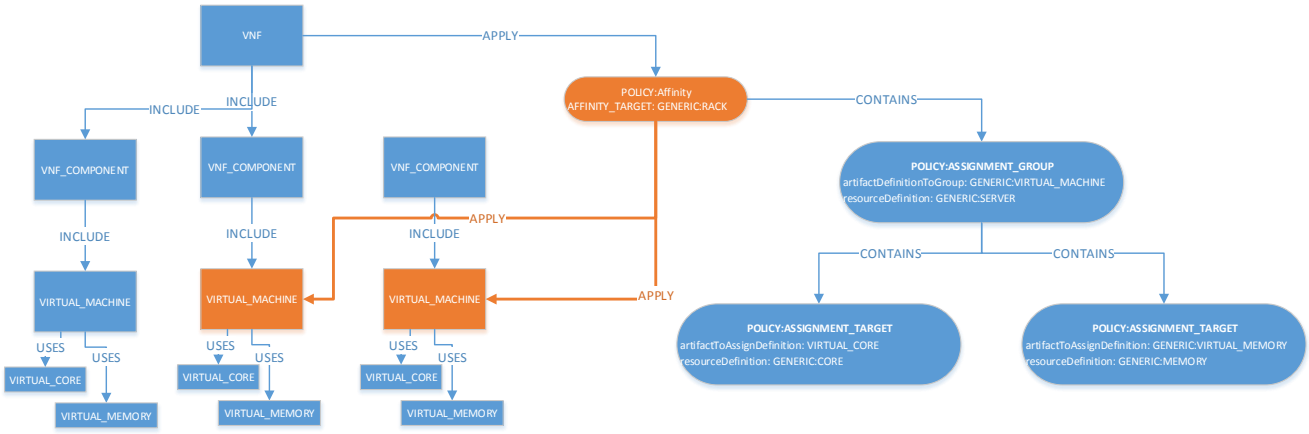


Figure 79 Policy Assignment process

For each affinity policy found:

- It queries all affine artifacts and calculates total amount of all policy targets defined in the affinity policy tree.

- It queries the resource tree for all resource candidates identified by the AFFINITY_TARGET attribute on the Affinity current policy. For each resource candidate, the free amount of current candidate is compared with total amount of all affine artifacts.

  If this resource can contain all artifacts, it proceeds to assign these affine artifacts to this resource (following the group and target policies). If not, it tries with the next resource candidate.

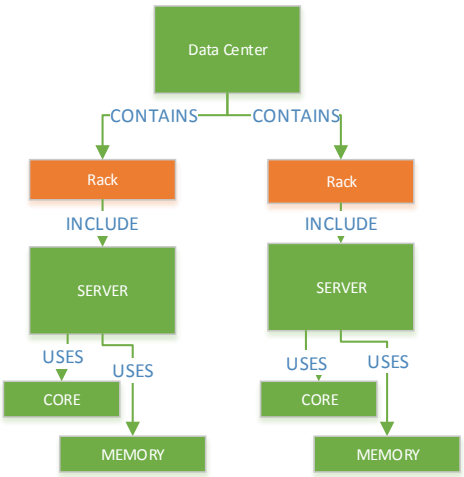


Figure 80 Policy Assignment Flow

Each assignment group of affinity policy is queried for all resources (children of affinity resource candidate) defined in the resourceDefinition attribute and all artifacts instances identified by the artifactDefinitionToGroup attribute.

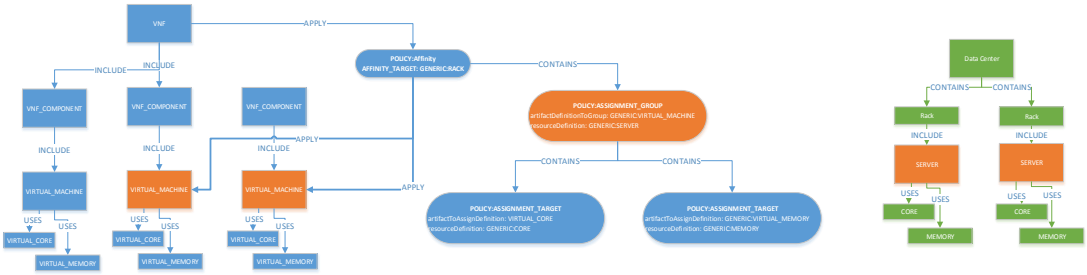


Figure 81 Policy Assignment Group

13. For each artifact and resource queried by the group policy, all target policies are queried.

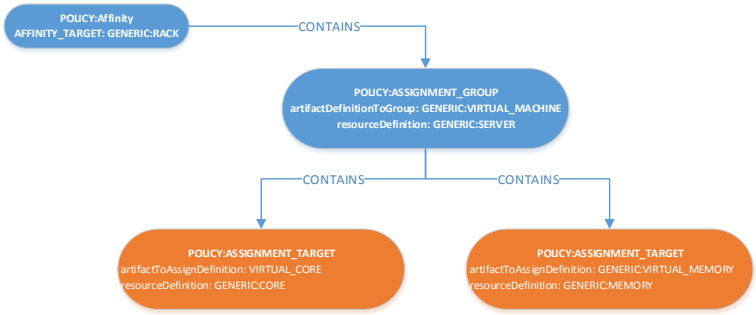


Figure 82 Query Target Policies

14. Query again for artifacts, including all child elements defined for each assignment target policy.

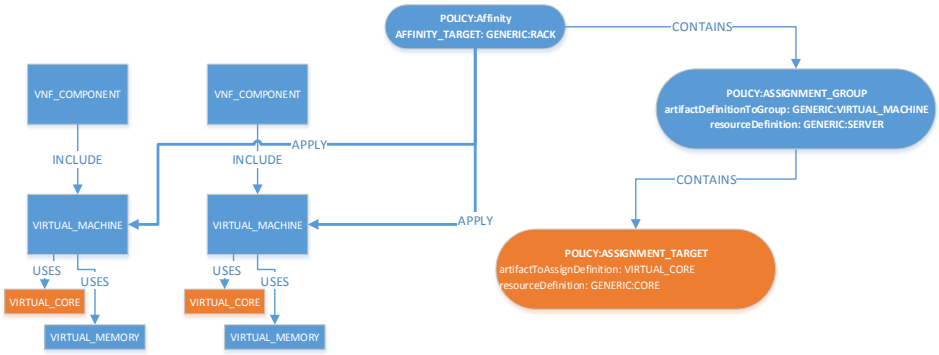


Figure 83 Query artifact for assignment target policy

15. Perform the same steps with resources.

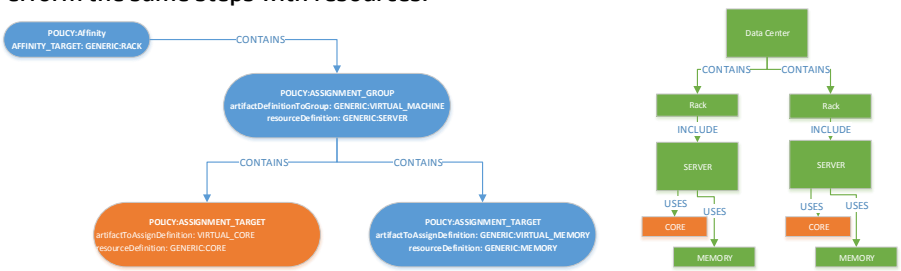


Figure 84 Query Resources for assignment target policy

16. Final instances and resources of each artifact are placed in the descending order and resources are placed in ascending order.

17. Assign if the resource allows an artifact.

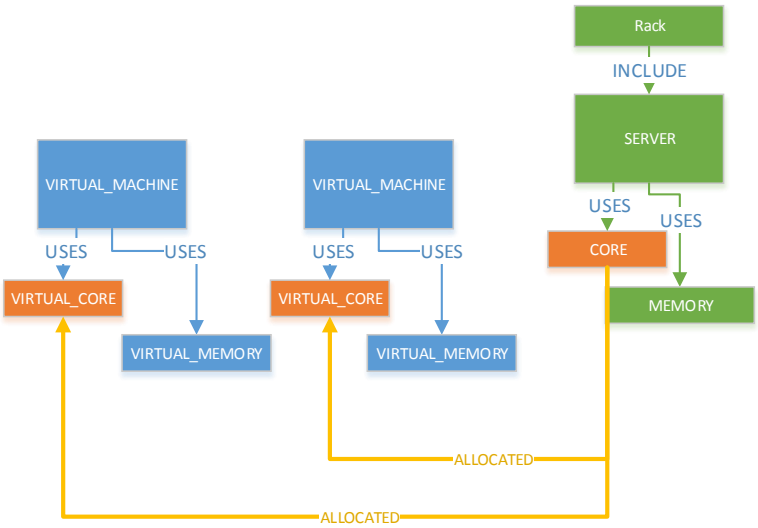


Figure 85 Assign Resource Artifact

Note

If the relationship already exists, it is assumed that it was created in the previous affinity of assignment process, but not reported as an error.

18. When all affinity policies are processed, a generic assignment process is launched, following similar steps as that of affinity, but on the resource tree.

For each assignment group policy, all resources defined in the resourceDefinition attribute and all artifacts instances identified by the artifactDefinitionToGroup attribute are queried.

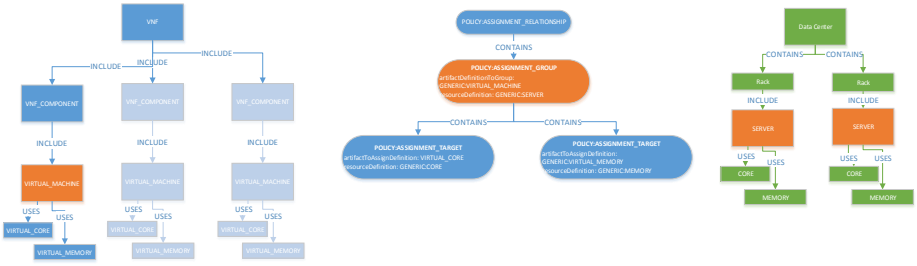


Figure 86 Query resources in Assignment Group policy

19. Each artifact and resource is queried for target policies and group policies.

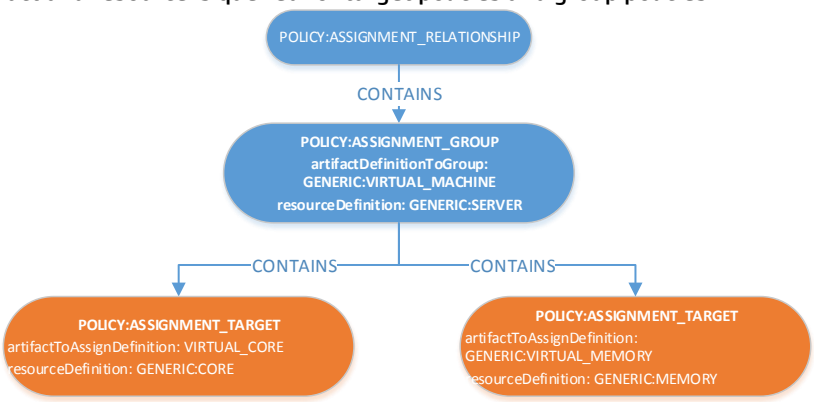


Figure 87 Query Target policy and Group policy for artifact and resource

20. Query again over artifact all children defined for each assignment target policy.

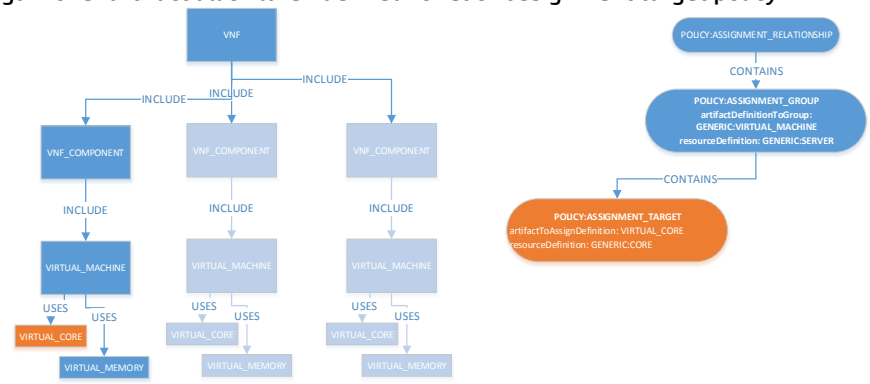


Figure 88 Query Assignment Target Policy for artifact children

21. Perform the same steps with resources.

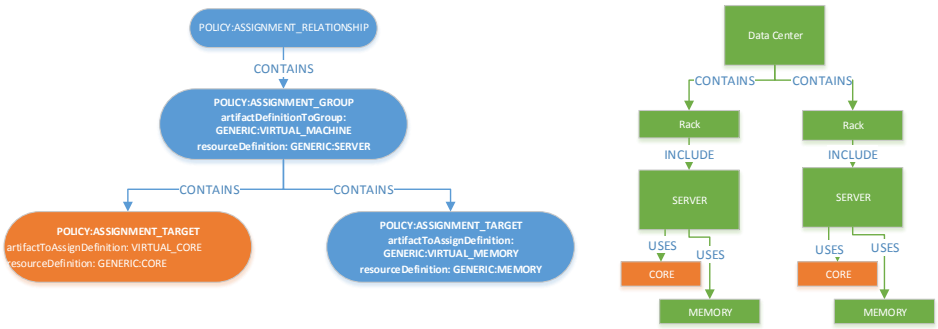


Figure 89 Query Assignment Target Policy for Resource Children

22. Final instances and resources of each artifact it will be ordered from greater to lower and resources will be ordered from lower to greater, and try to assign if the resource can allow artifact.

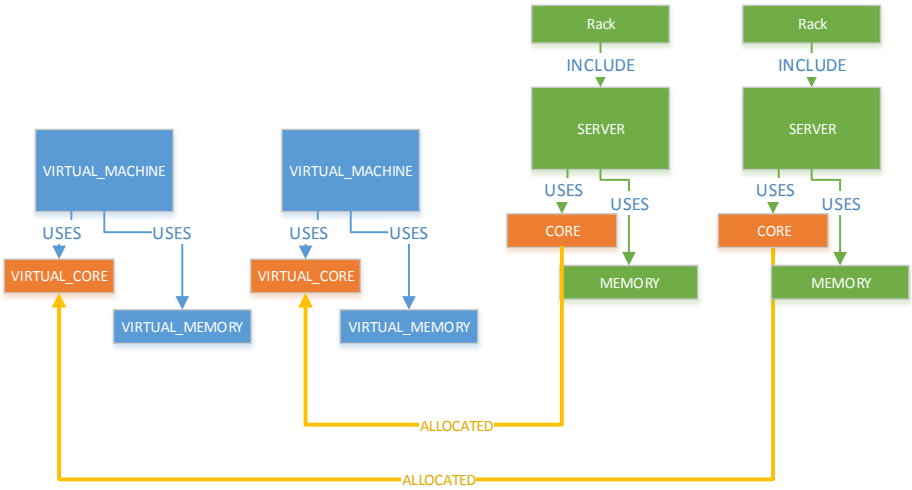


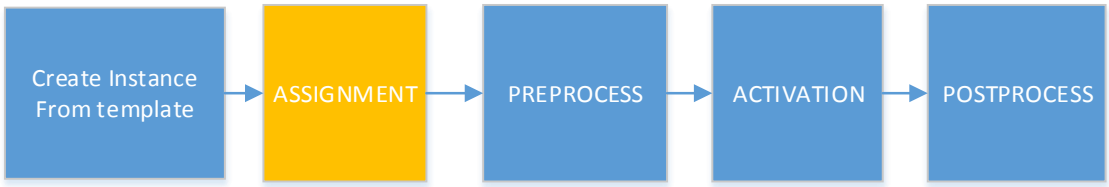
Figure 90 Assign artifact to Resource

Note

If the relationship already exists, it is assumed that it was created in the previous affinity of assignment process, but not reported as an error.

6.4 Inside Orchestration

Getting a view over complete orchestration process:



When the VNFs are instantiated, the new instances must be allocated or reserved within a concrete resource pool.

Chapter 7

Activation

The third step on global creation process is activation, and the process will be responsible to check the relationship between the TENANT and the machine, after that the resources will be detected in a flavor. Later the activation will be effective.

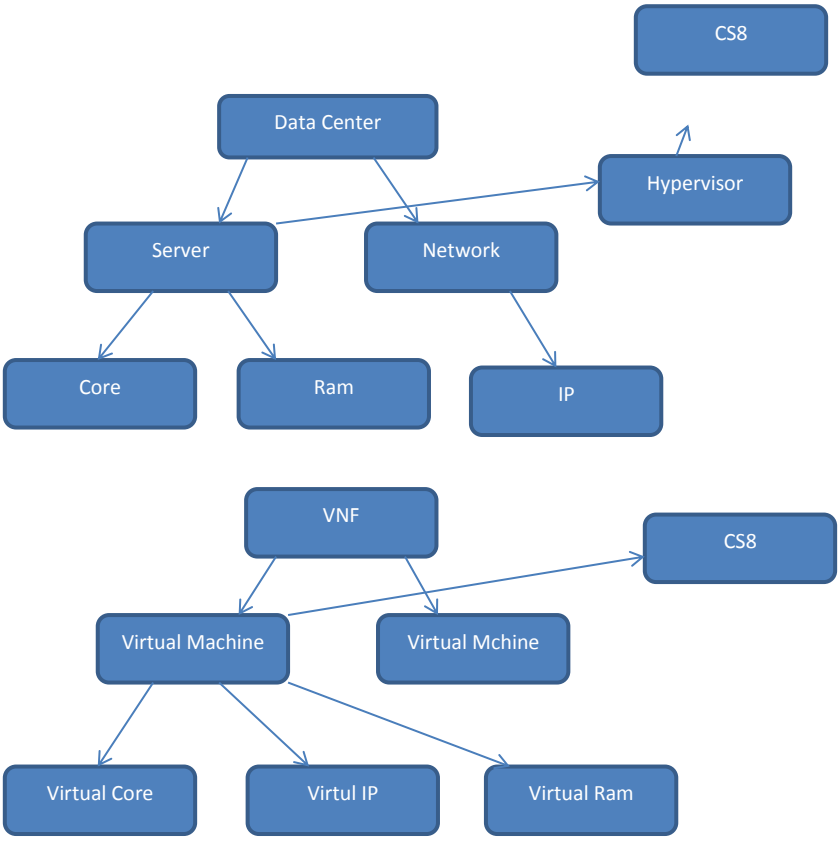


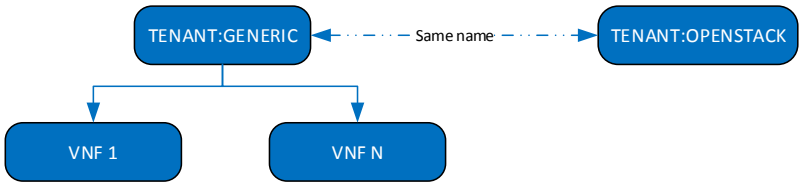
Figure 91 Activation flow

The basic steps in the process are:

- Check the image and take the image ID.
- Check a flavor with ram and disk.
- Check network Id.
- Create Server with image ID, RAM, disk, and network ID.

7.1 Checking and getting the Tenant and VIM

All VNFs must be created below a Tenant, and this tenant name will be used to create the instances of VMs under OpenStack Project with same name (OpenStack Tenant).



On the other hand, all VMs are assigned to resources of any VIM walking through Virtual Core allocation. Use this method to find the VIM data is required to instantiate that VM.

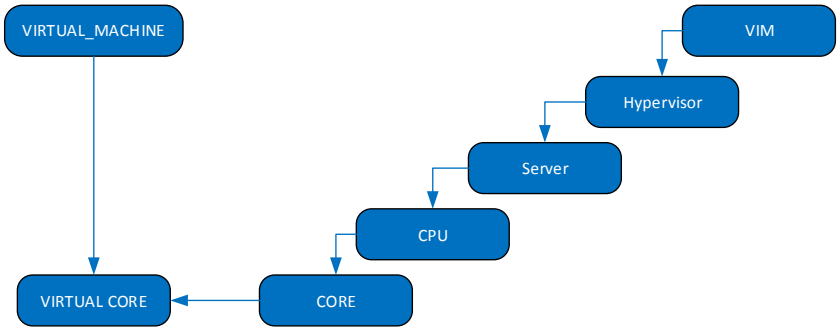


Figure 92 Activation: get vDC

This action manages the extraction of the entire data that is required later.

7.2 Checking a flavor with RAM and disk

After you get the Tenant, you must compose the flavor. It is mandatory to get the Virtual RAM and then the virtual disk.

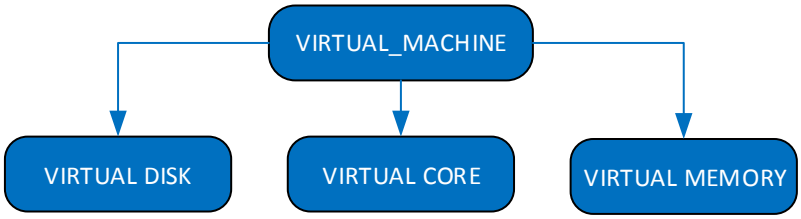


Figure 93 Activation: get Virtual Memory and Virtual Disk

When the virtual memory, virtual core, and the virtual disk are obtained, check the correct value of these in OpenStack flavor. The flavor is used later. The flavor ID is extracted to be used in the operation.

7.3 Getting Image ID and network ID

After getting the flavor, get the machine image.

23. For getting the image, the OpenStack workflow is called from the activation.
24. Other OpenStack workflow is called to assign the resources to an output object.
25. The network connected to the object is located.

    An OpenStack workflow is called to get the network.
26. When the resource is added, you should assign the output-object to the network.

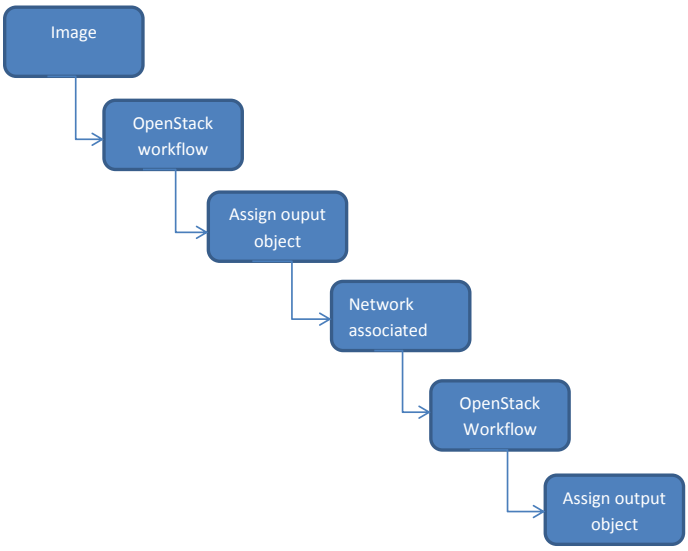


Figure 94 Activation: get Image ID and network ID

7.4 Creating Server

At this point, a different OpenStack workflow is called. This workflow is named with the category of the VIM instance. It means that you could start different activations depending on different types of VIMs.

7.5 Updating Status

When the server is created, the check operation is coming—the server previously created is called and associated to an output object, then the activation checks the server has been created properly and the machine is active, and it changes the status to activated, then it takes the ID returned by CS8 and sets it into VIM ID—the artifact is updated.

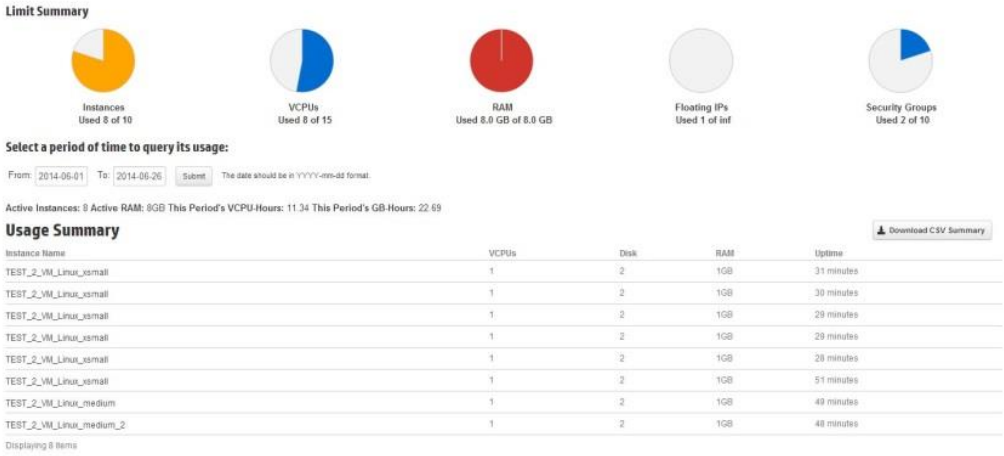


Figure 95 Activation: update Status

7.6 Activating workflow parent

The CS8 activation is called from a workflow parent. It works getting all the VNF components and getting the VIM components. Depending on the VIM status the full operation of activation is called or not.

7.6.1 Testing

The activation is the last part of other operations:

- Create instance from template
- Scale in
- Scale Out
- Scale Up/down

Test it properly to launch the other operations

7.7 Pre and post processing actions

Before and after any activation of the process, check if there exists any pre-processing or post-processing policy attached to any element (VNF, VNF_COMPONENT or VIRTUAL MACHINE), and then start the operation related to this policy.

This policy called `POLICY:POSTPRE_PROCESSING` contains the following attributes:

- Workflow: this is the name of workflow that will be launched. It must exist.
- Job_type: possible values: `PRE` or `POST`. This parameter is used when the workflow has to be launched, `PRE` means before activation and `POST` means after activation.
- Operation: possible values: CREATE, DELETE, SCALE_IN or SCALE_OUT. This parameter denotes the operation on which the workflow will be launched.

7.8 Deactivating workflow

The `deactivate_CS8` workflow is called to deactivate virtual machines. The process of deactivating involves using an artifact ID to get VIM, virtual datacenter, and through it, the server name.

When the server name is obtained, the `OpenStack delete` operation is called and after a test, the operation is completed.

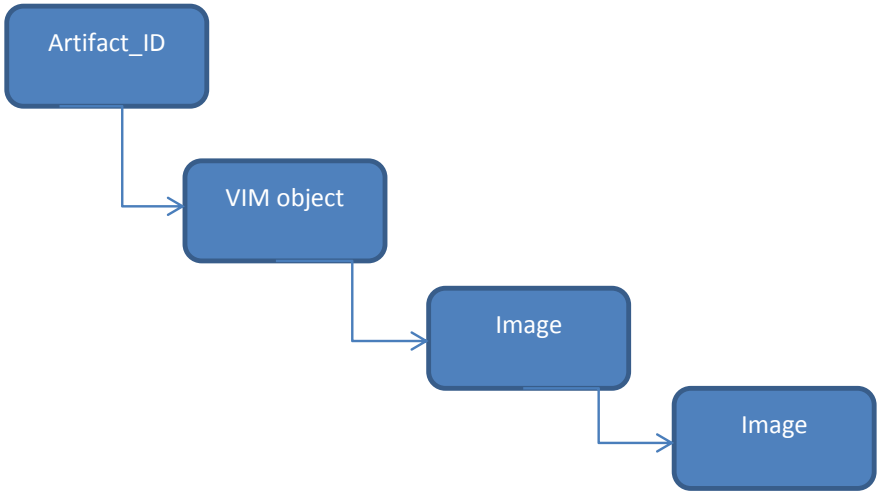


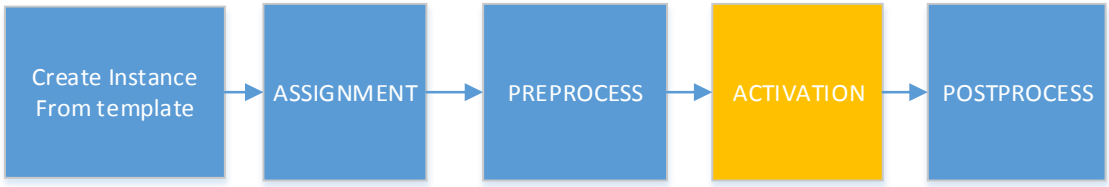
Figure 96 Deactivate flow

7.9 Error Recap

- 6XXX—Error related to active flow parent.
- 1XXXX—Error related to activate cs8.
- 14XXX—Error related to deactivate flow.

7.10 Inside Orchestration

Getting a view over complete orchestration process:



Chapter 8

VNF Scaling

When you execute a scale operation over a VNF, the service model can grow or shrink (scale out/in) or can maintain the same service model.

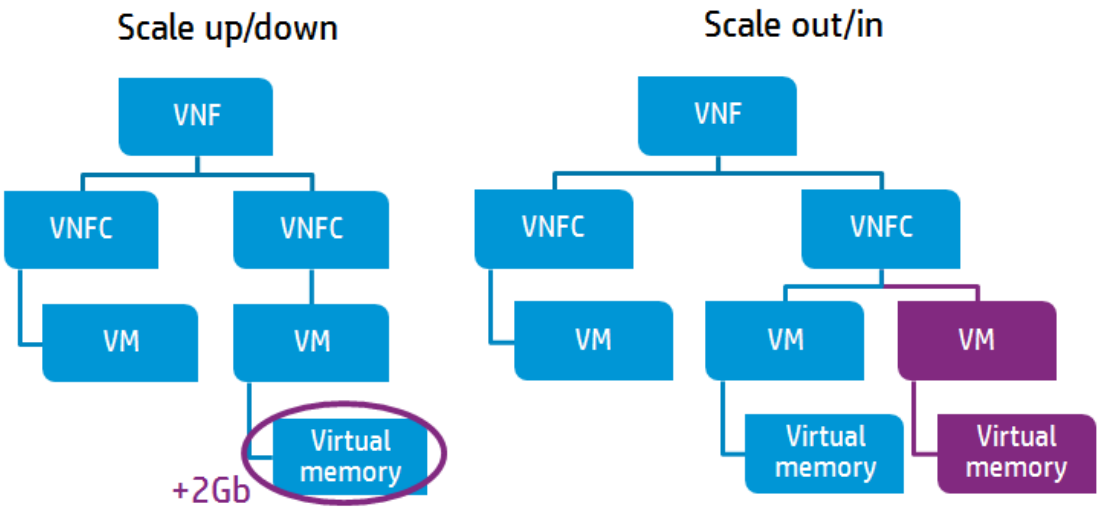


Figure 97 VNF Scales operations

If the VNF increases its capacity by adding more Virtual Machines, the scale operation over the VNF is a Scale Out: new elements are created in the VNF representation tree.

However, if the VNF decreases its capacity deleting some VMs, it is a Scale In operation and elements are deleted from the VNF representation tree.

When the VNF capacity is increased by increasing VM resources, the scale operation is a Scale Up.

Decreasing VNF capacity by decreasing VM resources (without deleting VMs), is a scale Down.

8.1 Scale-in

The scale in operation is built to decrease, assign, and activate resources on the instance tree. For example: one virtual machine with 2 children (Virtual Cores).

Actually this operation can be called only from a VNF artifact.

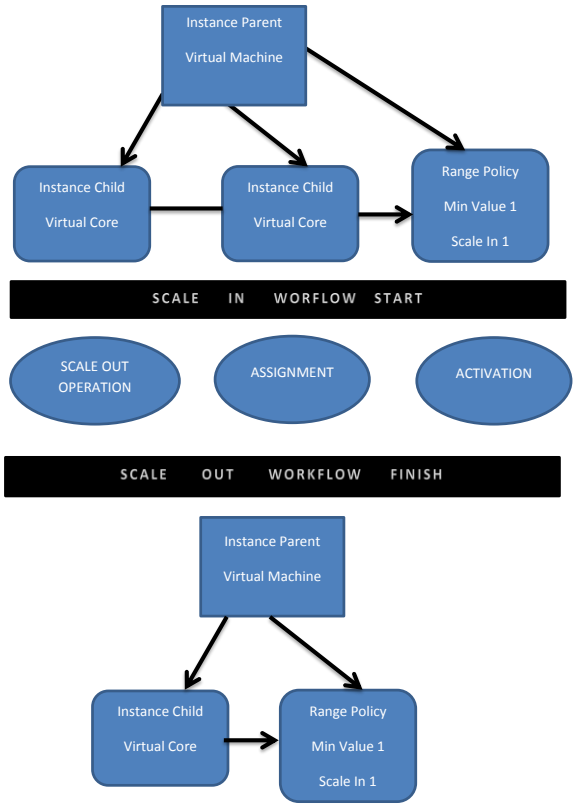


Figure 98 VNF Scale In

You can decrease the number of Virtual Cores according to the policy range using the scale-in operation. The Scale-In operation is also responsible for un-assigning and activating the resources.

Particular Case:

When scale in workflow is called, it tries to scale as many elements as possible. If any element tries to scale below the minimum, it does not scale. However, the workflow continues scaling other components and displays a warning.

For example, consider a scenario where a 2 Virtual machine VNF starting as:

VM1 = 5 instance, VM2 = 5 instance, where

VM1 default=5, scale in=5, min= 2,

VM2 default=5, scale in=1, min= 1

If it tries to scale, the result is VM1 = 5 instance, VM2 = 4 instance 1, because VM1 cannot scale below the minimum but VM2 can.

8.1.1 Scale in operation

Scale in operation is in charge of decreasing the amount of resources.

It starts checking the Instance ID and getting the artifact associated to this ID. After getting the artifact the operation continues getting the template associated with the instance.

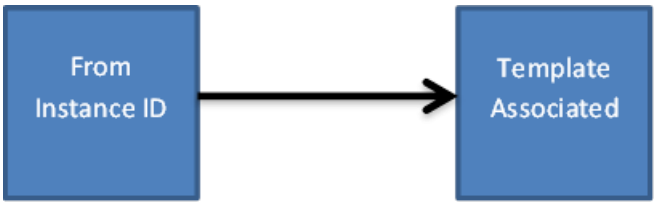


Figure 99 Scale In: get associated template

With the instance located the next step is to get all template children and check one by one whether it is a policy or not.

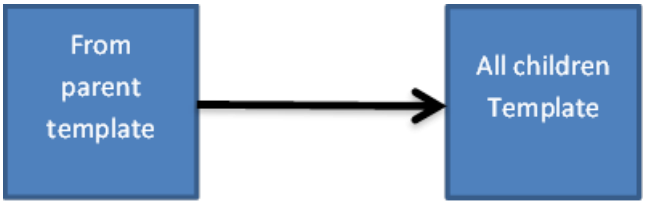


Figure 100 Scale In: get all template children

If it is not a policy, the template Id is gotten and the scale in operation is called to apply it recursively.

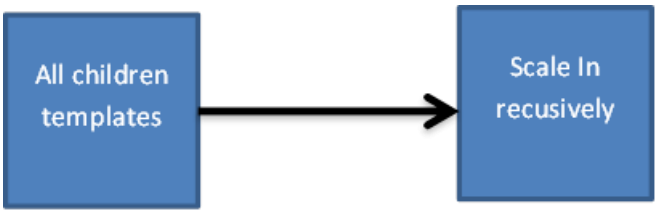


Figure 101 Scale In: Perform operation recursively

If it is a policy, the template id is gotten and all the children to get all the policies are called.

After policies evaluated and confirmed the process continues getting all the relationships between templates parents and children and calling create instance from template to create the new instances.

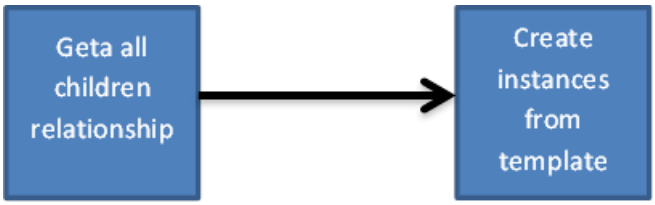


Figure 102 Scale In: Create new instance from template

Later, the operation gets all the relationships and checks the minimum range value of the scale in.

After that the scale in operation is finished.

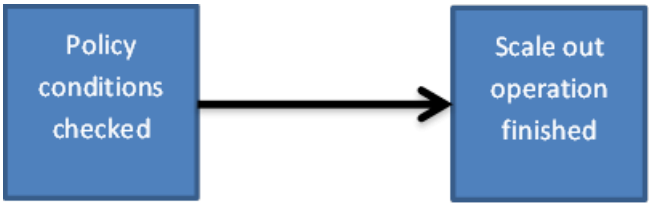


Figure 103 Scale In: Check policy

8.1.2 End-to-End Example

A Policy Node is available with DEFAULT=4, MIN=2 and DEFAULT_SCALE_IN=2. It also has OVER Parent Relationship to VIRTUAL_CORE.

The Artifact Template Tree is available with these relationships.

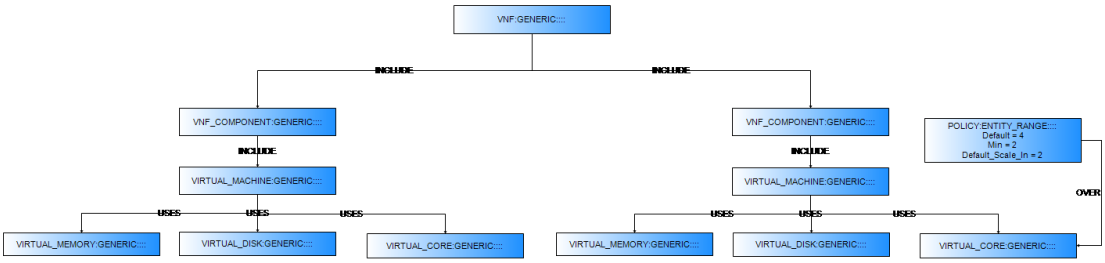


Figure 104 Scale In Example: Artifact Template Tree

Create the same Artifact Instance Tree with the same relationships and create 4 VIRTUAL_CORE Instances as policy index

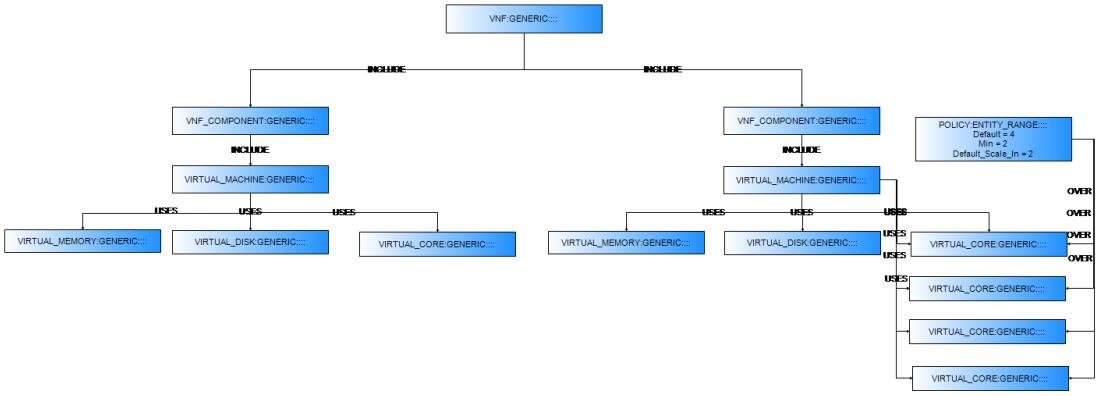


Figure 105 Scale In Example: Create Virtual Core Instance

The DEFAULT_SCALE_IN parameter is the number of instances you want to delete (scale).

The MIN parameter is the minimum of instances you must have.

You can get the Scale In operation if: MIN>=Instance amount in BBDD - DEFAULT_SCALE_IN.

Consider the parameters on the instance and not on the template.

In this case, 4 VIRTUAL_CORE instances depending on the POLICY are available.

MIN=2, DEFAULT_SCALE_IN=2 2<=4+2 √

The Scale In deletes 2 instances (DEFAULT_SCALE_IN parameter).

If the parameters do not satisfy the equation, the Scale In operation fails and does not delete any instance.

Apply the Scale In operation as a basic test.

Delete 2 instance children depending on the POLICY.

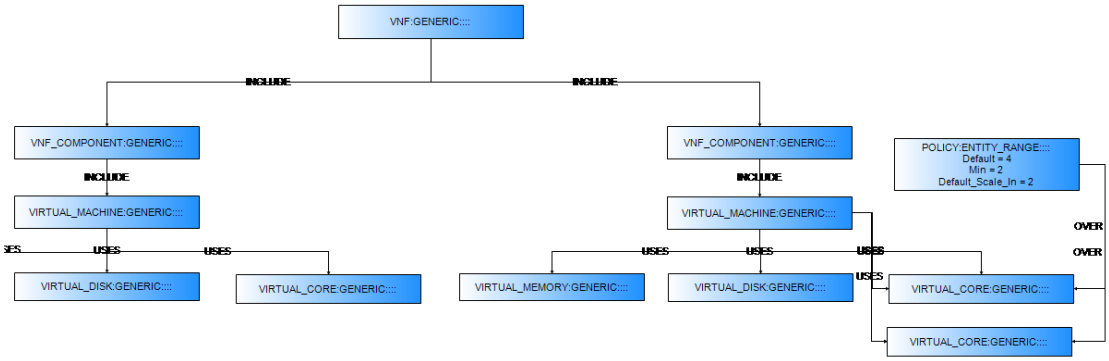


Figure 106 Scale In Example: Test Verification

8.1.3 Recap of End Messages

8.1.3.1 Errors

- 5001: ArtifactInstanceId is a mandatory input parameter.

- 5002: Artifact Instance with instanceId = %INPUT_INSTANCEARTIFACTID% does not exist in the system.
- 5003: Error delete instance from template.
- 5004: Recursive call failed.

8.1.3.2 Successful ends

- 0: Workflow ends ok.
- 0: OK. SCALE OUT Operation was not possible to do in all artifacts.

8.2 Scale-out

The scale out operation is built to generate, assign, and activate new resources on the instance tree. For example, if you have one virtual Machine with 1 child (Virtual Core), running the Scale Out operation increases the number of Virtual Cores according to the policy range.

Actually this operation can be called only from a VNF artifact.

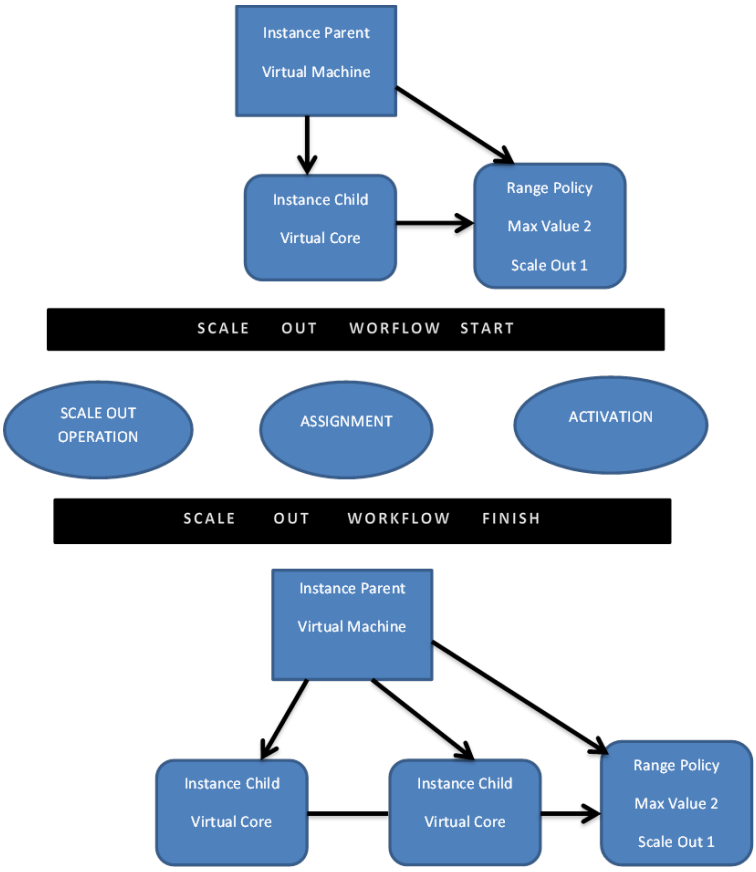


Figure 107 Scale Out

Particular case:

When the scale out workflow is called, it tries to scale as many elements as possible. If any element tries to scale above the maximum limit, it does not scale. However, the workflow continues scaling other components and displays a warning.

For example, consider a scenario where 2 virtual machine VNF starting as:

VM1 = 1 instance, VM2 = 1 instance, where

VM1 default=1, scale out=5, max= 2,

VM2 default=1, scale out=5, max= 10

If it tries to scale, the result is VM1 = 1, instance VM2 = 6 instance 1, because VM1 cannot scale above the maximum but VM2 can.

8.2.1 Scale-out operation

Scale out operation takes care of increasing the amount of resources.

It starts checking the Instance ID and getting the artifact associated to this ID. After getting the artifact the operation continues getting the template associated with the instance.

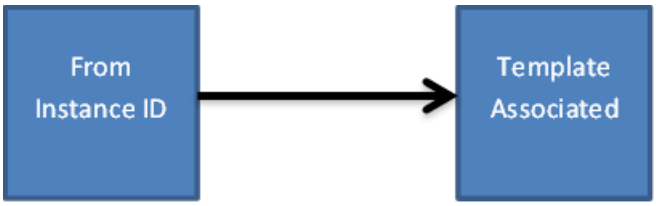


Figure 108 Scale out: query associated template

With the instance located the next step is to get all template children and check one by one whether it is a policy or not.

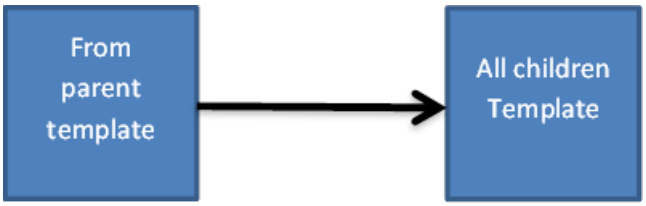


Figure 109 Scale out: query template children

If it is not a policy, the template Id is gotten and the scale out operation is called to apply it recursively.

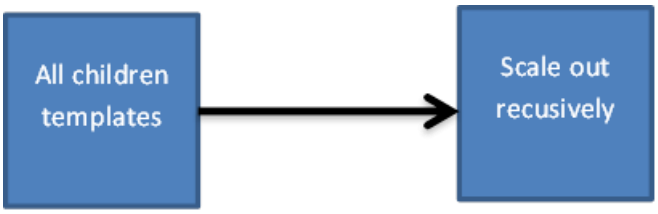


Figure 110 Scale out: apply scale out recursively

If it is a policy, the template id is gotten and all the children to get all the policies are called.

After policies evaluated and confirmed the process continues getting all the relationships between templates parents and children and calling create instance from template to create the new instances.

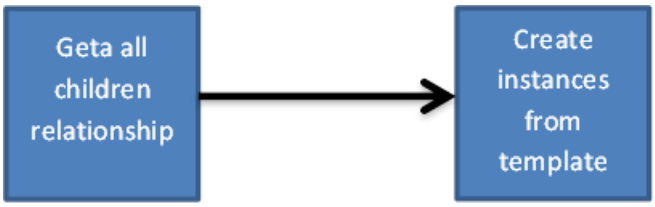


Figure 111 Scale out: create instances from template

Later, the operation gets all the relationships and checks the maximum range of the scale out.

After that the scale out operation is finished.

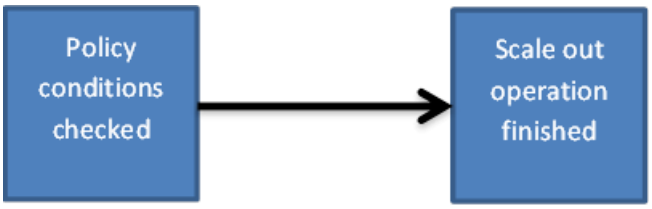


Figure 112 Scale out: operation completed

8.2.2 End-to-End Example

A policy node is available with DEFAULT=2, MAX=5, and DEFAULT_SCALE_OUT=2. Also, the node has OVER Parent Relationship to VIRTUAL_CORE.

This scenario has an Artifact Template Tree with the following relationships.

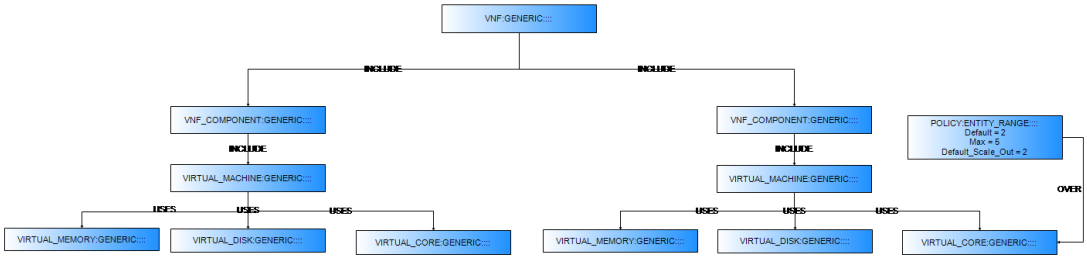


Figure 113 Scale out Example: Artifact template tree

Create Instance from the desired Template.

Create the same Artifact Instance Tree with the same relationships.

Even create 2 VIRTUAL_CORE Instances as policy index.

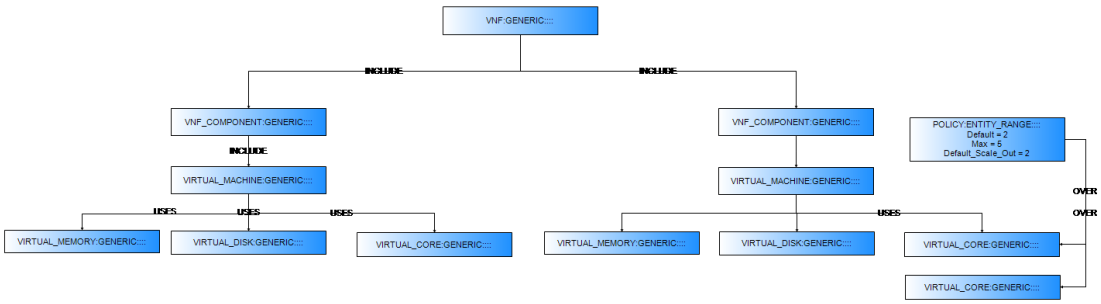


Figure 114 Scale out Example: Create virtual core

The DEFAULT_SCALE_OUT parameter is the number of instances you want to create (scale).

The MAX parameter is the maximum instances that you can have.

You can scale out only if MAX>=Instance amount in BBDD + DEFAULT_SCALE_OUT.

Consider the parameters on the instance and not on the template. This scenario has 2 VIRTUAL_CORE instances depending on the policy.

MAX=5, DEFAULT_SCALE_OUT=2 5>=2+2 √

The Scale Out creates 2 instances (DEFAULT_SCALE_OUT parameter). If the parameters do not satisfy the equation, the Scale Out operation fails and does not create any instance.

Apply the Scale Out operation as the basic test.

Create the 2 new instance child elements depending on the policy.

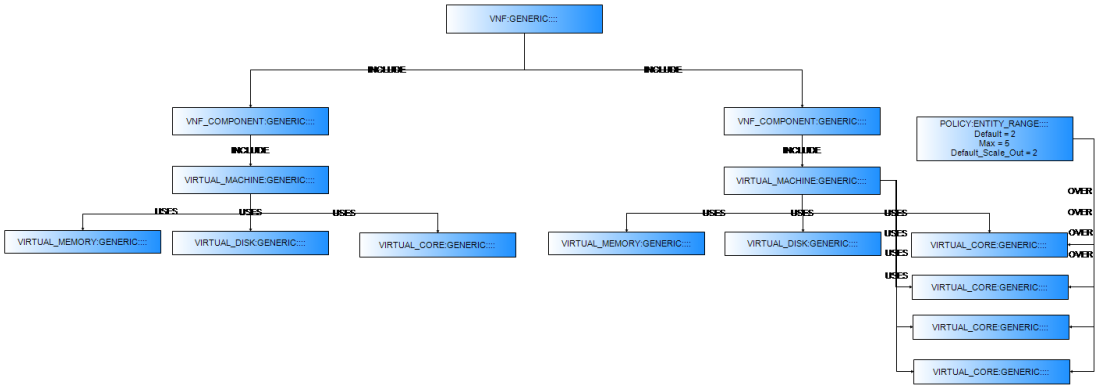


Figure 115 Scale out Example: Test verification

8.2.3 Recap of End Messages

8.2.3.1 Errors

- 4001: ArtifactInstanceId is a mandatory input parameter.
- 4002: Artifact Instance with instanceId = %INPUT_INSTANCEARTIFACTID% does not exist in the system.
- 4003: Scale out operation is only supported for instances created from template. InstanceId = %INPUT_INSTANCEARTIFACTID%.
- 4004: Recursive call failed.

- 4005: Malformed Template: The number of children of templateID = %VAR_TEMPLATE_PR.Id% is not 1.
- 4006: Malformed template: parent (%VAR_ARTIFACT_TEMPLATEID%) of the policy (%VAR_TEMPLATE_PR%), must be parent of the OVER Child (%VAR_TEMPLATE_PR_OVER_CHILD%) too.
- 4007: ERROR Create Instance From Template WF.
- 4008: ERROR Create New Child Relationship.

8.2.3.2 Successful Ends

- 0: Workflow ends ok.
- 0: OK. SCALE OUT Operation was not possible to do in all artifacts.

8.3 Scale Up

This operation enlarges attributes for an instance based on the entity-scale. It is necessary to have defined relationships that are required.

The operation has an impact both in DB and on the OpenStack side, changing the VM flavor, according to the attributes set in DB.

For example, if you have one virtual Machine with 3 children (Virtual Core, Virtual Disk and Virtual Memory), and the Virtual Memory is parent of the scale policy, running the Scale Up operation increases the content of Virtual Core Amount attribute according to the policy range.

On the OpenStack side, when the increase is successful, it tries to find a flavor that matches the DB amounts. If it finds it, it automatically changes the flavor of the VM.

8.3.1 Scale Up operation

The purpose of Scale Up operation increases the quantity of the attribute INFO.Amount allocated in the VIRTUAL_CORE, VIRTUAL_MEMORY and VIRTUAL_DISK.

The increment is defined by the scale policy in its attribute SCALE:INCREASEAMOUNT. This policy has to be child of the artifact ready to scale.

Once the scale is done in DB, immediately it looks for a flavor in the VIM (Icehouse, CS8, ...) according to the amount of core, memory and disk in DB and if it finds it, it changes the VM flavor.

As we have seen, this operation has an implicit restart over VM. This process will check the state of monitors and VMs in case it needs to be re-launched.

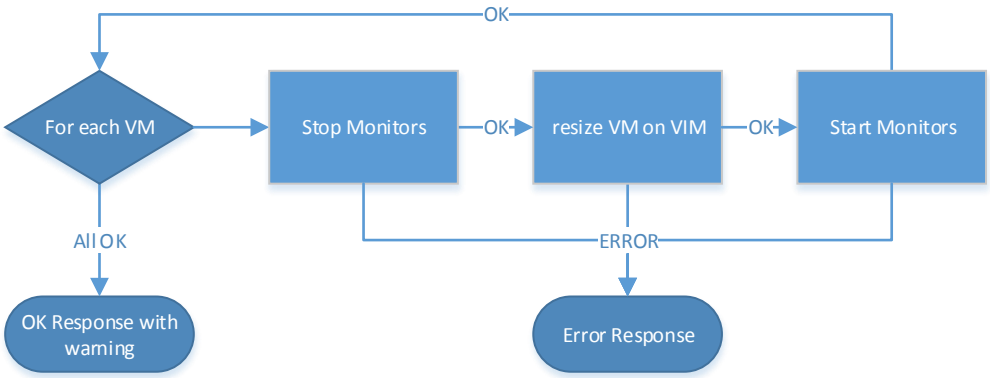


Figure 116 Scale up flow

Next, the details of the completion will be listed.

On the DB side, the operation starts looking for the VNF and sets its status as “LOCKED”.

Then it looks for scale policies. When it finds one, it gets the MAX and INCREASEAMOUNT attribute values in order to check if the scale is possible, validating through this equation:

CURRENT_VALUE (INFO.AMOUNT of artifact to scale)+INCREASEAMOUNT<=MAX

Next, it checks if there are free physical resources to allocate and validates the new value with WF_NFVD_INSTANCE_VALIDATION operation.

Finally, it sets the VM status to “TO_BE_SCALE”.

When the scale in DB ends OK, it starts the activation side.

On this side, first it queries the VMs to change the flavor.

Then, it looks for the VIRTUAL_DATACENTER or TENANT and gets GENERAL.Name (tenant name) and stops the monitor if it is possible.

For each VM, it updates LAST_OPERATION attribute to “resize”, and gets the VIM artifact to extract the URL, user and pass of the VIM.

Immediately, it launches the WorkFlow that resizes the flavor and sets the result of this operation in the VM attributes called LAST_OPERATION.Result_code and LAST_OPERATION.Result_description.

Finally, it starts the monitors and sets the VNF status to “ENABLE”.

8.3.2 End-to-End Example

A scale-policy node is available with MAX=2048, INCREASEAMOUNT=512, and DESTINY=INFO.Amount. Also, the node has APPLY Child Relationship to VIRTUAL_MEMORY.

This scenario has a VNF Tree with the following relationships.

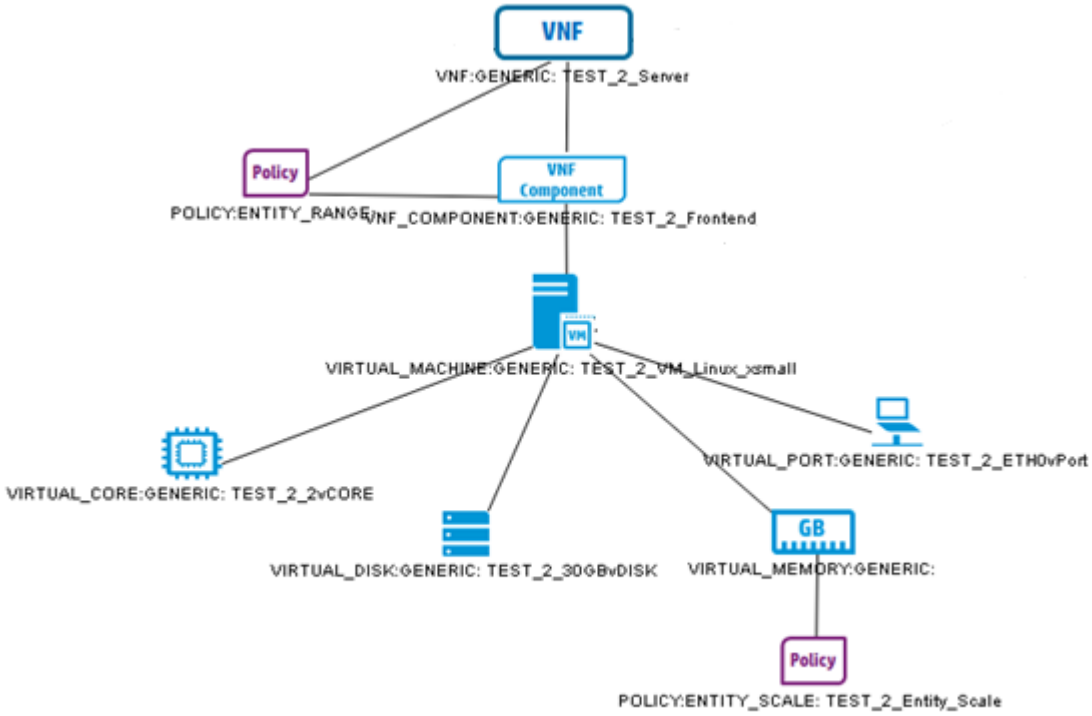


Figure 117 VNF tree for scale up

The VIRTUAL_MEMORY, belonged to the VM, in DB has 512mb of memory (INFO.Amount).

The VIM, Icehouse in this case, has the VM activated with a flavor of 512mb memory.

☐	Instance Name	Image Name	IP Address	Size
☐	test	cirros_new	16.17.101.19	1vcpu_512RAM_1_Disk | 512MB RAM | 1 VCPU | 1.0GB Disk

Figure 118 Memory amount instance

Scale Up operation from the inventory tree.

The results are the increase of 512mb in DB.

Even the resized flavor in the VIM (Icehouse).

☐	Instance Name	Image Name	IP Address	Size
☐	test	cirros_new	16.17.101.19	1vcpu_1024RAM_1_Disk | 1GB RAM | 1 VCPU | 1.0GB Disk

Figure 119 Scale up: resize flavour

8.3.3 Recap of End Messages

8.3.3.1 Errors

WF_NFVD_SCALE_UPDOWN_ACTIVATION

- 16001: Mandatory input parameter ArtifactInstanceId is not present

- 16002: Exist more than 1 tenants
- 16003: Exist more than 1 Virtual Datacenters16004: Recursive call failed.
- 16005: Dont exist VirtualDatacenter or Tenant
- 16006: Dont exist VNF or exists more than 116007: ERROR Create Instance From Template WF.
- 16009: Stop monitor was not ok.
- 16010: Start monitor was not ok
- 16011: Change Flavor was not ok

WF_NFVD_SCALE_UPDOWN_INVENTORY

- 17001: Mandatory input parameter ArtifactInstanceId is not present
- 17002: Instance Artifact Id dont exist in system
- 17003: Mandatory input parameter INPUT_OPERATION is not present or is not valid
- 17004: Dont exist VNF or exists more than 1
- 17005:Min limit exceeded
- 17006: Max limit exceeded
- 17007: It can't be allocated
- 17008:Exist more than VM Parent

WF_NFVD_SCALE_UPDOWN_INVENTORY

- 15001: Mandatory input parameter ArtifactInstanceId is nor present
- 15002:No Flavor Found
- 15003: ERROR Stopping the server
- 15004: ERROR Starting VM
- 15005: ERROR INCONSISTENT
- 15006: ERROR Resizing the server
- 15007: VM has not a valid state
- 15008: ERROR Confirming the resize of the server
- 15009: ERROR Starting VM
- 15010: Query flavor was not ok
- 15011:Query flavors was not ok
- 15012: Stop server was not ok
- 15013:Start server was not ok
- 15014:Revert resize was not ok
- 15015: Starting server was not ok in the rollback

8.3.3.2 Successful Ends

- 0: End OK.
- 0: End OK. No scale policies to execute

8.4 Scale Down

This operation decreases attributes for an instance based on the entity-scale policy. It is necessary to have defined relationships that are required.

It is the opposite operation of Scale Up.

The operation has an impact both in DB and on the OpenStack side, changing the VM flavor, according to the attributes set in DB.

On the OpenStack side, when the decrease is successful, it tries to find a flavor that matches the DB amounts. If it finds it, it automatically changes the flavor of the VM.

8.4.1 Scale Down operation

The description of the Scale Up is exactly the same as the Scale Down, but the result is a decrease of the amounts. To achieve the decrease, it is necessary to accomplish this equation:

CURRENT_VALUE (INFO.AMOUNT of artifact to scale)+DECREASEAMOUNT>=MIN

8.4.2 End-to-End Example

A scale policy node is available with MIN=512, DECREASEAMOUNT=512, and DESTINY=INFO.Amount. Also, the node has APPLY Child Relationship to VIRTUAL_MEMORY.

This scenario has a VNF Tree with the following relationships.

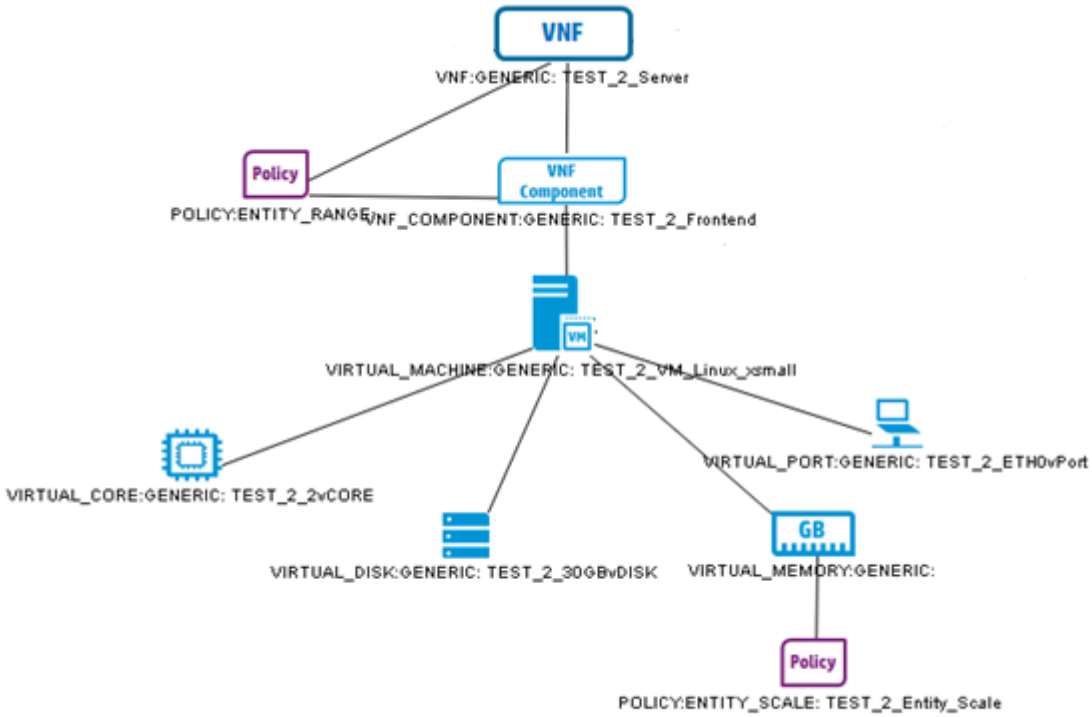


Figure 120 VNF tree for scale down

The VIRTUAL_MEMORY, belonged to the VM, in DB has 1024mb of memory (INFO.Amount).

The VIM, Icehouse in this case, has the VM activated with a flavor of 1024mb memory.

	Instance Name	Image Name	IP Address	Size
☐	test	cirros_new	16.17.101.19	1vcpu_1024RAM_1_Disk | 1GB RAM | 1 VCPU | 1.0GB Disk

Figure 121 Memory amount instance

Scale Down operation from the inventory tree.

The results are the decrease of 512mb in DB.

Even the resized flavour in the VIM (Icehouse).

	Instance Name	Image Name	IP Address	Size
☐	test	cirros_new	16.17.101.19	1vcpu_512RAM_1_Disk | 512MB RAM | 1 VCPU | 1.0GB Disk

Figure 122 Scale down: resize flavour

8.4.3 Recap of End Messages

8.4.3.1 Errors

The same that the Scale Up operation.

8.4.3.2 Successful Ends

The same that the Scale Up operation.

Chapter 9

Network Service

From VNF Director v2.0 is included the possibility to share networks between different VNFs.

A Network Service is a complex template that includes different VNFs and modeling how these VNFs are connected between them.

The basic scenario for Network Services is show in next figure:

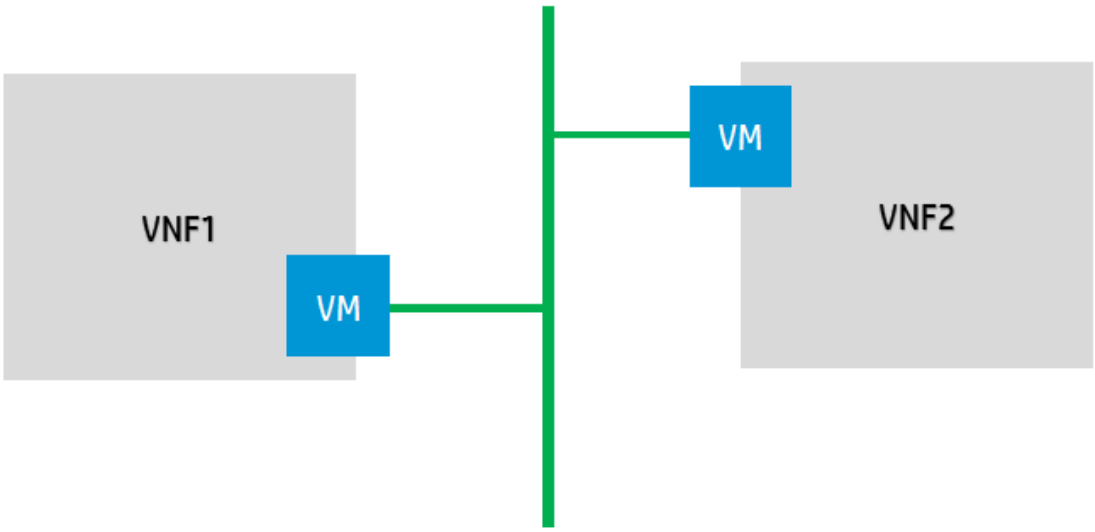
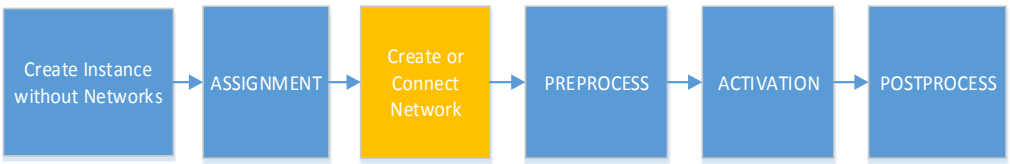


Figure 123 Two VNF connected using one network

The possibility to share networks between different VNFs is defined on templates, setting property `CREATION_MODE. Connect` to `true` on NETWORKS, SUBNETWORKS and IPADDRESS.

This process is similar to the other creation process; the only difference is that creation of Network instance will be delayed until the VM is already assigned. Additional step is added between assignment and activation. This step is responsible to create new instances of networks or connect the VNF to an existing network.



This section describes the scenarios and operations with Network Service. The steps are similar to create a VNF with network. In this section, an explanation of how to model and deploy is given.

9.1 Network Service

A Network Service (NS) is a complex template that includes different VNFs and modeling connections between them. A simple example is the following. We have a template of a NS that contains 2 VNFs, and the same template defines the connectivity between them.

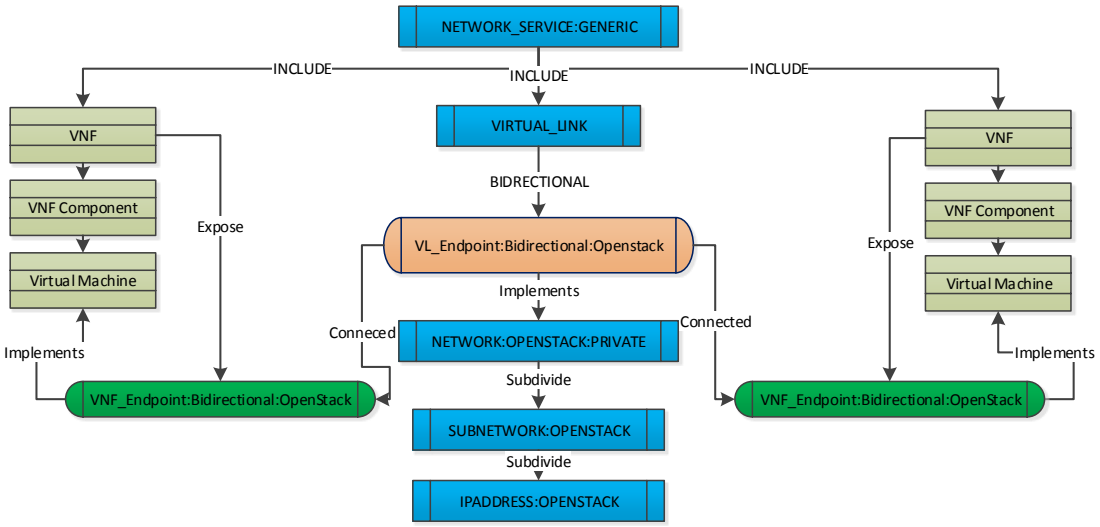


Figure 124 Two VNF connected to the same network

Each VNF must contain a VNF_ENDPOINT; this is the "interface" that the VNF exposes its connectivity outside. And this end point is implemented by one element of that VNF, in our example is a Virtual Machine. This endpoint may connect with VL_Endpoint of any VIRTUAL_LINK, and in same way this endpoint has an implementation, for us an OpenStack Network.

These joins of end points means that the process must understand what it needs to do to make a real connection, for example, a network will create if it does not exist under a Tenant (OpenStack) and a new VIRTUAL_PORT will be created and allocated on this Network through a relationship from IPADDRESS and this new Virtual port.

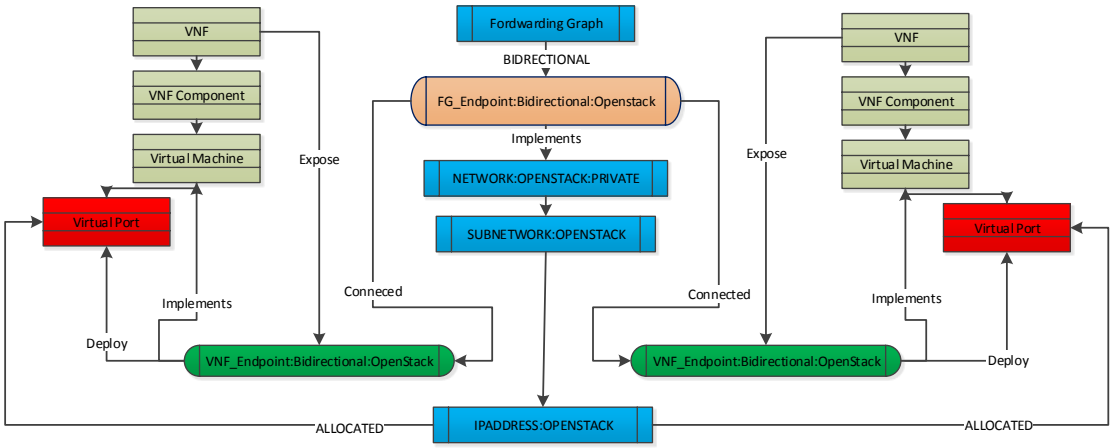


Figure 125 Two VNF connected to the same network: Virtual Port

9.2 Supported Scenarios

Currently NFVD supports three basic scenarios along with any combinations of them.

9.2.1 Two VMs in same network

This is the basic scenario, where some VNF has got an endpoint implemented by a Virtual Machine, and all of these VNFs must be in same network. This network is the implementation of the endpoint of a Virtual Link.

Same example is explained above.

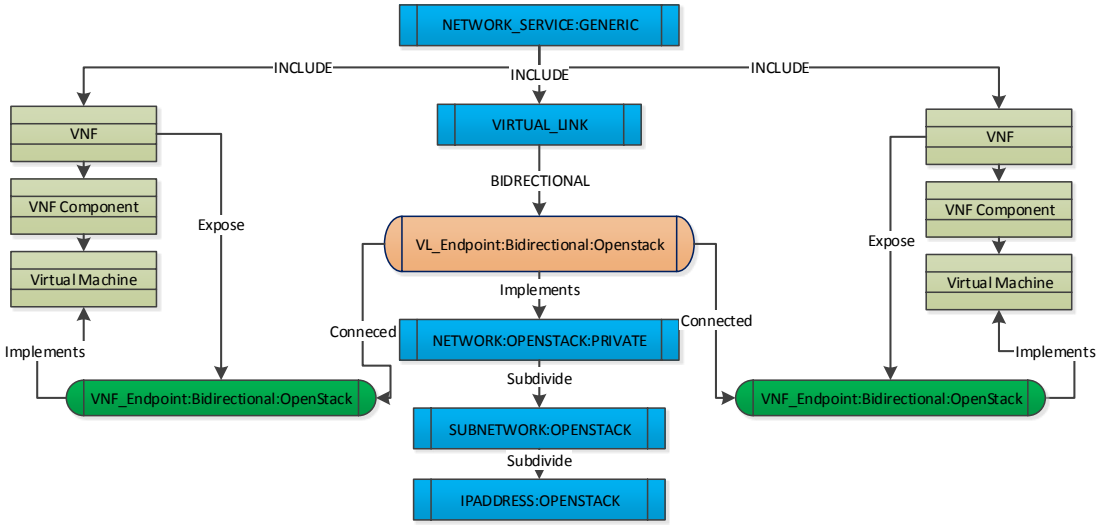


Figure 126 Two VNF connected to the same network

The result of this Network Service is:

- one network is instantiated
- two VMs instantiated, attaches to same network

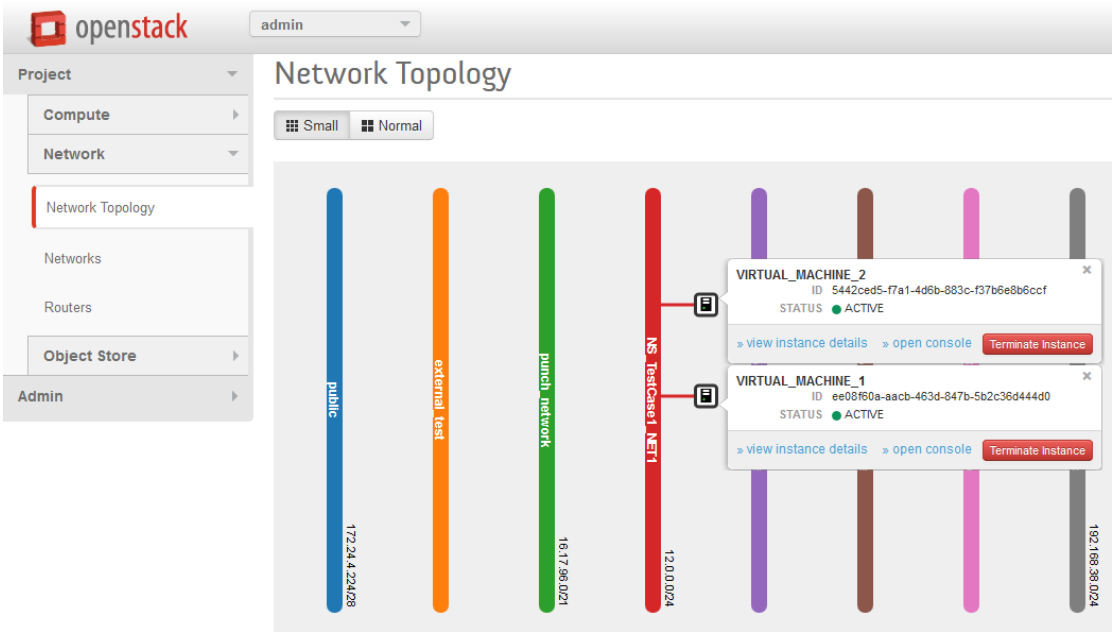


Figure 127 Two VNF connected to the same network - VIM

9.2.2 Two VMs connected through a third one

This scenario is to create connectivity between two VNFs that are not directly connected between them, and is necessary to redirect the traffic through another VNF.

This scenario has two VNFs that have got an endpoint implemented by a Virtual Machine. VMs of each VNF are connected to a different network. A third VNF that has a VM connected to both networks offers the connectivity between other VNFs.

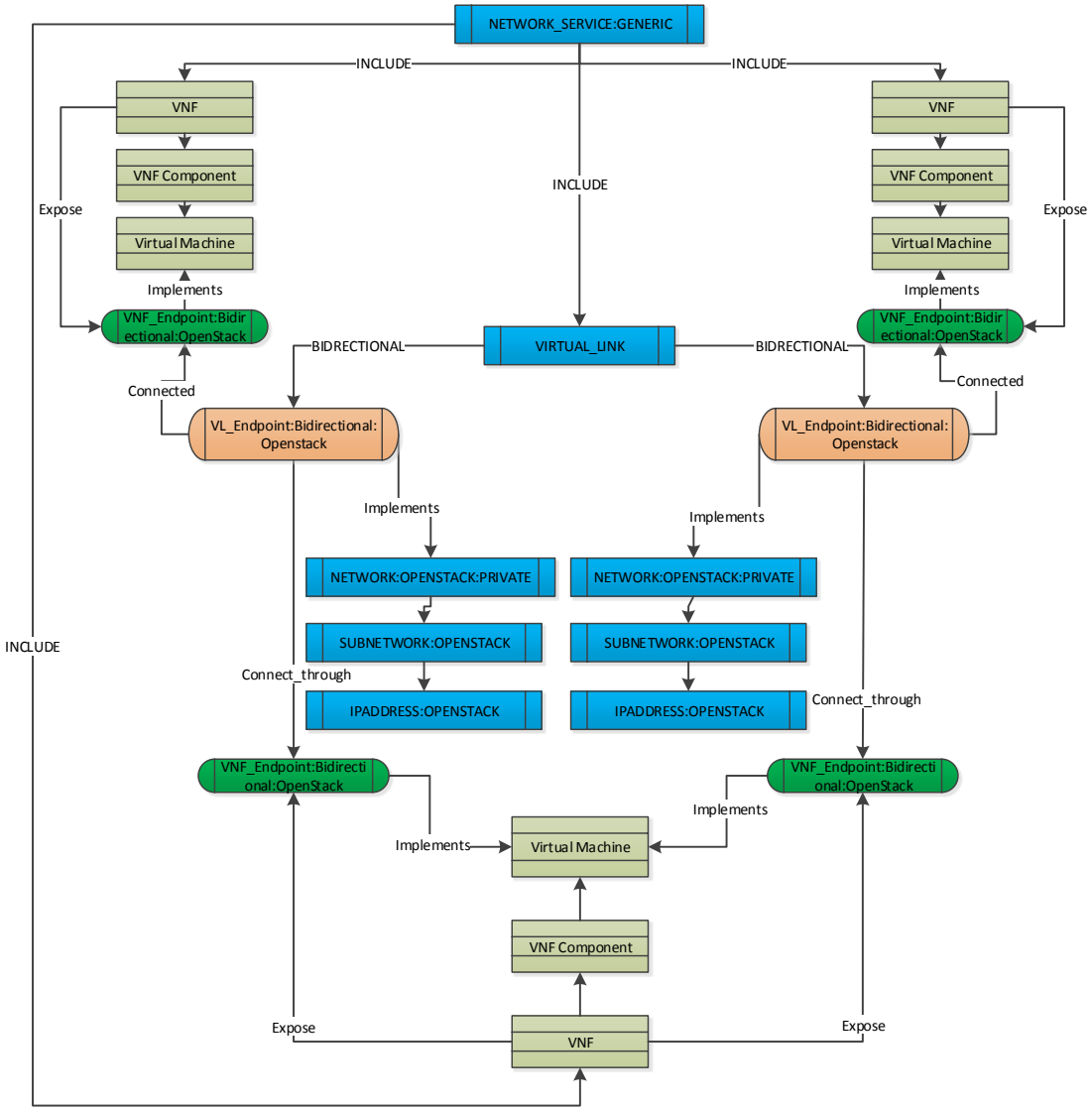


Figure 128 Two VNF connected through a third VM

The result of this Network Service is:

- Two networks instantiated
- First VM instantiated and attaches network 1
- Second VM instantiated and attaches network 2
- Third VM instantiated and gets attached to both networks

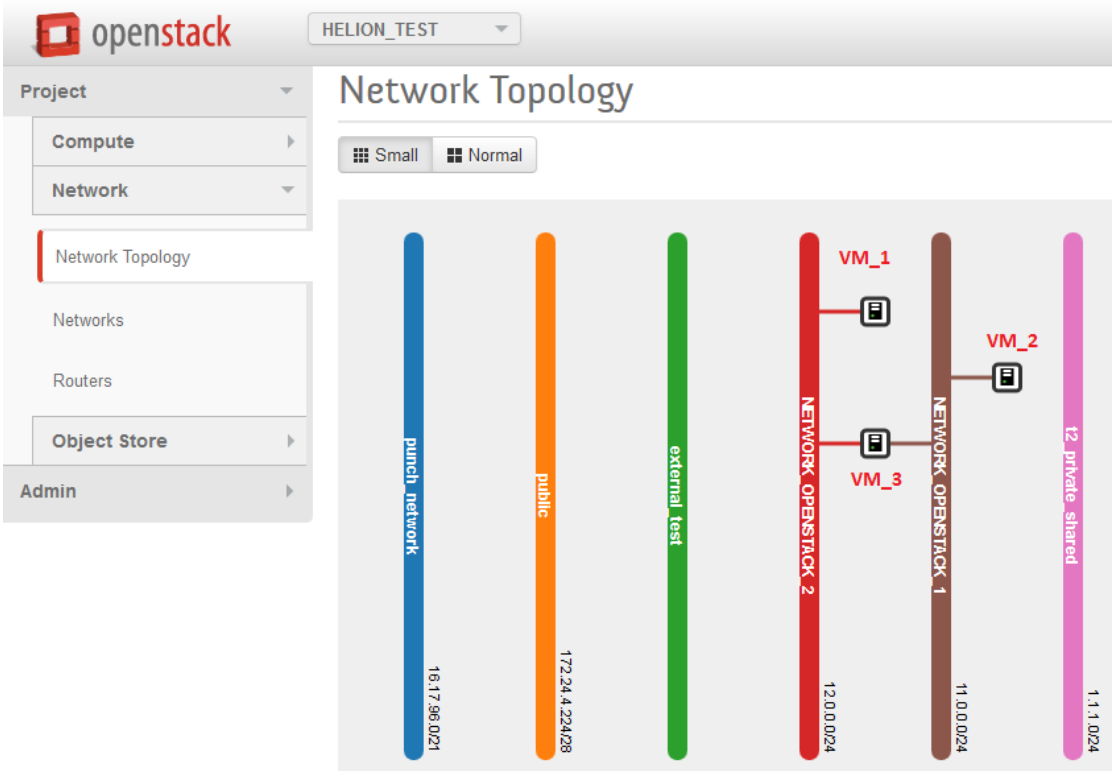


Figure 129 Two VNF connected through a third VM - VIM

9.2.3 Two VMs connected with an external network through a VM

This scenario models the connectivity between two VMs without direct connectivity but using another network that is external from these VNFs. In this case, it is necessary to go through two other VMs that complete the connectivity.

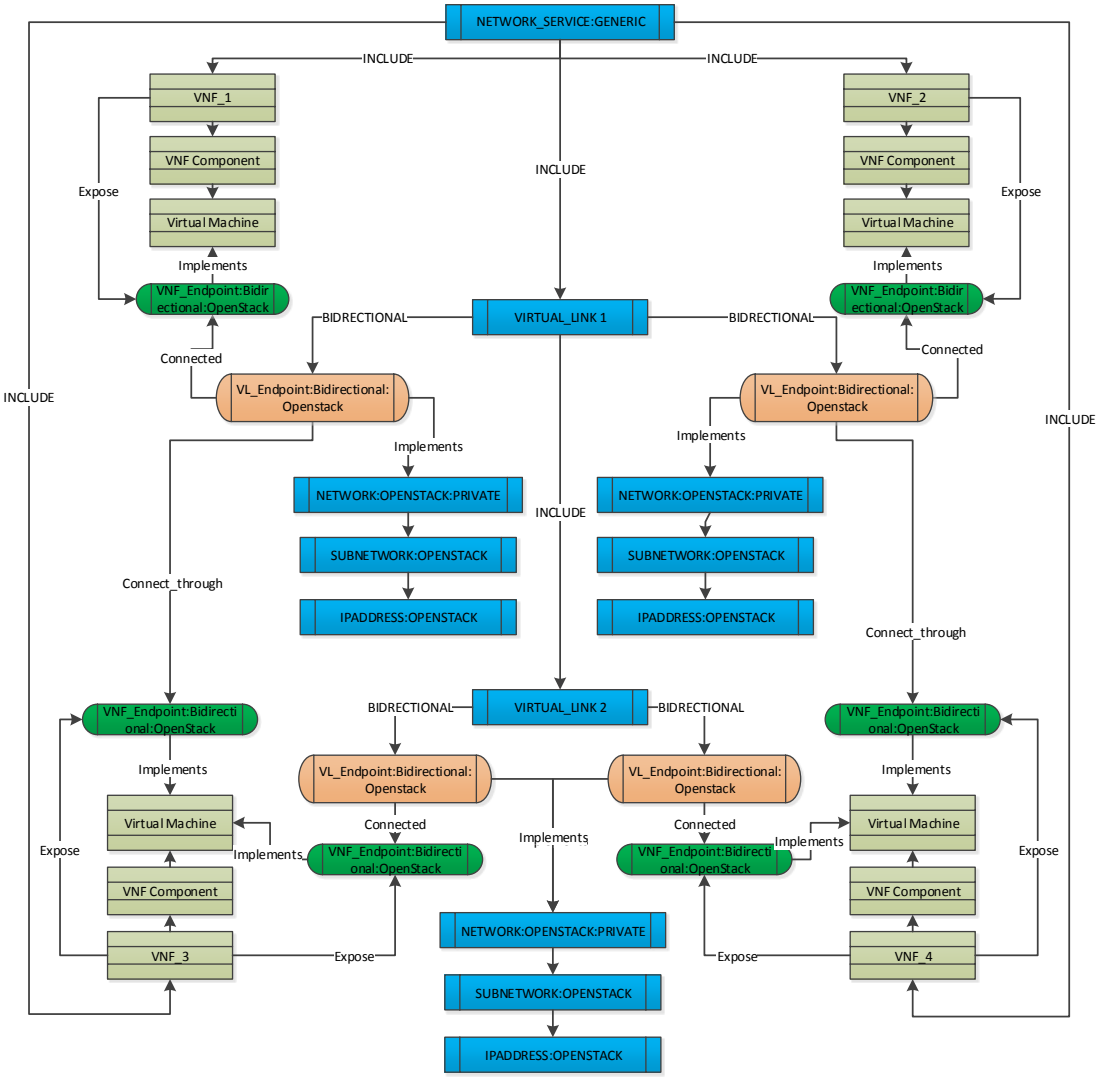


Figure 130 Two VNF connected with an external network through VM - VIM

The result of this Network Service is:

- Three networks instantiated
- First VMs instantiated and attaches network 1
- Second VM instantiated and attaches network 2
- Third VM instantiated and attached to network 1 and 3
- Fourth VM instantiated and attached to network 2 and 3

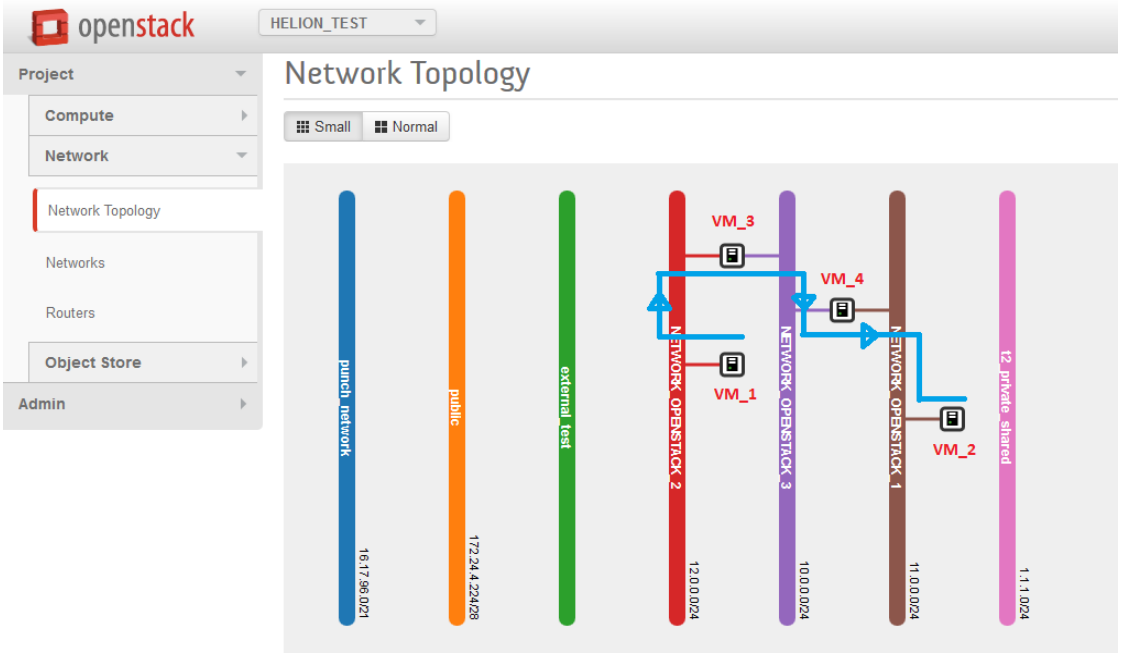


Figure 131 Two VNF connected with an external network through VM - VIM

Chapter 10

Administering NFV Director Components

This chapter describes the procedure to manage or administer various components of NFV Director.

10.1 Stopping/Starting NFV Director

The `nfv-director.sh` script starts and stops the NFV-Director. The script is available in the `<Base RPM Install Path>/opt/HP/nfvd/bin` directory.

Script usage:

```
nfv-director.sh:
```

Usage:

```
nfv-director.sh [OPTIONS...]
 -a start | stop | restart | status
 [-c] [ activator | sosa | ecpool | lockmgr | ppasdb |
openmediation | SiteScope | uca-ebc | uca-atm | nfvd-agw
]
```

To get help on `nfvd-director.sh`:

```
#nfv-director.sh -h
```

```
#./nfv-director.sh -h

Usage: nfv-director.sh [OPTIONS...]
 -a start | stop | restart | status
 [-c] [ activator | sosa | ecpool | lockmgr | ppasdb | openmediation | sitescope | uca-ebc | uca-atm | nfvd-agw ]
#
```

Note

The default stop script works for components installed in the default location. Otherwise, the user should run the start/stop commands provided by the individual components in their respective installation directories.

Alternate commands are provided wherever applicable if the product is installed in non-default locations.

The following is the list of components start/stop in order when the script is run.

1. HP Service Activator(HPSA)
2. HP Service Order Smart Adapter (SOSA)
3. HP Equipment connection pool (ECP)
4. HP Lock Manager
5. HP Open Mediation
6. HP SiteScope
7. HP UCA-EBC
8. HP UCA Automation Console
9. HP NFVD Assurance Gateway

10.1.1 Starting all components

To start all components, run the following command:

```
#nfv-director.sh -a start
```

10.1.2 Stopping all NFV-Director components

To stop all NFV-Director components, run the following command:

```
#nfv-director.sh -a stop
```

10.1.3 Getting status of NFV-D components

To check status of NFV-D components, run the following command:

```
#nfv-director.sh -a status
```

10.1.4 Restarting NFV-Director components

To restart NFV-Director components, run the following command:

```
#nfv-director.sh -a restart
```

10.2 Fulfillment components

10.2.1 Starting the Activator

To start the activator, run the following command:

```
#nfv-director.sh -a start -c activator
```

Alternate command:

```
/etc/init.d/activator start
```

10.2.2 Getting status of the Activator

To check activator status, run the following command:

```
#nfv-director.sh -a status -c activator
```

Alternate command:

```
/etc/init.d/activator check
```

10.2.3 Stopping the activator

To stop the activator, run the following:

```
#nfv-director.sh -a stop -c activator
```

Alternate command:

```
/etc/init.d/activator stop
```

10.2.4 Starting HP SOSA

To start the HP SOSA, run the following command:

```
#nfv-director.sh -a start -c sosa
```

Alternate Command:

```
/opt/OV/ServiceActivator/EP/SOSA/bin/sosa.sh start
```

10.2.5 Getting status of HP SOSA

To check the HP SOSA status, run the following command:

```
#nfv-director.sh -a status -c sosa
```

Alternate command:

```
/opt/OV/ServiceActivator/EP/SOSA/bin/sosa.sh test
```

10.2.6 Stopping HP SOSA

To stop HP SOSA, run the following command:

```
#nfv-director.sh -a stop -c sosa
```

Alternate command:

```
/opt/OV/ServiceActivator/EP/SOSA/bin/sosa.sh stop
```

10.2.7 HP Equipment connection pool

To start the HP Equipment connection pool (ECP), run the following command:

```
#nfv-director.sh -a start -c ecpool
```

Alternate Command:

```
/opt/OV/ServiceActivator/EP/ECP/bin/StartServer.sh
```

10.2.8 Stopping HP ECP

To stop HP Equipment connection pool (ECP), run the following command:

```
#nfv-director.sh -a stop -c ecpool
```

Alternate Command:

```
/opt/OV/ServiceActivator/EP/ECP/bin/StopServer.sh
```

10.2.9 Getting status of HP ECP

To check HP Equipment connection pool (ECP) status, run the following command:

```
#nfv-director.sh -a status -c ecpool
```

Alternate Command:

```
/opt/OV/ServiceActivator/EP/ECP/bin/showStatus.sh
```

10.2.10 Starting HP Lock Manager

To start HP Lock Manager, run the following command:

```
#nfv-director.sh -a start -c lockmgr
```

Alternate command:

```
/opt/OV/ServiceActivator/EP/LockManager/bin/StartServer.s
h
```

10.2.11 Getting status of HP Lock Manager

To Check HP Lock Manager Status, run the following command:

```
#nfv-director.sh -a status -c lockmgr
```

Alternate command:

```
/opt/OV/ServiceActivator/EP/LockManager/bin/showStatus.sh
```

10.2.12 Stopping HP Lock Manager

To stop HP Lock Manager, run the following command:

```
#nfv-director.sh -a stop -c lockmgr
```

Alternate command:

```
/opt/OV/ServiceActivator/EP/LockManager/bin/StopServer.sh
```

10.3 Assurance components

10.3.1 Starting Assurance Gateway

To start NFVD Assurance Gateway, run the following command:

```
#nfv-director.sh -a start -c nfvd-agw
```

10.3.2 Stopping Assurance Gateway

To stop NFVD Assurance Gateway, run the following command:

```
#nfv-director.sh -a stop -c nfvd-agw
```

10.3.3 Getting status of Assurance Gateway

To check status of NFVD Assurance Gateway, run the following command:

```
#nfv-director.sh -a status -c nfvd-agw
```

10.3.4 Restarting NFVD Assurance Gateway

To restart NFVD Assurance Gateway, run the following command:

```
#nfv-director.sh -a restart -c nfvd-agw
```

10.3.5 Starting Open Mediation

To start Open Mediation, run the following command:

```
#nfv-director.sh -a start -c openmediation
```

Alternate command:

```
/opt/openmediation-V62/bin/nom admin --start-container -
all
```

10.3.6 Stopping Open Mediation

To stop Open Mediation, run the following command:

```
#nfv-director.sh -a stop -c openmediation
```

Alternate command:

```
/opt/openmediation-V62/bin/nom admin --shutdown-container
-all
```

10.3.7 Getting status of Open Mediation

To check status of Open Mediation, run the following command:

```
#nfv-director.sh -a status -c openmediation
```

Alternate command:

```
/opt/openmediation-V62/bin/nom_admin --list-container
```

10.4 Monitoring components

10.4.1 Starting HP SiteScope

To start HP SiteScope, run the following command:

```
#nfv-director.sh -a start -c SiteScope
```

Alternate command:

```
/opt/HP/SiteScope/start
```

10.4.2 Stopping HP SiteScope

To Stop HP SiteScope, run the following command:

```
#nfv-director.sh -a stop -c SiteScope
```

Alternate command:

```
/opt/HP/SiteScope/stop
```

10.4.3 Getting status of HP SiteScope

To check status of HP SiteScope, run the following command:

```
#nfv-director.sh -a status -c SiteScope
```

10.4.4 Restarting HP SiteScope

To restart HP SiteScope, run the following command:

```
#nfv-director.sh -a restart -c SiteScope
```

10.5 Automation components

10.5.1 Starting HP UCA-EBC

To start HP UCA-EBC, run the following command:

```
#nfv-director.sh -a start -c uca-ebc
```

Alternate command:

```
su - uca /opt/UCA-EBC/bin/uca-ebc start
```

10.5.2 Getting status of HP UCA-EBC

To check the status of HP UCA-EBC, run the following command:

```
#nfv-director.sh -a status -c uca-ebc
```

Alternate command:

```
su - uca /opt/UCA-EBC/bin/uca-ebc show
```

10.5.3 Stopping HP UCA-EBC

To stop HP UCA-EBC, run the following command:

```
#nfv-director.sh -a stop -c uca-ebc
```

Alternate command:

```
su - uca /opt/UCA-EBC/bin/uca-ebc stop
```

10.5.4 Starting HP UCA-automation

To start HP UCA-Automation, run the following command:

```
#nfv-director.sh -a start -c uca-atm
```

Alternate command:

```
/opt/UCA-ATM/bin/ucautomation-ui start
```

10.5.5 Getting status of HP UCA-automation

To check status of HP UCA-Automation, run the following command:

```
#nfv-director.sh -a status -c uca-atm
```

Alternate command:

```
su - uca /opt/UCA-ATM/bin/ucautomation-ui show
```

10.5.6 Stopping HP UCA-automation

To stop HP UCA-Automation, run the following command:

```
#nfv-director.sh -a stop -c uca-atm
```

Alternate command:

```
/opt/UCA-ATM/bin/ucautomation-ui stop
```

10.6 NFVD GUI Components

10.6.1 Starting UOC

To start UOC, run the following command as uoc user

```
$ /opt/uoc2/bin/uoc2 start
```

10.6.2 Getting status of UOC

To check status of UOC , run the following command as uoc user

```
$ /opt/uoc2/bin/uoc2 show
```

10.6.3 Stopping UOC

To stop UOC, run the following command as uoc user

```
$ /opt/uoc2/bin/uoc2 stop
```

Chapter 11

VNF Manager

There are two main models of interaction between NFV Orchestrator, VNF Manager and VIMs.

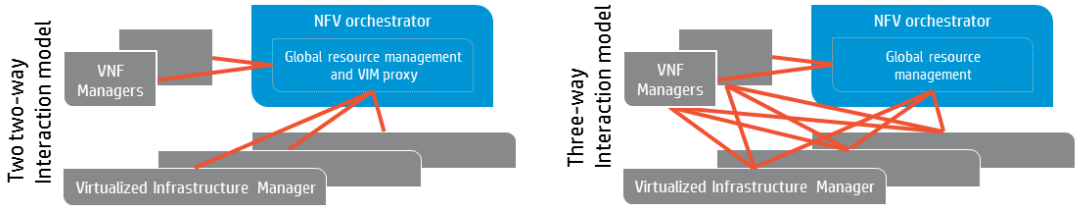


Figure 132 VNF Manager

The first model is a two two-way interaction model (Indirect VNF Manager). In this model, VNF Manager uses VNF Orchestrator as proxy for VIM and allows VNF Orchestrator to perform additional functions beyond actions of VIM (e.g., provisioning assurance).

The second model is a three way interaction model (Direct VNF Manager). In this case, the VNF Orchestrator grants permission to the VNF Manager to perform particular (resource-affecting) lifecycle operations.

Optionally, in this second model, the NFV Orchestrator provides resource reservation and the VNF Manager interacts directly with each VIM, and the synchronization is required.

HP NFV Director supports both models and can perform both Direct VNF Manager as Indirect VNF Manager actions.

11.1 Interaction modes

11.1.1 Indirect VNF Manager

The following figure shows indirect mode for NFV Director interaction with VNF Managers:

Figure 133 VNF Indirect Manager

The steps in this mode are:

- The VNF manager asks NFV Director to perform every action (create VM, scale, VNF, monitor, etc ...)

- NFV director inform the VNF manager when each operation finishes
- NFV director has the capacity consumption detail of each component

11.1.2 Direct VNF Manager

Next figure illustrates direct mode interaction:

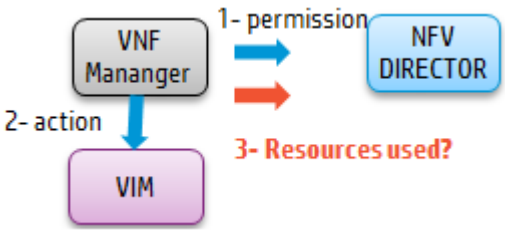


Figure 134 VNF Direct Manager

Steps in this mode are:

- The VNF manager asks NFV Director only for permission.
- The VNF managers perform every action (create VM, scale, VNF, monitor, etc ...)
- VNF manager informs NFV Director when each operation finishes
- NFV director would have the capacity consumption detail of each component ONLY if the VNF manager informs NFV director

11.1.3 Direct VNF Manager (Proxy)

The following figure shows direct mode (proxy mode):

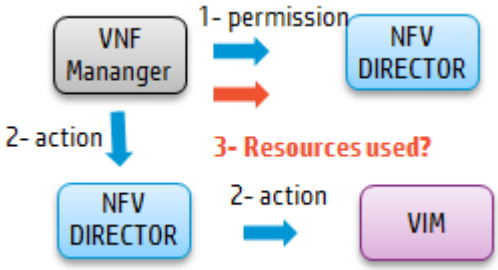


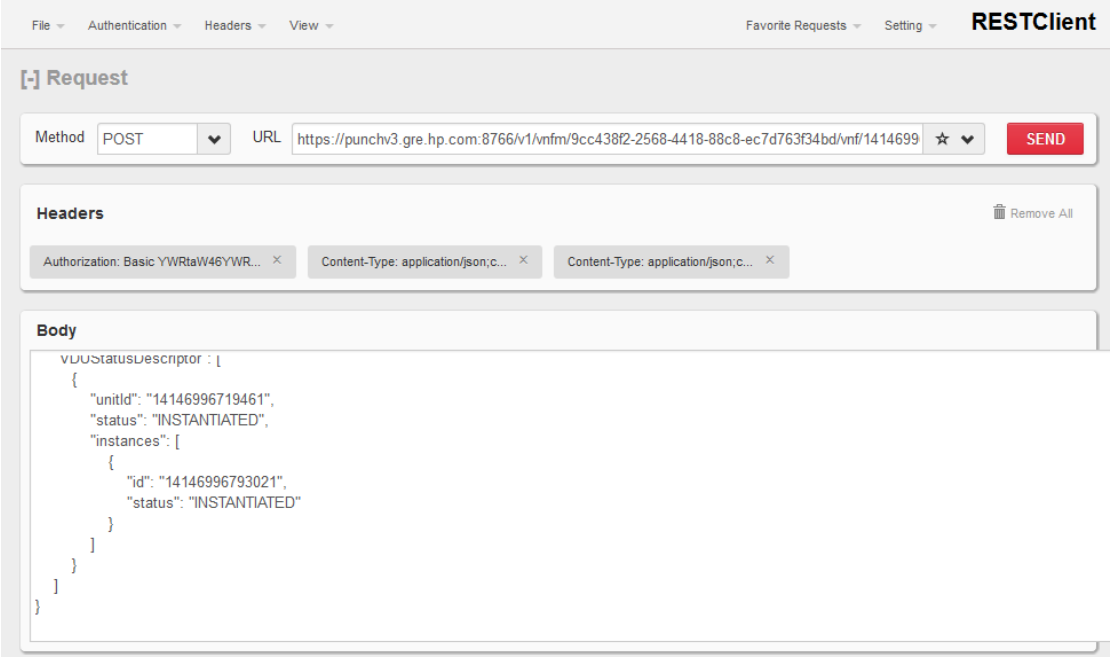
Figure 135 VNF Direct Manager (Proxy)

Steps in this mode are:

- The VNF manager may ask NFV Director only for permission
- The VNF managers perform every action (create VM, scale, VNF, monitor, etc ...) but they point to NFV Director instead to the VIM
- NFV Director to perform every Openstack action (create VM, scale, VNF, monitor, etc ...)
- NFV director would have the capacity consumption detail of each component ONLY if the VNF manager informs NFV director (it will have the detail but would not be possible to associated the consumption to an specific component without extra information

11.2 VNF Manager

A VNF Manager is an external entity which is capable of performing multiple activating tasks. As a consequence, a VNF Manager requires NFVD services as Orchestrator. The



Manager will ask for activation permissions to NFVD, which will respond with a concrete VIM and credentials. In the overall process, NFVD would model an inventory representation of the VNF to be activated by VNF Manager.

Figure 136 VNF Manager interaction from RESTClient

11.3 NFV Director - VNF Manager Interactions

These are the ways VNFM and NFVD interact.

11.3.1 VNF Manager Protocol Adapter: from VNFM to NFVD

Protocol Adapter communicates to the manager and NFVD through a REST protocol.

This Protocol Adapter expects POST / PUT / GET / DELETE operations defined by a URL with parameters and a JSON body.

These operations can be invoked with a REST client like Firefox's RESTClient. It is necessary to specify some headers like Content-Type, Accept or Authentication (with HPSA credentials as well).

Note that every operation is asynchronous and returns a jobID for Manager to request Job Status (excluding get Job Status).

11.3.1.1 VNF Status OK notification

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

```
Method:POST
```

```
URL:$HOST:$VNFM_PA_PORT/v1/vnfm/$MANAGER_INSTANCE_ID/vnf/$VNF_I
NSTANCE_ID/vnfstatus
Authentication--> Basic Authentication --> Username(hpsa user)
Password(hpsa password)
Headers--> Content type --> Name:Content-Type
Value:application/json;charset=UTF-8
Headers--> Accept --> Name:Accept
Value:application/json;charset=UTF-8

Example JSON:
{
    "id": "$VNF_INSTANCE_ID",
    "version": "",
    "noOfVDUElements": "1",
    "status": "INSTANTIATED",
    "VDUStatusDescriptor": [
        {
            "unitId": "14146996719461",
            "status": "INSTANTIATED",
            "instances": [
                {
                    "id": "14146996793021", "status":
"INSTANTIATED"
                }
            ]
        }
    ] }
```

11.3.1.2 VNF Status Fail notification

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

```
Method:POST
URL:https:// $HOST:$VNFM_PA_PORT/v1/vnfm/9cc438f2-2568-4418-
88c8-ec7d763f34bd/vnf/14146996652991/fault
Authentication--> Basic Authentication --> Username(hpsa user)
Password(hpsa password)
Headers--> Content type --> Name:Content-Type
Value:application/json;charset=UTF-8
Headers--> Accept --> Name:Accept
Value:application/json;charset=UTF-8

JSON:
{
    "vnfId": "14146996652991",
    "faultCode": "10001",
    "faultDescription": "....",
    "VDUFaultDescriptor": [
        {
            "unitId": "14146996719461",
            "faultCode": "10001",
            "faultDescription": "Failed due to ..."
        }
    ]
}
```

11.3.1.3 Grant Scale

Scales up a VNF in the inventory, and then sends back the JobID for VNF Manager to request the status.

NFVD is compliant to each type of scale: out, in, up, down. The operation, however, is always the same.

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

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

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

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

```
JSON 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"
                            }
                     }
              }
       }
}
}
```

11.3.1.4 Get Grant Details

This is a synchronous operation that converts a concrete VNF tree to a JSON structure understandable by a Manager.

```
Method:GET
URL:https://$HOST:$VNFM_PA_PORT/v1/vnfm/$MANAGER_INSTANCE_ID/vn
f/$VNF_INSTANCE_ID/grant
Authentication--> Basic Authentication --> Username(hpsa user)
Password(hpsa password)
Headers--> Content type --> Name:Content-Type
Value:application/json;charset=UTF-8
Headers--> Accept --> Name:Accept
Value:application/json;charset=UTF-8
```

```
Method:GET
URL:https://$HOST:$VNFM_PA_PORT/v1/vnfm/$MANAGER_INSTANCE_ID/vn
f/$VNF_INSTANCE_ID/grant
Authentication--> Basic Authentication --> Username(hpsa user)
Password(hpsa password)
Headers--> Content type --> Name:Content-Type
Value:application/json;charset=UTF-8
Headers--> Accept --> Name:Accept
Value:application/json;charset=UTF-8
```

11.3.1.5 Get Job Status

This is a synchronous operation. This returns a specific job (scale, create, change status) status.

```
Method:GET
URL:https://$HOST:$VNFM_PA_PORT/v1/vnfm/$MANAGER_INSTANCE_ID/jo
bs/$JOBID
Authentication--> Basic Authentication --> Username(hpsa user)
Password(hpsa password)
Headers--> Content type --> Name:Content-Type
Value:application/json;charset=UTF-8
Headers--> Accept --> Name:Accept
Value:application/json;charset=UTF-8
```

11.3.2 VNF Manager Plugin: from NFVD to VNFM

NFVD communicates with VNFM through a Plugin just like it does with OpenStack. This plugin contains workflows and Activation Command Templates for each operation. These templates can be observed in HPSA Extension Pack future-gui.

They are defined by Manufacturer: GENERIC, Element Type: VNFM and a specific NAME. The templates content is, at the end, which Manager receives.

11.4 VNF Manager Operations

GPM has been chosen for running manager operations. There are 2 GPM process (first of them has 2 parts) to perform these actions. Finding the GPM processes and instantiating them will run specific manager operations. Actions-> Instantiate

Template information:

» Name:	SCALE_OUT_VNF
» Description:	Scale Out Operation for VNF
» Common Config:	
» Manufacturer:	GENERIC
» Element Type:	VNFM
» OS Version:	NFVO_TO_VNFM_v1

» Actions

```
[TEMPLATE:Config]
http.operation=PUT
 http.url.sufix=/v1/vnf/${vnfid}/scale_out

[TEMPLATE:Do]

[TEMPLATE:Section 0]
#if($JSON_REQUEST=="")
 {
 "vnfdescriptorid":"$VNF_DESCRIPTOR_ID",
 "flavor":"$FLAVOR",
 "vnfdescriptorversion":"$VNF_DESCRIPTOR_VERSION",
 #if($VNF_DETAILS_DESCRIPTOR!="")
 "vnfDetailsDescriptor":"$VNF_DETAILS_DESCRIPTOR",
 #if($GRANT_APPROVAL_DESCRIPTOR!="")
 ",
 #end
 #end
 #if($GRANT_APPROVAL_DESCRIPTOR!="")
 "grantApprovalDescriptor":"$GRANT_APPROVAL_DESCRIPTOR"
 #end
 }
 #else
 $JSON_REQUEST
 #end
```

Figure 137 Template for SCALE_OUT_VNF

From each instance one (or many) task is generated. Interacting with these tasks allows us to set the values required by each process. AD-> Tasks -> Search

» File | » AD | » Search | » SNMP Tool | » Configuration Management | » Administrator | » Inventory | » Help

Search Process Definition Section:

» Search

» Actions

Name	Type	Global status	Status
Instantiate a VNF			UNLOCKED

Found one record. Page **1**
Export: CSV | Excel | XML

Figure 138 Solution Container Search

11.4.1 Deploy and Register a VNF Manager

This process instantiates and register a VNF Manager. This is not a completely automatic process, so it is needed to do some manual actions after that (explained later).

The form to do this is VNFSelectionForm.

Actions -> Interact to view the form.

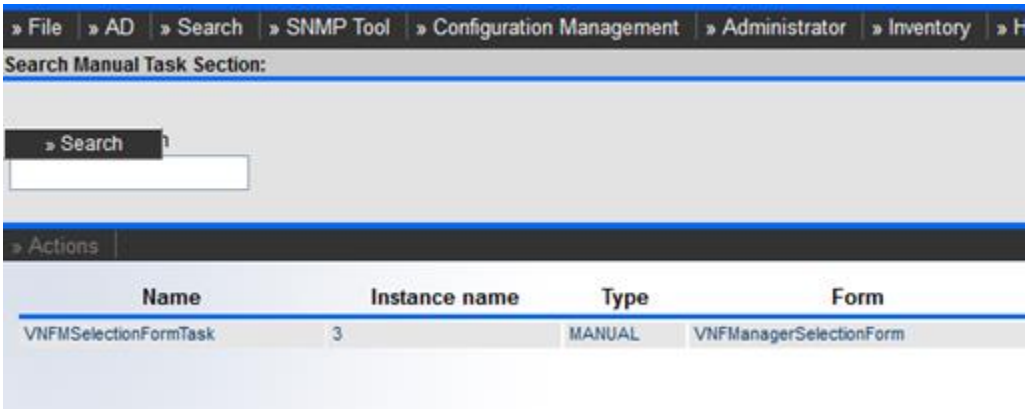


Figure 139 Solution Container: Deploy and register VNF Manager

In the GPM Task form all fields must be filled excluding the VNFManager ID.

After that an automatic process is launched which will register out orchestrator in the manager DB, get the descriptor list from manager and, for each different descriptor, create a VNF template. These templates will serve as a reference to "Instantiate a VNF through a Manager" process.

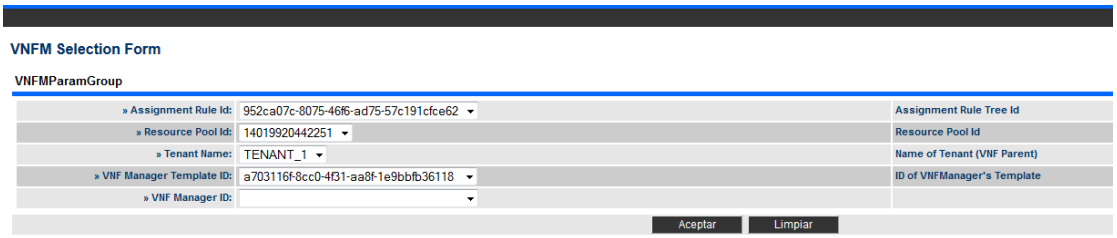


Figure 140 GPM Task

The manual tasks needed to be performed after the GPM ends are:

1. Ensure Tenant Instance Artifacts are related with QUOTAS.
2. Ensure there is a valid Assignment Relationship tree for VNF Manager and also VNF.
3. Ensure Templates for VNF have been crated.
4. Assign manually tenants to VNF Manager
5. Assign resources to VNF Manager
6. Complete the VNF templates taking the reference ID from templates created before.

Ending this instantiation process GPM will redirect automatically to the Instantiation of a VNF and a VFN Descriptor Selection Form will appear.

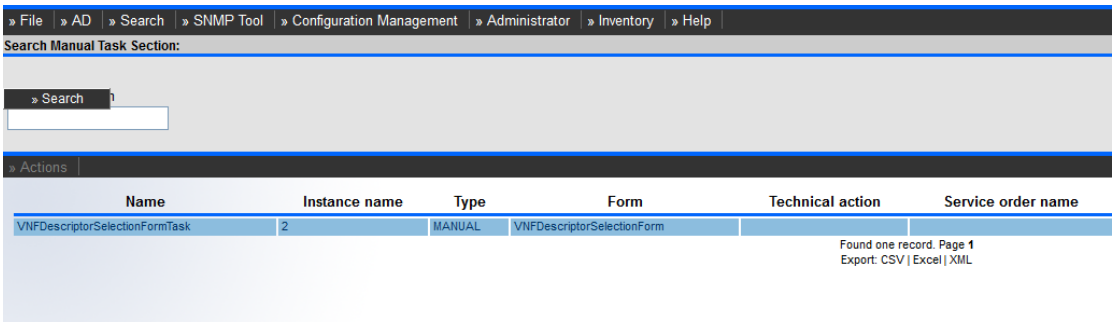


Figure 141 Manual tasks

11.4.2 Instantiate a VNF through a VNF Manager

This process starts at the same point that previous section 2.3.1 but it is necessary to select a VNF Manager ID in the VNF Manager Selection Form.

Figure 142 Instantiate VNF

Then, the task will change to `VNFDescriptorSelectionForm`. Three selection fields must be filled: Tenant, Resource Pool Id and Template for VNF.

Then GPM sends the Create Order to the Manager which theoretically would send Director back a "Grant Create VNF" Request. After that NFV Director starts communicating with Manager asking for the create Job Status. If response contains a "Finished" or "Error" status, NFVD stops querying and goes forward or stops operation, respectively. Last step is querying for Descriptor Details and drop it into a temp file just for compare. Last form will contain the path to that file.

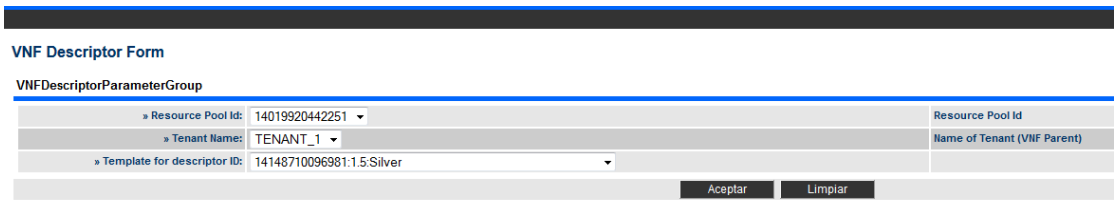
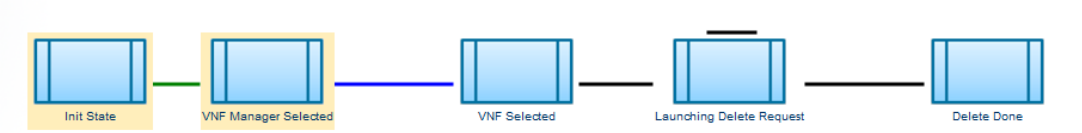


Figure 143 VNF DescriptorParameterGroup



11.4.3 Delete a VNF through a VNF Manager

This process will request the user for a VNF Manager Instance Id and a VNF Instance Id which will be deleted. After communicating manager the delete order and checking the job status is finished, NFV Director will delete VNF instance and children from Inventory.

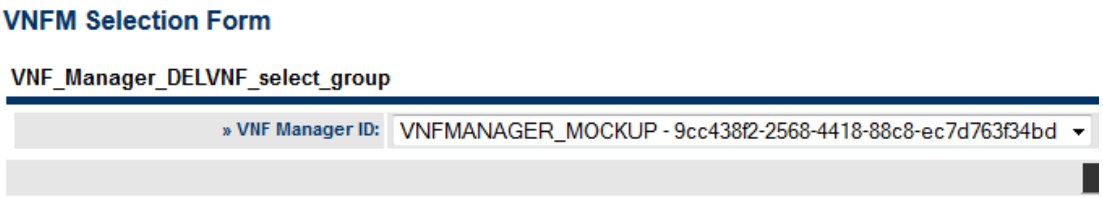


Figure 144 VNF Selection

Figure 145 Delete VNF

Visualizing a process status is possible going AD->Process Instance -> Search -> Find required process instance and click on Actions -> Details. As an example:

Chapter 12

State Propagation

If a Virtual machine's status is affected due to the single CPU failure, then its operation status may get affected, similarly the operational status of its parent component(s) may also get affected.

Therefore, this automated change of operational status for a calculated hierarchy of component here is considered as state propagation.

In a chain like NS->VNF->VNFC->VM, if there is no propagation rule at any intermediate level, then the state does not get propagated further up in the chain.

Propagation of status happens based on propagation modes: LATEST, HIGHEST or NONE.

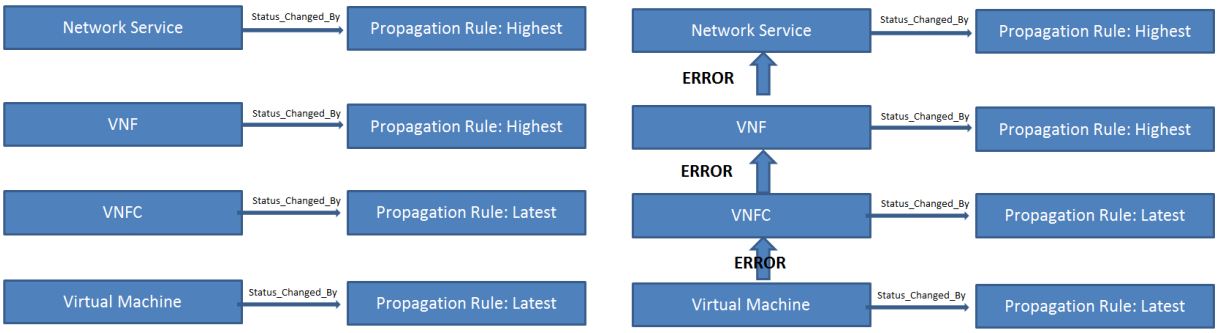


Figure 146 Propagation modes

12.1 State propagation functionality

State propagation is a concept of changing and propagating the operational status of a component and its affected parent components which is triggered by an alarm on source component.

For example:

If a Virtual machine's status is affected due to the single CPU failure, then its operation status may get affected, similarly the operational status of its parent component(s) may also get affected.

Therefore, this automated change of operational status for a calculated hierarchy of component here is considered as state propagation.

12.2 Flow

Status change propagation can be triggered by SiteScope or a VNFM alarm.

Typically after the completion of state propagation user can see the status change in fulfillment UI and the same is notified back to assurance gateway via update notification calls.

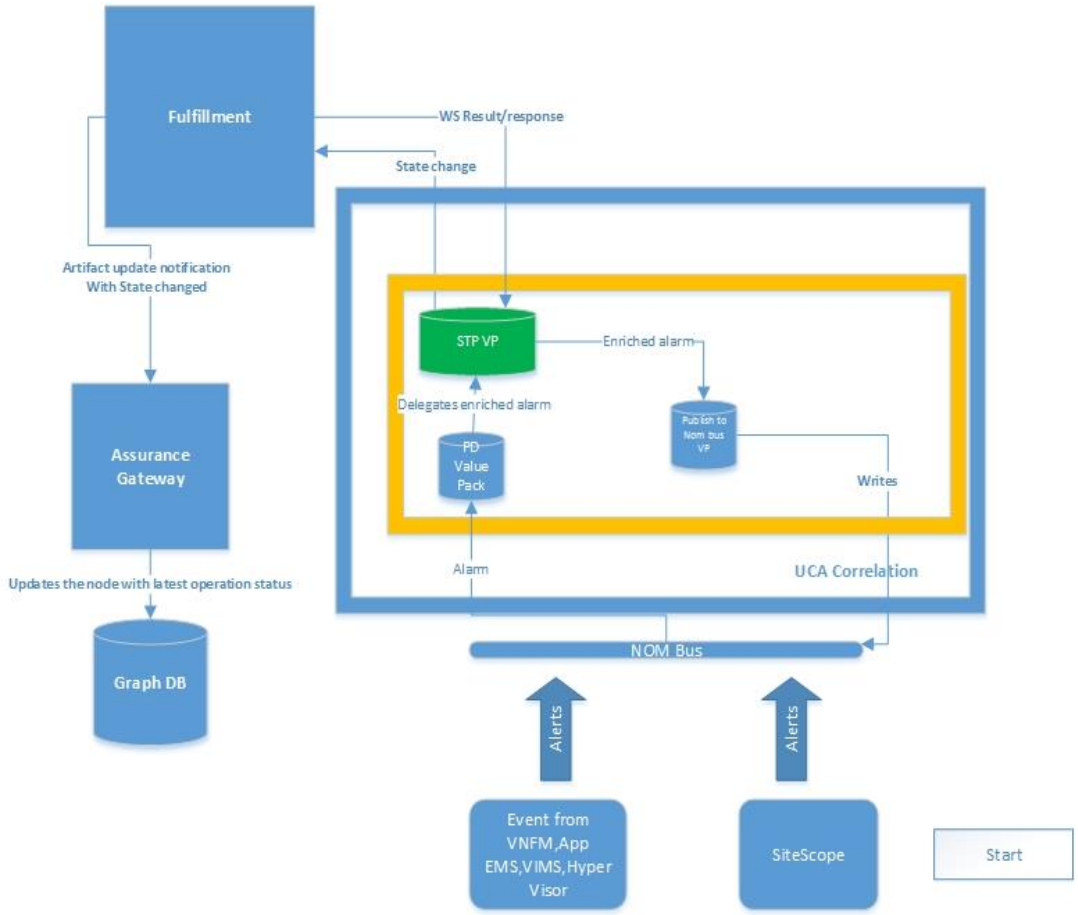


Figure 147 State propagation flow

12.3 State propagation process

In NFV Director there are two sources of alarm, but new source of alarm can be dynamically handled if they send the alarm in given X.733 format with few NFVD related data.

12.3.1 Site Scope alarms

Site scope is a component of NFVD and it generates alarms when any KPI breach occurs.

It is configured to send `alertseverity` in `AdditionalText` field of X.733 of an alarm.

```
<additionalText>_customPropertiesValues=|_httpPort=8888|_webser
verAddress=15.154.112.75|alertHelpURL=http://127.0.0.1:18088/Si
teScope/sisdocs/doc_lib/index.htm?single=false&amp;context=syst
em_avail&amp;topic=config_sis_alert|diagnosticTraceRoute=|error
Only=|goodOnly=
VirtualMachine/37add4bb-80f1-49d6-93eb-
b1203d5eafba/Realtime/cpu/usage.average[]: 0|FullGroupId=
|SiteScopeURL=http://127.0.0.1:18088/SiteScope|SiteScopeuserurl
=http://127.0.0.1:18088/SiteScope/userhtml/SiteScope.html|state
=VirtualMachine/37add4bb-80f1-49d6-93eb-
b1203d5eafba/Realtime/cpu/usage.average[]=0%|tag=|targetHost=cm
s-
vcentre.ind.hp.com|targetIP=15.213.49.32|targetIPVersion=IPV4|t
emplateDeployPath=Script
Path/nfvddemo/TEST_2_Server/TEST_2_Frontend/TEST_2_VM_Linux_xsm
all/artifactId-14170989360221/VMWARE VM CPU Monitor|time=4:39
```

```
AM
11/28/14|warningOnly=|customerId=&amp;lt;customerId&amp;gt;|ale
rtSeverity=good</additionalText>
```

The field **`alertSeverity`** contains the severity of an alarm. The values of `alertSeverity`, sent by **`SiteScope`** can be `Warning`, `Error`, `Good` and so on.

Note

Alert severity with `good` value is clearance of an alarm and signifies normal condition.

12.3.1.1 Operational status mapping

User needs to map the `alertSeverity` of an alarm to a meaningful operational status of their choice, in the following property file.

```
$UCA_EBC_DATA/instances/default/deploy/UCA_NFVD_StatePropagation3.0/conf/
alarmmapping.property
```

12.3.1.2 Sample operational state mapping (Out of the box)

```
#Maintain the order or severity for alarms, lowest first.
INTERMDEDIATE=
good=normal operation
warning=degraded operation:Warning
MINOR=degraded operation:Minor
MAJOR=degraded operation:Major
CRITICAL=degraded operation:Critical
error=degraded operation
CLEAR=normal operation
WARNING=degraded operation:WARNING
```

12.3.1.3 Alarm severity level

As mentioned in the property file that the order of severity will be taken from the order in which they are mentioned in the property file that is lowest severity first.

This will be used when there is a need to compare the severity of operational status from its previous status.

12.3.1.4 Operational status calculation

The concept of state propagation is quite simple, that for a given component's alertSeverity, the corresponding mapped `Operational status` will be read from the property file and the same will be set and propagated up the hierarchy.

12.3.1.5 State propagation model changes

The changes in artifact definitions, instance and relationships are as follows.

1. Artifact definitions

All the components have a category called STATUS as displayed as follows:

- Key attributes

The `STATUS` category has three key attributes related to `State propagation`.

- `Operational_Status` indicates the current operational status of that component. There is no default value for this field. Its values will be derived from the `alarmmapping.property` file.
- `Operational_Status_Date` indicates the date and time in UTC format when the operational status was changed, basically driven by alarm raised time.
- `Source` denotes if the state propagation was triggered by NFVD (that is, Internal) or by some other external entity like VNFM.

2. PROPAGATION_RULE artifact

   This new artifact is introduced to further decide how and if operational status of components needs to be changed.

- Key Attributes
  - `Name`: Any user text to distinguish the rule.
  - `Propagation_mode` decides how the `operational_status` for a given component must be calculated.

    This can have the following values:

    - LATEST: This means as soon as the alarm is received by STP value pack, it will fetch the corresponding operational status from `alarmmapping.property` file for a given `alertSeverity` and will set and propagate the same status, without any comparison or calculation.
    - HIGHEST: In this mode, there will be a comparison of previous operations status with current operation status for a given component and the highest (highest severity) will be set and propagated.
    - RULE_BASED: In this mode, the operational status will be calculated based on user defined rules. The alarm will be delegated to the rule based value pack.
    - NONE: In this mode, state propagation will not take place.
    - Propagation_rule: An optional attribute, if the propagation mode is RULE_BASED, then the user needs to specify the rule value pack name.

3. Relationship definitions

   When you wish to enable the state propagation for any component, starting from any level in the hierarchy, you have to define a relationship between that component and PROPAGATION_RULE artifact with relationship type as `STATUS_CHANGED_BY`.

4. Configurations

   State propagation value pack requires the details of assurance database and fulfillment web service endpoint. You can configure the same at the following location and redeploy the value pack.

12.3.2 NFVD database and Fulfillment configurations

```
/var/opt/UCA-EBC/instances/default/deploy/UCA_NFVD_StatePropagation-3.0/conf/statepropagation.property
```

```
#The URL for fulfilment for state propagation
FULFILLMENT_URL=http://localhost:8071/ngws/service?wsdl
#The URL for NFVD database
NFVD_DB_URL=http://localhost:17373/db/data
#Set if alarm after STP needs to be published to NOM Bus. value true/false
PUBLISH_TO_NOM=true
FULFILLMENT_REST_URL=http://localhost:8080
USE_SOSA_API=false
```

12.3.2.1 FULFILLMENT_URL

This is the URL of fulfillment where the status of all the affected components will be changed and propagated.

12.3.2.2 NFVD_DB_URL

This is URL of NFVD DB, from where all related parents will be fetched.

12.3.2.3 PUBLISH_TO_NOM

Once the states are propagated, the enriched alarm will be published to OM bus, if this flag is set to `true`. Else, it will not be published.

12.3.3 Parent hierarchy configuration

The following xml configurations define the available parent child relationship as per current artifact definition. If any new type of parent or child is introduced, then it can be configured here to propagate the states to the new type of parents or child.

```
$UCA_EBC_DATA/instances/default/deploy/UCA_NFVD_StatePropagatio
n-3.0/conf/CypherQueryData.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This xml describes what are the possible parent family type for any
        child family type -->
<CypherQueryData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="CypherQueryData.xsd">
        <ChildParentRel>
                <childNode>
                        <childFamilyType>VIRTUAL_MACHINE</childFamilyType>
                        <parentNode>
                                <parentFamilyType>VNF_COMPONENT</parentFamilyType>
                        </parentNode>
                </childNode>

                <childNode>
                        <childFamilyType>VNF_COMPONENT</childFamilyType>
                        <parentNode>
                                <parentFamilyType>VNF_COMPONENT</parentFamilyType>
                        </parentNode>
                        <parentNode>
                                <parentFamilyType>VNF</parentFamilyType>
                        </parentNode>
                </childNode>

                <childNode>
                        <childFamilyType>VNF</childFamilyType>
                        <parentNode>
                                <parentFamilyType>VNF</parentFamilyType>
                        </parentNode>
```

```
                        <parentNode>

<parentFamilyType>NETWORK_SERVICE</parentFamilyType>
                        </parentNode>
                </childNode>

                <childNode>
                        <childFamilyType>NETWORK_SERVICE</childFamilyType>
                        <parentNode>

<parentFamilyType>NETWORK_SERVICE</parentFamilyType>
                        </parentNode>
                </childNode>
        </ChildParentRel>

        <whereArtifactIdClause> where has(n.artifactId) and
n.artifactId=</whereArtifactIdClause>
        <whereArtifactIdInClause> where has(n.artifactId) and n.artifactId IN
</whereArtifactIdInClause>
        <whereFamilyClause>where has(m.artifactFamily) and
m.artifactFamily=</whereFamilyClause>
        <whereNameClasue>where has(n.`GENERAL.Name`) and
n.`GENERAL.Name`=</whereNameClasue>
        <startCypher>START n = node(</startCypher>
        <startCypherAllNode>START n = node(*) </startCypherAllNode>
        <matchCypher>) match </matchCypher>
        <nTomIncomingRel> (n) &lt;-- (m) </nTomIncomingRel>
        <statusChangeRel> (n)-[:STATUS_CHANGED_BY]->(m) </statusChangeRel>
        <returnM> RETURN m</returnM>
        <returnN> RETURN n</returnN>
        <singleQuote>'</singleQuote>
</CypherQueryData>
```

For example:

If you have a use case, where the status needs to be propagated to parent of Network service, you just need to add one more `parentNode` tag with a given parent family type in `NETWORK_SERVICE childNode` tag.

12.3.4 VNFM alarms

The source of alarm or state propagation can be VNFM. The alarms coming from VMFM has different attributes, and those alarms already contain old and new states.

Therefore, the mapping from severity of alarm to actual operation state will not be required, but the severity level of operational status has to be maintained in the following property file.

VNFM alarm will be received by assurance gateway as update artifact notification and then will be sent to OM bus.

```
/var/opt/UCA-EBC/instances/default/deploy/UCA_NFVD_StatePropagation-
3.0/conf/operationastatuslist.property
```

```
#Maintain the order or severity for operational status, in ascending order with
lowest first
power-on
power-down
degraded_operation
```

```
normal_operation
```

12.3.5 Key attributes in VNFM alarms

12.3.5.1 SourceArtifactId

This will contain the artifact ID of the alarm originating node.

12.3.5.2 alarmName

This will have value as `OperationalStatusChange` indicating that the alarm is operational state change alarm.

12.3.5.3 newState

This will contain the actual operational state of the component that needs to be propagated.

12.3.5.4 oldState

This will contain old operational state of the component.

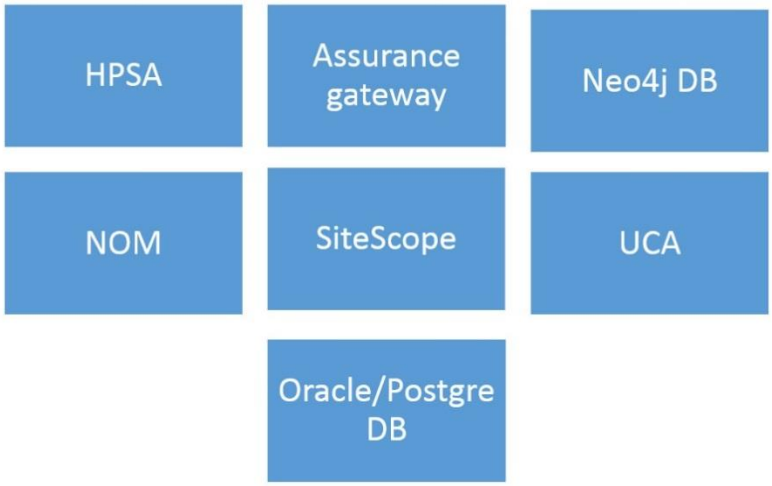
Chapter 13

Self Monitoring

Self-monitoring is a concept of monitoring and managing components of NFVD.

13.1 Components managed

The following list of components can be monitored as part of self-monitoring process.



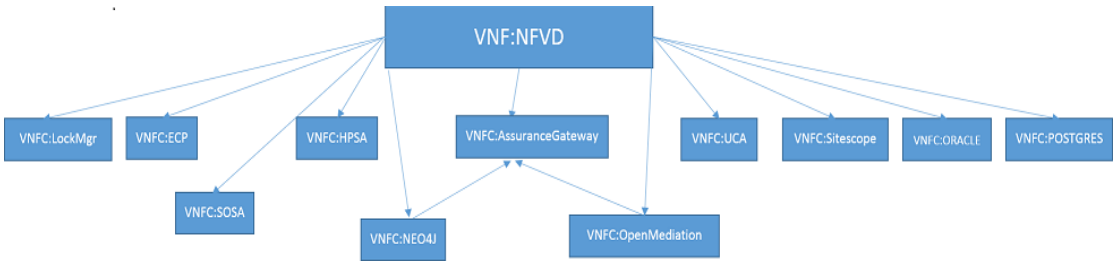


Figure 148 Self monitoring components

13.2 Monitoring process

All the component are essentially monitored by agentless monitoring component of NFVD i.e. HP SiteScope.

Note

To ensure the correct functioning of the NFVD system as a whole the monitoring engine i.e. SiteScope system itself must be up and running with good performance.

13.2.1 Site scope templates

Since monitoring is done by HP Site scope, various inbuilt site scope templates for monitoring each component are available.

As part of self-monitoring templates, below is the list of templates that will primarily perform process and log monitoring.

- Assurance Gateway template
- SiteScope template
- HPSA template
- OpenMediation template
- UCA template
- Neo4j template
- Oracle DB template
- Postgres DB template
- SOSA template
- ECP template
- LockManager template

Each of the templates will have thresholds configured based on the application/server that is being monitored. Each template will have associated alert actions configured that will get triggered on breach of the kpi's defined as part of thresholds.

13.2.2 Modelling for self-monitoring

The model has been designed in alignment with NFVD as a VNF.

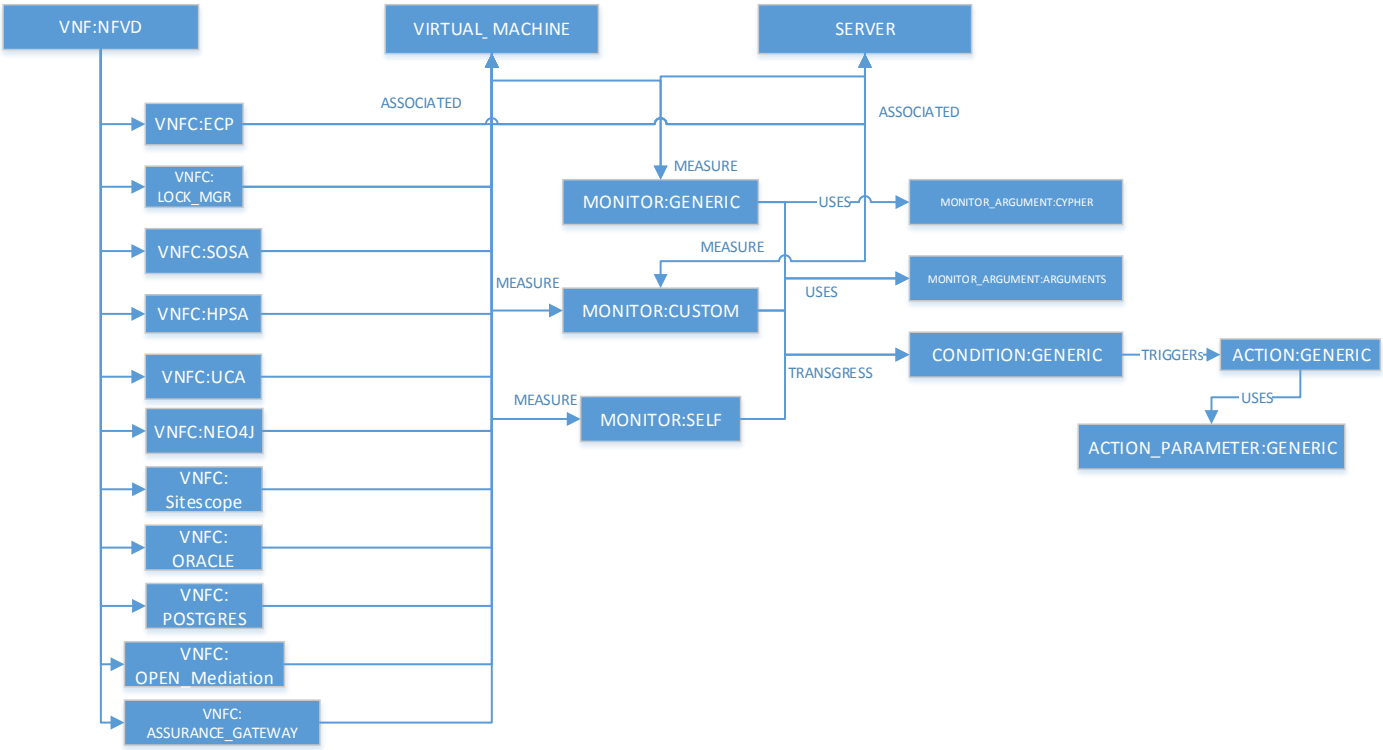


Figure 149 Self monitoring Model

13.2.3.1 Artifact definitions

Below are the definitions for self-monitoring related artifacts

5. VNF: NFVD
6. VNFC :LockManager
7. VNFC :ECP
8. VNFC :SOSA
9. VNFC :HPSA
10. VNFC :NEO4J
11. VNFC :AssuranceGateway
12. VNFC :OpenMediation
13. VNFC :UCA
14. VNFC :SiteScope
15. VNFC :ORACLE
16. VNFC :Postgres

There is an attribute ENABLED under the category MONITOR. By default the value is set to true. If the value is set to true Process and log monitors for the VNFC and also all monitors under it will be deployed. A daemon thread would be running in Assurance gateway to

deploy the monitors under VNF:NFVD. If user needs to disable the monitors for any of the VNFC, he has to update the VNFC by setting the attribute value as false. All monitors will get undeployed.User can disable Log Monitors for a VNFC by setting the attributes under the category LOG_MONITOR to empty.

User needs to fill up all the details under the CONNECTION category.

Deployment path can be specified in the attribute PATH under DEPLOYMENT category.

If user wants to trigger an action for the process and log monitors at VNFC, MONITOR:SELF artifact needs to be created and associated to the VNFC with relationship “MEASURE”. Then CONDITION:GENERIC and ACTION:GENERIC can be created. CONDITION:GENERIC is associated to MONITOR:SELF with relationship type “TRANSGRESS” and ACTION:GENERIC is associated to CONDITON:GENERIC with relationship type “TRIGGERS”.

Counters supported for self monitors are status. If status is equal to 0 means process is running fine, if status is equal to -1 means process is not running fine.

For log monitors supported counter is “matches”, indicates the number of occurrences of the log pattern in the log file. Currently for error cases expression is “matches > 10” and good cases expression is “matches < 10”. Value 10 is hardcoded it cannot be configured. User can associate condition with above expression to Monitor:SELF and an action can be associated to condition. So that action can be triggered by UCA when threshold is breached.

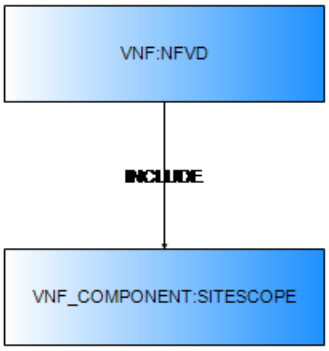


Figure 150 VNF : NFVD artifact

MONITOR: SELF

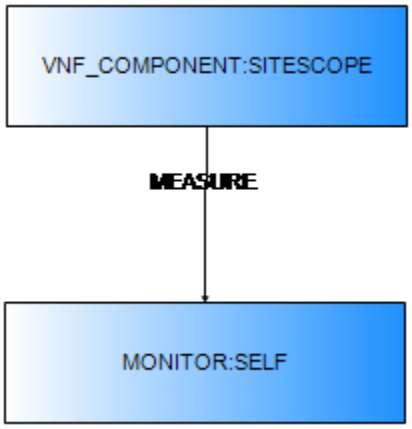


Figure 151 MONITOR : SELF artifact

MONITOR: CUSTOM

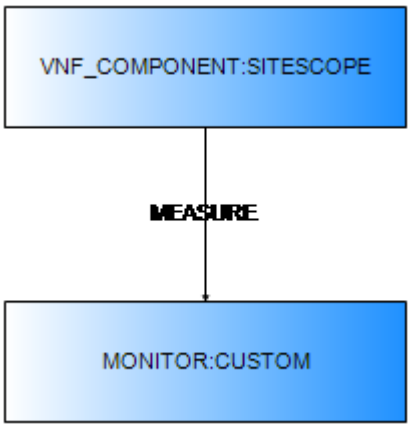


Figure 152 MONITOR : CUSTOM artifact

MONITOR: GENERIC

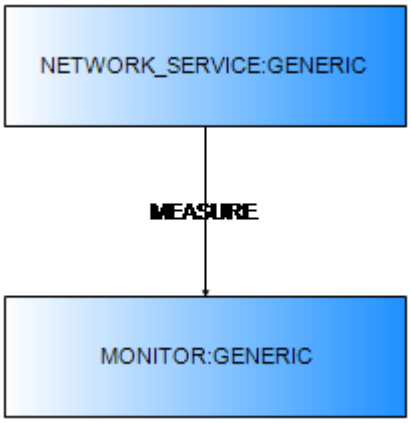


Figure 153 MONITOR : GENERIC artifact

MONITOR_ARGUMENTS

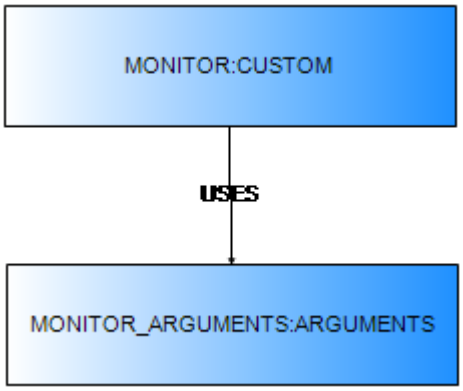


Figure 154 MONITOR_ARGUMENTS: GENERIC artifact

This artifact instance further provides details for monitoring the component.

13.2.4 Triggering self-monitoring

On upload of nfvd instances in fulfilment as part of installation, it will send notifications to Assurance Gateway. Assurance Gateway will have these instances inserted in topology. Assurance Gateway will trigger a service thread on start of Jboss which runs periodically at specific interval (as configured in nfvd.properties file).

It fetches all the monitors associated with VNF of Type NFVD from Topology DB. If attribute "ENABLED" is set to true under the category "MONITOR" in the VNFC artifact, associated monitor will be deployed. It will neglect the monitors which have already been deployed. Once the Monitor has been deployed successfully it will update the status of that monitor to DEPLOYED.

13.2.5 NFVD configuration for Self-monitoring

The self-monitoring related configuration file is available at location: /var/opt/HP/nfvd/conf/nfvd.properties

13.2.5.1 nfvd.property

Here user can configure the interval of monitoring thread in assurance gateway.

```
# Configure RESYNC_AT_STARTUP as true/yes, for synchronization during Assurance
startup
RESYNC_AT_STARTUP=false
# Fulfillment URL connection timeout limit in millisecond, default 1.5 min
FULFILLMENT_CONNECTION_TIMEOUT=90000
# Fulfillment URL response for query timeout limit in millisecond, default 1.5 min
FULFILLMENT_RESPONSE_TIMEOUT=90000

FULFILLMENT_REST_URL=http://localhost:8080

FULFILLMENT_SOAP_URL=http://localhost:8071/ngws/service?wsdl


#cache related requests
#cacheEnabled = (true)/(false), to enable/disable assurance graph database cache
cacheEnabled = false
#size of the cache, maximum number of objects in the cache at a time.
maxCacheSize = 10000

#Self Monitors Run Frequency in MilliSeconds
SELF_MONITORS_RUN_FREQUENCY=300000
```

13.2.6 Stopping self-monitoring

On deletion of any self-monitoring instances in Fulfilment, it will send notifications to Assurance Gateway. For these instances Assurance Gateway will first trigger un-deployment of monitors and then also trigger deletion of this instance from topology.

This service thread runs continuously still the Assurance Gateway is alive and will deploy if some new monitors get added or will undeploy if some monitors get deleted.

Chapter 14

Extending Monitors using Site Scope

This section describes how to use the NFV Director to monitor resources.

NFV is a complex system that needs constant monitoring of the physical and the logical entities. The provisioning and monitoring functions can be brought together through rules that define manual or autonomous scaling and placement actions, based on measured key performance indicators (KPIs).

To resolve this, HP NFV Director includes the agent-less monitoring component. Built using HP SiteScope, it can monitor a wide variety of monitoring points, issuing events, or executing commands when predefined thresholds are crossed. Since different virtual network functions have different monitoring needs, the monitoring points and thresholds are automatically configured by HP NFV Director as the VNF is provisioned or modified. As an agent-less solution, HP NFV Director does not require the installation of monitoring agents on the target systems.

The following sections use SiteScope v11.23 for illustrating the product capabilities. Refer to its documentation for more details.

14.1 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.

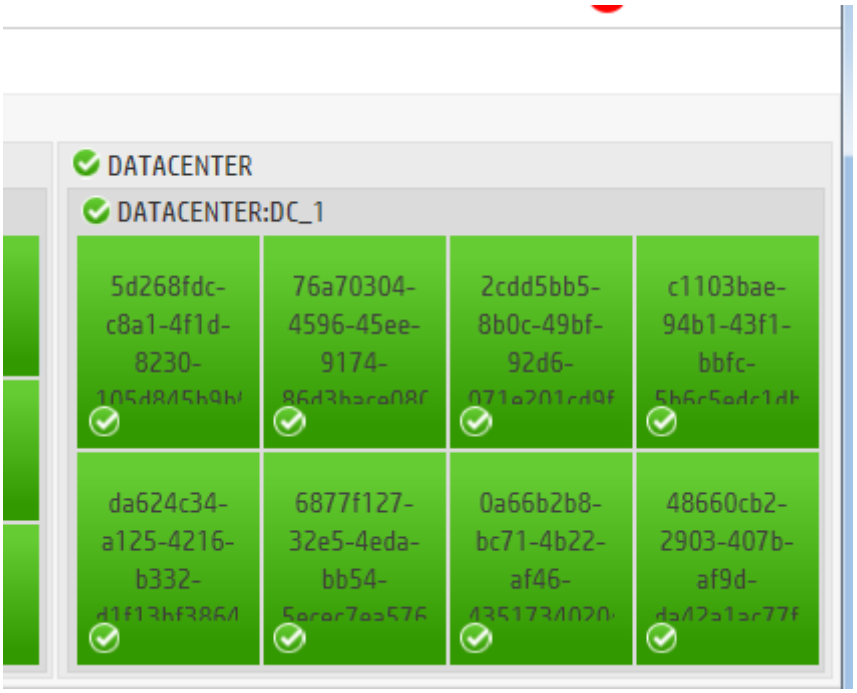
a. Multiple VNFC:SITESCOPE can be created but only one of them can have IS_DEFAULT attribute set to true. The flag marks the VNFC:SITESCOPE as default sitescope which can be used for monitor deployment in case when no VNFC:SITESCOPE is attached to resource tree ex:DATACENTER etc. If the user has set 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.

14.2 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.



14.3 Accessing SiteScope

Use the following steps to access SiteScope.

1. To access SiteScope from a browser, enter the SiteScope address in a Web browser.

   The default address is: `http://<server name>:<port>/SiteScope`.

2. (Optional) To access SiteScope from the Start menu (Windows platforms only), select **Start** > **Programs** > **HP SiteScope** > **Open HP SiteScope**.

3. Enter the login credentials and click the **Login** button.

14.4 Overview of SiteScope dashboard

When you connect to SiteScope, default dashboard can be seen on successful login. For the first time SiteScope is deployed, there is a delay for initialization of the interface elements. Dashboard displays current performance data for the infrastructure elements being monitored by SiteScope and provides access to functions you use to define filters. Dashboard displays a table of groups and monitors for the elements highlighted in the monitor tree or listed in the path. You can double-click each group or monitor node to navigate to child nodes and monitors.

SiteScope dashboard window—contains the following key elements:
Common toolbar—Provides access to page options, documentation, and additional resources. This toolbar is located on the upper part of the window.
Context toolbar—Contains buttons for frequently-used commands in the selected SiteScope context.

Context tree—enables you to create and manage SiteScope objects in a tree structure.

Context buttons—Provide access to the SiteScope Monitors, Remote Servers, Templates, Preferences, Server Statistics, and Diagnostic Tools.

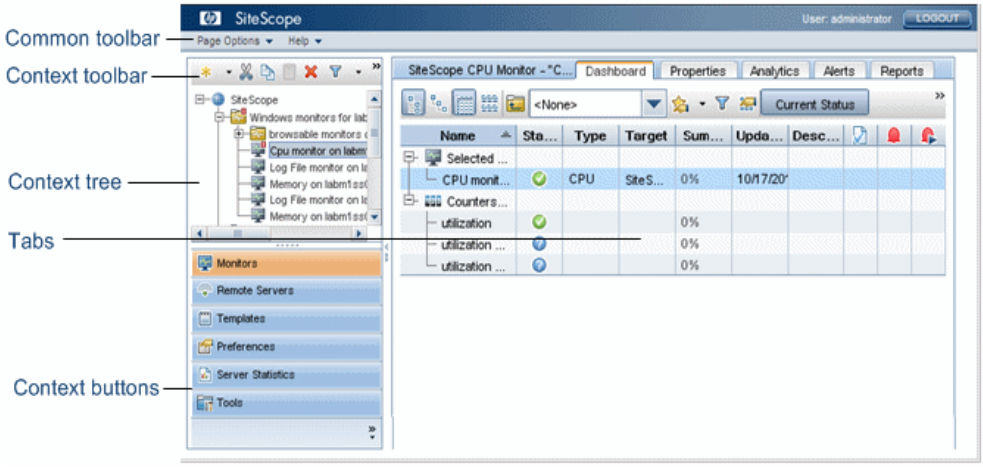


Figure 155 SiteScope dashboard

SiteScope monitoring provides a real-time picture of system availability and performance. You configure SiteScope monitors to collect metrics from a range of infrastructure components, including Web, application, database, and firewall servers. The status and metrics are then aggregated for presentation in SiteScope Dashboard.

Dashboard is linked to the SiteScope monitor tree hierarchy. The data displayed in Dashboard represents the selected context in the monitor tree. The highest level is the SiteScope node and any applicable monitor groups. The lowest-level element for display in a Dashboard view is an individual SiteScope monitor and its measurements.

Dashboard includes functions that you can use to customize the display of monitor information. This includes defining named filter settings to limit the display of data to those matching defined criteria. You can also select various data display options.

Dashboard also includes hyperlinks and menus that you can use to navigate through the hierarchy of monitor elements, manually run a monitor, disable monitors, and access alert definitions.

SiteScope Dashboard has the following context buttons that are available from the left pane:

UI Element	Description
Monitors	Enables you to create and manage SiteScope groups and monitors in a hierarchy represented by a monitor tree.
Remote Servers	Enables you to set up the connection properties so that SiteScope can monitor systems and services running in remote Windows and UNIX environments.
Templates	Enables you to use templates to deploy a standardized pattern of monitoring to multiple elements in your infrastructure. You can use preconfigured SiteScope solution template or create and manage your own templates.
Preferences	Enables you to configure specific properties and settings related to most of the administrative tasks within SiteScope.
Server Statistics	Enables you to view key SiteScope server performance metrics.
Tools	Displays diagnostic tools that can help you troubleshoot problems in SiteScope and facilitate monitor configuration.

Figure 156 SiteScope Dashboard context buttons

14.5 Analyzing data in SiteScope dashboard

This task describes the steps to analyze data in SiteScope Dashboard.

1. View monitor and measurement status and availability.

   When viewing SiteScope data in the Current Status view of Dashboard, you can explore the monitor tree to view monitor and measurement status and availability.

Example: Measurement status and availability for a monitor:

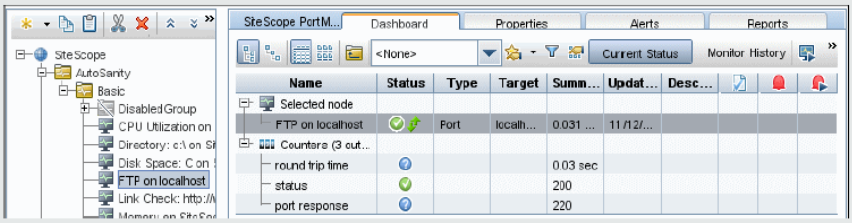


Figure 157 SiteScope Dashboard: view monitor status

2. View configured and triggered alerts.

   You can view data about alerts in the configured alerts and triggered alerts columns. If alerts are configured for a monitor, you can double-click the Configured Alert icon to see the list of configured alerts, and select an alert to view or edit the alert properties.

Example: Configured alerts

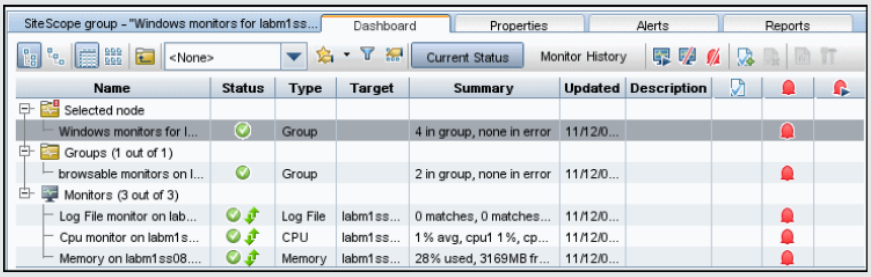


Figure 158 SiteScope Dashboard: view configured and triggered alerts

3. Disable monitors or monitors in group.

   Depending on the diagnosis, you can disable the monitor or monitors in group, or disable alerts associated with the monitor or group and continue to use the monitor.

4. Acknowledge monitors.

   To acknowledge monitor status, select a monitor or group and click the Add **Acknowledgment** icon or select **Add Acknowledgment** from the context menu, and enter the details in the **Acknowledge Monitors In Group** dialog box.

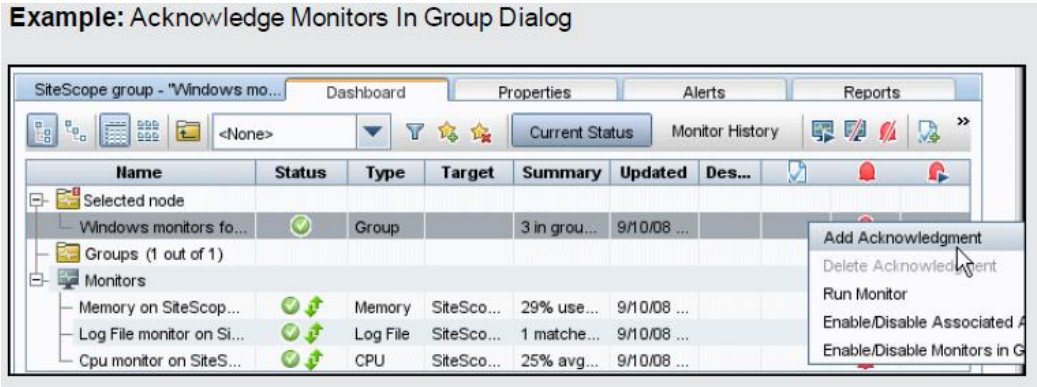


Figure 159 SiteScope Dashboard: Acknowledge monitors

5. Monitor your Microsoft Windows/UNIX server's resources.

   You can create a Microsoft Windows or UNIX Resources monitor to monitor your Windows or UNIX Server.

6. View monitor history.

   You enable and configure monitor history in the General Preferences. To view monitor history, click the Monitor History button in SiteScope Dashboard.

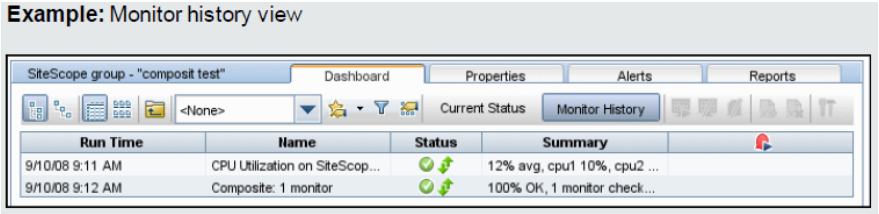


Figure 160 SiteScope Dashboard: view monitor history

14.6 Dashboard - status and availability levels

The following table provides details on different status and availability levels:

Icon	Description
	Good Status. All performance measurements are within the Good threshold level.
	Warning Status. At least one performance measurement is within the Warning range, but no measurements are within the Error or Poor range.
	Error/Poor Status. At least one performance measurement is within the Error or Poor range. This indicates either of the following: • The performance measurement has a value, but at poor quality level. • There is no measurement value due to some error.
	Status Not Defined (No Data). There is no data for the group or monitor. This can be caused by any of the following reasons: • A new monitor has not yet run. • Monitor counters have not yet been collected. • The monitors on which the group or monitor depend are not reporting a Good condition.
	No Thresholds Breached Status. No thresholds were defined for the monitor counter, so no status is assigned.

Figure 161 SiteScope Dashboard: status and availability levels

14.7 SiteScope templates and monitoring

SiteScope Templates provide an enterprise solution for standardizing the monitoring of different IT elements in your enterprise, including servers, applications, databases, network environments, and so on. You can use templates to rapidly deploy sets of monitors that check systems in the infrastructure that shares similar characteristics. You can create and customize your own templates to meet the requirements of your organization.

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 in the network environments.

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

Note

- Make sure that the monitor-run frequency is always greater than the time taken to scale-in/scale-out a VNF. Otherwise, multiple scale-in/scale-out requests might be sent for a single scale-in/out condition.
- In Fulfillment artifact templates, each Monitor artifact should be associated with a separate Monitor Handler artifact (*even* if the handler/hypervisor is the same). One-to-one mapping should be present between a Monitor artifact and a Monitor Handler artifact.

The following table describes the objects used in templates:

Icon	Object Type	Description
	Template Container	A template container enables you to manage your template monitoring solutions. You can add a template to a template container only.
	Template	The template contains the SiteScope group, monitors, remote servers, variable definitions, and alerts that make up the template monitoring solution.
	Template Variable	A variable is used to prompt for user input during template deployment. Template variables are either user-defined or predefined system variables.
	Template Remote Server	A template remote server is used to define Windows or UNIX remote server preferences that are created when the template is deployed.
	Template Group	A template group contains the template monitors and associated alerts. You use template groups to manage the deployment of monitors and associated alerts in your infrastructure.
	Template Monitor	Template monitors are used to define monitors that are created when the template is deployed.
	Template Alert	Template alerts are used to define alerts on groups and monitors that are created when the template is deployed. If an alert has been set up for the template monitor or group, the alert symbol is displayed next to the monitor or group icon.

Table 21 SiteScope template objects

SiteScope provides template examples for monitoring in Windows and UNIX environments. These templates are available from the Template Examples folder in the template tree. You can use the template examples to help you use SiteScope templates.

The following example shows the Windows basic template. The template contains a template group, Windows monitors for %%host%%, two template monitors (CPU and Memory), four user-defined variables (host, user, password, and frequency), and a template remote server.

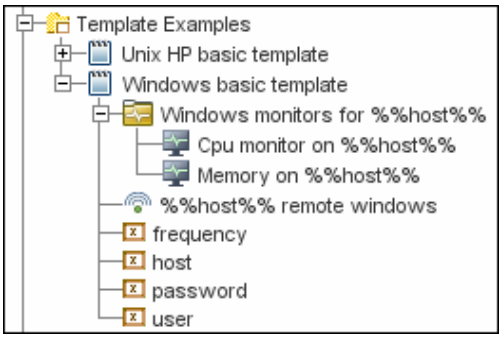


Figure 162 SiteScope: sample template

For deploying a monitor template path is a very essential input which decides which KPI has to be monitored. In the following example, the Template Path for CPU is NFVDirector/VIRTUAL_MACHINE/KVM/CPU. After triggering the respective template for deployment with the associated variables, it moves to Deployed state and this monitor can be accessed from Monitors Context.

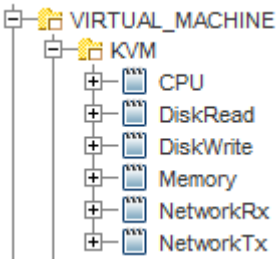


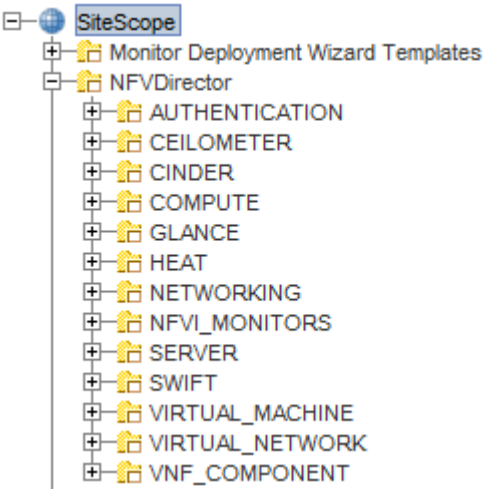
Figure 163 SiteScope: monitor context

Note

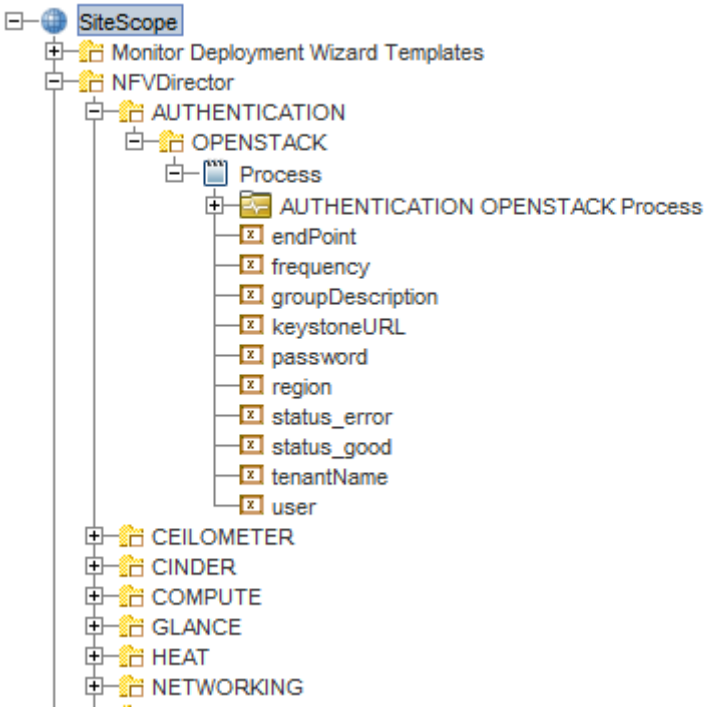
When configuring variables for Frequency and Error frequency in the Monitor Run Settings, the variable values can only be in time units of seconds.

When a monitor is copied or moved from one template to another, any user-defined variables in the monitor are also copied or moved.

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

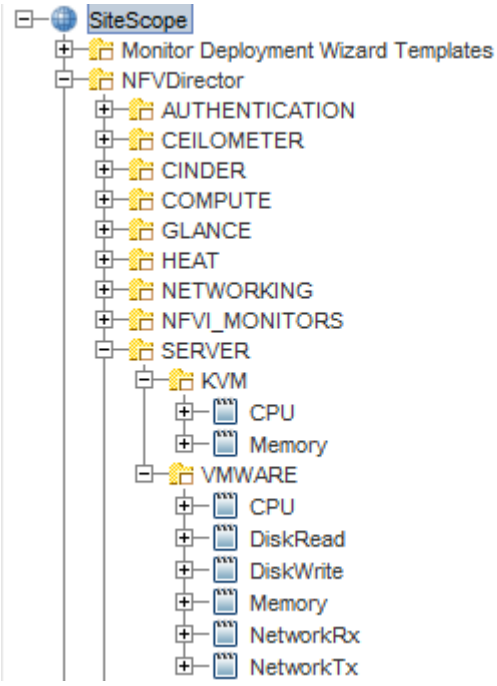


Process monitors for OpenStack services like authentication(i.e. Keystone), ceilometer, cinder, swift, heat, glance, networking(i.e. neutron):
Counters supported here is status, i.e. only status can be used in condition expression. Only ERROR and GOOD condition is supported. Status for GOOD will be set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



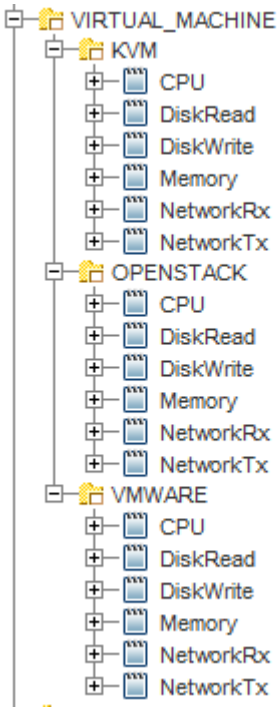
Physical Sever monitors:
Counters:

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



VirtualMachine monitors: VMWARE monitor also supports VCENTER.
Counters:

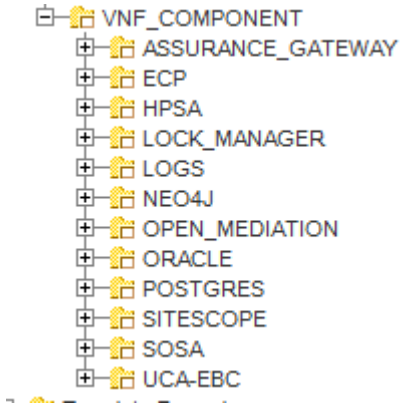
CPU: cpu_usage_average
Memory: memory_usage_average
DiskRead:disk_read_requests
DiskWrite:disk_write_requests
NetworkRx:network_bytes_received
NetworkTx:network_bytes_transmitted



SelfMonitors: Process and Log monitors are supported.

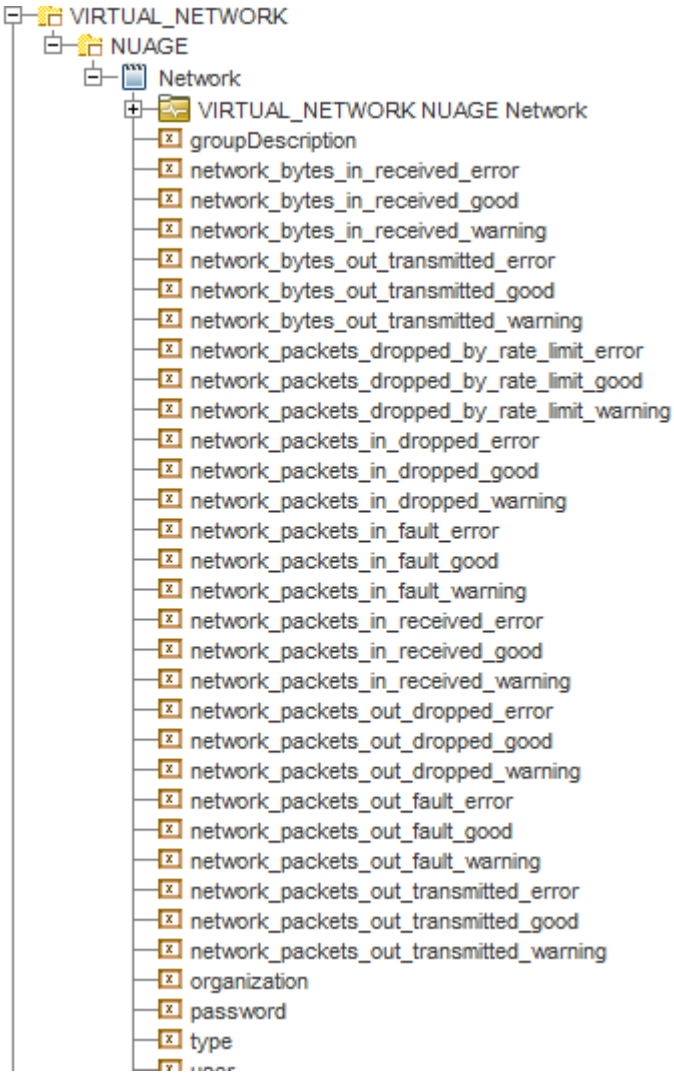
Counters: status: Value of status for error condition is <0

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



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



14.7.1 Creating custom templates

Custom templates broaden the capabilities of the regular SiteScope monitors other than the NFVD supported monitors. They help in tracking availability and performance of monitored environments. The custom templates enable you to create your own monitor by using any existing monitors provided by SiteScope.

1. Select the templates context.
2. Right-click the SiteScope root node from the tree and select **New** > **Template Container**.

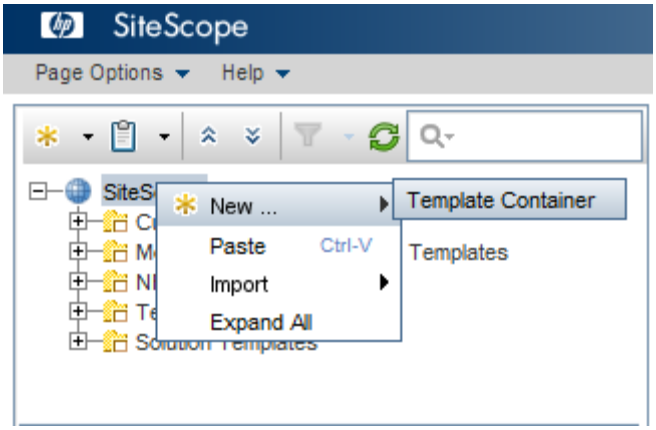


Figure 164 SiteScope: Create custom templates

3. Enter the name of the template container and click the **OK** button.
4. Right-click this new template container node and select **New** > **Template**.

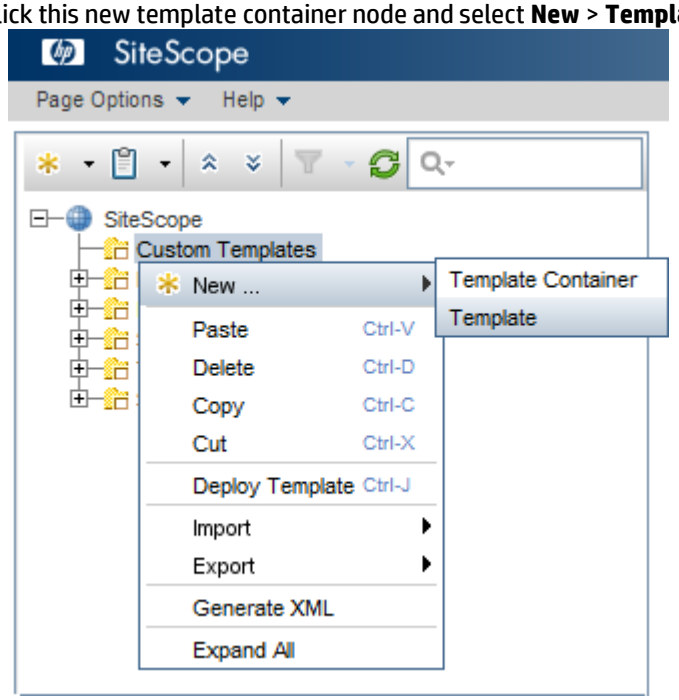


Figure 165 SiteScope: new template

5. Enter the name of the template and click the **OK** button.

   In the example provided in this section, a template container is created with the name Custom Templates.

6. Right-click this new template node and select **New** > **Group**.

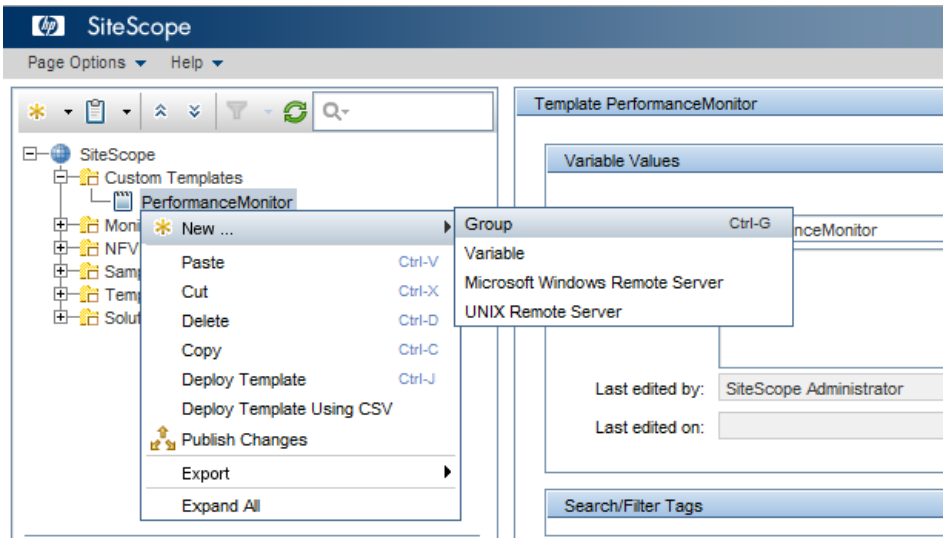


Figure 166 SiteScope: new group

7. Enter the name of the group and click the **OK** button.

8. In the example provided in this section, a template is created with name PerformanceMonitor.

9. Right-click this group node and select **New** > **Monitor**.

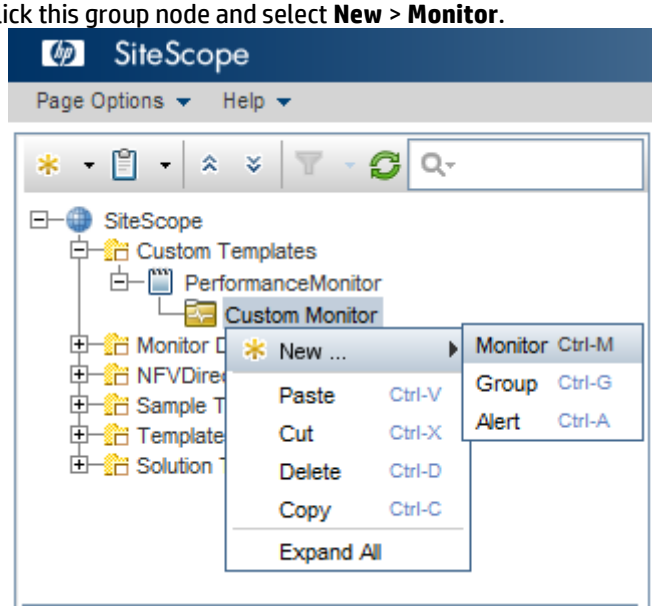


Figure 167 SiteScope: new Monitor

10. Select any one of the monitors of your choice from the list.

11. In the example provided in this section, a group is created with the name Custom Monitor.

12. Right-click the PerformanceMonitor template node and select **New** > **Variable**.

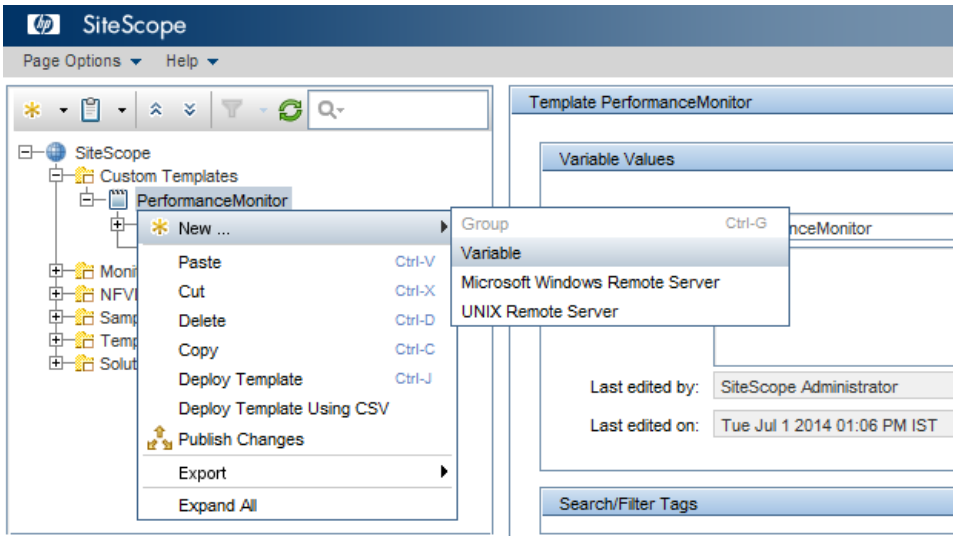


Figure 168 SiteScope: new Variable

The following window opens.

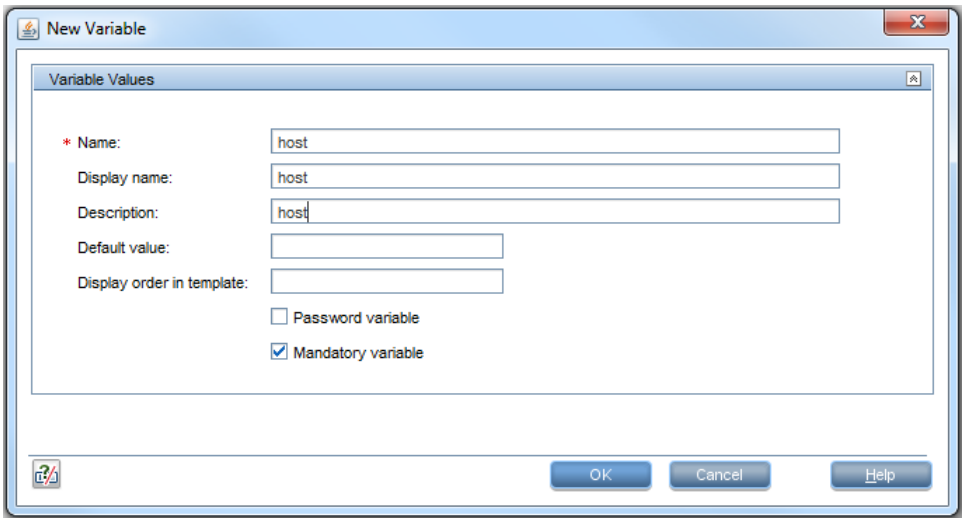


Figure 169 SiteScope: new Variable details

13. Enter the details for configuring a variable to associate it with the template.

    In the following example, the host variable is configured.

    After the variable is configured, it appears in the tree under the Template node. You can configure any number of variables. The following example shows how to use these variables in the monitor depicted.

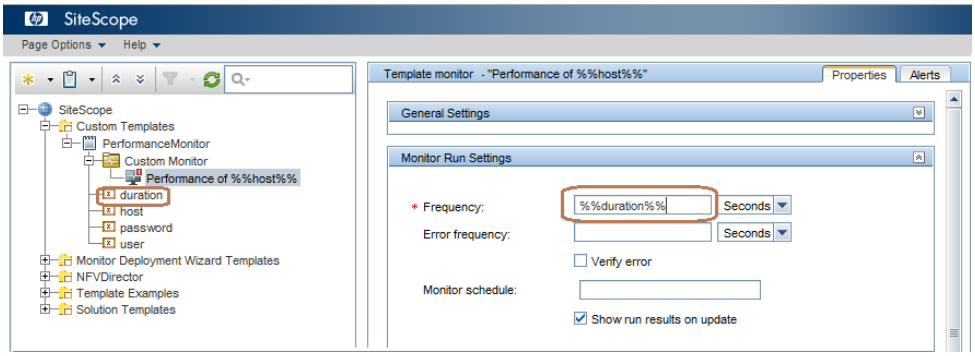


Figure 170 SiteScope: enter variable to associate with template

Following is the complete hierarchy of the Custom Template created.

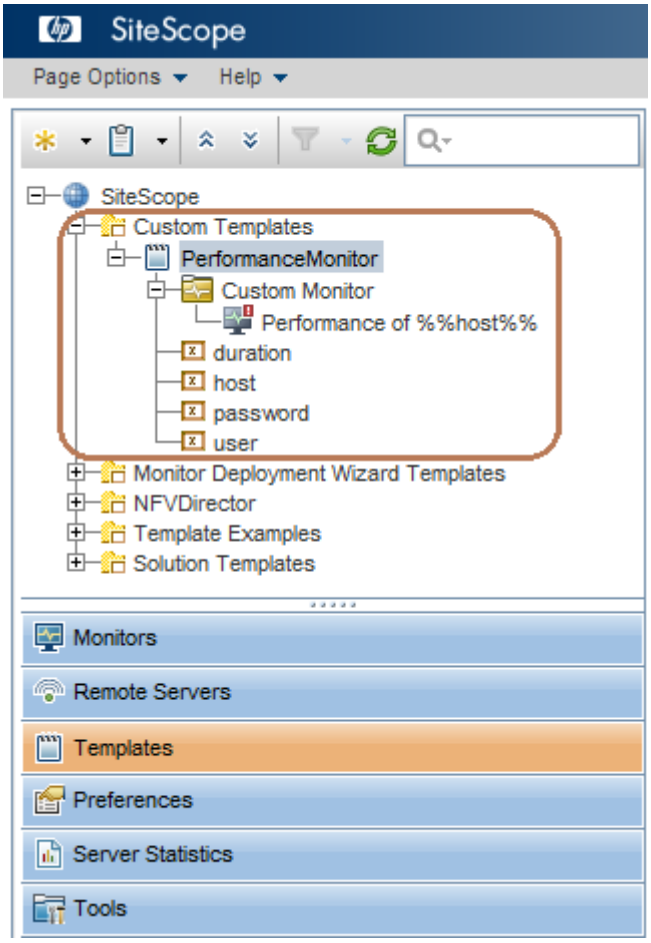


Figure 171 SiteScope: hierarchy of custom template

14. Create variables for conditions for custom monitors to achieve threshold in below format:

    a. For Error Condition: <variable>_error

    b. For Warning Condition: <variable>_warning

    c. For Good Condition: <variable>_good

For eg: If variable name is ‘network_bytes_in_received’ then we will have below template variables for threshold:

network_bytes_in_received_error

network_bytes_in_received_warning

network_bytes_in_received_good

15. To create the KPI alert, use the <variable> name used in the previous step.

    Example: network_bytes_in_received.

14.8 Alerts section

SiteScope alerts are notification actions that are triggered when the conditions for the alert definition are detected. You use an alert to send some notification of an event or change of status in some element or system in your infrastructure. For example, an alert can be triggered when a SiteScope monitor detects a change from Good to Error indicating that the monitored system has stopped responding.

- Alerts can be of three types namely Error, Warning, and Good.
- Error alert will be triggered on breach of error threshold condition and will be sent to the configured SNMP target. An error alert will be sent to the destination target only if the threshold breach has occurred at least 4 times.
- Warning alert will be triggered on breach of warning threshold condition and will be sent to the configured SNMP target. A warning alert will be sent to the destination target only if the threshold breach has occurred at least 4 times.
- Good alert will be triggered on meeting normal/safe threshold condition and will be sent to the configured SNMP target. A good alert will be sent to the destination target only if the monitored entity was previously in error condition.
- If an alert is defined for a monitor, then it is activated on that monitor only. If an alert is defined for a template, then it is activated for all the monitors in the template.
- SNMPTarget has to be configured in Preferences section. Destination address and port have to be configured to map to the endpoint to where alerts have to be sent.

14.9 Configuring an alert

This task describes the steps involved in configuring an alert definition.

14.9.1 Prerequisites

Only a SiteScope administrator user or a user granted the appropriate alerts permissions can view, create, or edit alerts.

14.9.2 Creating/copying an alert

You can create a new alert or copy an existing alert into any group or monitor container in the SiteScope tree.

14.9.2.1 Creating a new alert

1. Right-click the container to which you want to associate the alert and select New > Alert.
2. Enter a name for the alert.
3. Select the targets to trigger the alert.
4. Configure an alert action.

In the **Alert Actions** panel, click the **New Alert Action** to start the **Alert Action** wizard.

14.9.2.2 Copying an Alert Definition

1. In the **Alerts** tab, select the alert you want to copy.
2. Copy and paste it **into** the desired group or monitor container.

The alert target automatically changes to the group or monitor into which the alert is copied.

Note

If you copy an alert definition from one group container to another, the Alert targets for the pasted alert are automatically reset to include all of the children of the container into which the alert is pasted. After pasting an alert, edit the alert definition properties to be sure that the assigned Alert targets are appropriate to the new alert context and your overall alerting plan.

14.9.2.3 Testing the alert

1. Select the alert in the Alerts tab of the monitor tree.
2. Click Test.
3. Select the monitor instance you want to test and click **OK**.

   A dialog box opens with information about the alert test.

Note

The monitor you select does not have to be reporting the same status category that is selected to trigger the alert to test the alert. For example, the monitor does not have to currently be reporting an error to test an alert that is triggered by error conditions.

14.9.2.4 Disabling an alert - optional

You can disable alerts from the Alerts tab.

1. Select the alerts that you want to disable.
2. Click the **Disable** button.

   Alerts disabled from the Alerts tab cannot be triggered; this overrides the associated alerts status set for a monitor in the monitor Properties tab or Dashboard.

14.10 SNMP preferences

You use SNMP Preferences to configure the settings SiteScope needs to communicate with an external SNMP host or management console. These are the default SNMP parameters for use with SNMP Trap alerts.

To access, select **Preferences context** > **SNMP Preferences**.

Note

You must be an administrator in SiteScope, or a user granted View SNMP lists permissions to be able to view SNMP Preferences.

SNMP Preferences enable you to define settings that are used by SiteScope SNMP Trap alerts when sending data to management consoles. It also enables you to define SNMP Trap receivers, and listen to multiple local addresses and ports at the same time. SiteScope uses

the SiteScope SNMP Trap Alert type to integrate with SNMP-based network management systems.

User interface elements are described below:

UI Element	Description
General Settings	
Name	Name string assigned to the setting profile when creating a new SNMP recipient.
Description	Description for the setting profile, which appears only when editing or viewing its properties. You can include HTML tags such as the , <HR>, and <B> tags to control display format and style. **Note:** HTML code entered in this box is checked for validity and security, and corrective action is taken to fix the code (for example, code is truncated if it spans more than one line). If malicious HTML code or JavaScript is detected, the entire field is rejected. The following is prohibited HTML content: • Tags: **script**, **object**, **param**, **frame**, **iframe**. • Any tag that contains an attribute starting with **on** is declined. For example, `onhover`. • Any attribute with **javascript** as its value.
Preferences Settings: Main Settings Area	
Send to host	Domain name or IP address of the machine that receives all SNMP trap messages. This machine must be running an SNMP console to receive the trap message. **Examples:** `snmp.mydomain.com` or `206.168.191.20`.
SNMP port	SNMP port to which the trap is sent. **Default value:** 162

Table 22 SNMP User Interface Elements

Preferences Settings: SNMP Connection Settings Area	
UI Element	**Description**
Timeout (seconds)	Amount of time, in milliseconds, to wait for the SNMP trap requests (including retries) to complete. **Default value:** 5
Number of retries	Number of times each SNMP trap GET request should be retried before SiteScope considers the request to have failed. **Default value:** 1
Community	Default SNMP community name used for sending traps. The community string must match the community string used by the SNMP management console. **Default value:** public
SNMP version	Default SNMP protocol version number to use. SNMP V1 and V2c are currently supported. **Default value:** V1
Authentication algorithm	Authentication algorithm used for SNMP V3. You can select MD5, SHA, or None. **Note:** This field is available only if SNMP V3 is selected.
User name	User name to be used for authentication if you are using SNMP version 3. **Note:** This field is available only if SNMP V3 is selected.
Password	Password to be used for authentication if you are using SNMP version 3. **Note:** This field is available only if SNMP V3 is selected.

Table 23 SNMP Preference Settings

Preferences Settings: Advanced Settings Area	
UI Element	**Description**
SNMP trap ID	Select the type of trap to send. There are several predefined ID types for common conditions: • **Generic SNMP trap ID.** Select a generic SNMP type from the drop-down list. • **Enterprise-Specific SNMP trap ID.** To use an enterprise specific SNMP ID type, enter the number of the specific trap type in the box. **Note:** When integrating SiteScope with NNMi, you must select **Enterprise-Specific SNMP trap ID**, and enter **1**. SiteScope sends a different notification ID for each SNMP version: • SNMP V1: .1.3.6.1.4.1.11.15.1.4 • SNMP V2: .1.3.6.1.4.1.11.15.1.4.1
SNMP object ID	Identifies to the console the object that sent the message. • **Preconfigured SNMP object IDs.** Select one of the predefined objects from the drop-down list. • **Other SNMP object ID.** To use another object ID, enter the other object ID in the box. **Note:** • In SiteScope version 11.20 and later, all logged traps have an object ID that starts with a dot (“.”). For example, oid= .1.3.6.1.2.1.0.1.3.6.1.4.1.11.2.17.1. • When integrating SiteScope with NNMi, select **Preconfigured SNMP object IDs** and choose **HP SiteScope Event** from the list.
Add System OID as a prefix to SNMP Trap	Adds the default system OID (1.3.6.1.2.1) as a prefix to all SNMP Trap OIDs. Clear the check box if you do not want to use this prefix. **Default value:** Selected
SNMP source	The SNMP trap source: SiteScope Server or the monitor target server. **Default value:** Monitored Host

Table 24 SNMP Preferences Advanced Settings

14.11 Sending SiteScope Alerts

SiteScope triggers the alert as soon as any monitor it is associated with matches the alert trigger condition.

The following examples illustrate how different alert configurations send alerts after the error condition has persisted for more than one monitor run. If a monitor runs every 15 seconds and the alert is set to be sent after the third error reading, the alert is sent 30 seconds after the error was detected. If the monitor run interval is once every hour with the same alert setup, the alert is not sent until 2 hours later.

Example 1 - Always, after the condition has occurred at least N times:

Example 1a. An alert is sent for each time monitor is in error after condition persists for at least three monitor runs. Compare this with Example 1b below.

Alert setup	**Always, after the condition has occurred at least 3 times**								
sample interval	0	1	2	3	4	5	6	7	8
status	✓	✗	✗	✗	✗	✗	✓	✗	✗
count	c=0	c=1	c=2	c=3 alert!	c=4 alert!	c=5 alert!	c=0	c=1	c=2

Example 1b. An alert is sent for each time monitor is in error after condition persists for at least three monitor runs. Shows how the count is reset when the monitor returns one non-error reading between consecutive error readings. Compare this with Example 1a above.

Alert setup	**Always, after the condition has occurred at least 3 times**								
sample interval	0	1	2	3	4	5	6	7	8
status	✓	✗	✗	✓	✗	✗	✗	⚠	✓
count	c=0	c=1	c=2	c=0	c=1	c=2	c=3 alert!	c=0	c=0

Figure 172 SiteScope: send alerts always

Example 2 - Once, after the condition has occurred exactly N times:

An alert is sent only once if monitor is in error for at least three monitor runs, regardless of how long the error is returned thereafter.

Alert setup	**Once, after the condition has occurred exactly 3 times**								
sample interval	0	1	2	3	4	5	6	7	8
status	✓	✗	✗	✗	✗	✗	✗	✗	✗
count	c=0	c=1	c=2	c=3 alert!	c=4	c=5	c=6	c=7	c=8

Example 3 - Initially, after X times, and repeat every Y times:

Example 3a. An alert is sent on the fifth time monitor is in error and for every third consecutive error reading thereafter. Compare this with Example 3b below.

Alert setup	**Initially, after 5 times, and repeat every 3 times**								
sample interval	0	1	2	3	4	5	6	7	8
status	✓	✗	✗	✗	✗	✗	✗	✗	✗
count	c=0	c=1	c=2	c=3	c=4	c=5 alert!	c=6	c=7	c=8 alert!

Example 3b. An alert is sent on the third time monitor is in error and for every fifth consecutive error reading thereafter. Compare this with Example 3a above.

Alert setup	**Initially, after 3 times, and repeat every 5 times**								
sample interval	0	1	2	3	4	5	6	7	8
status	✓	✗	✗	✗	✗	✗	✗	✗	✗
count	c=0	c=1	c=2	c=3 alert!	c=4	c=5	c=6	c=7	c=8 alert!

Figure 173 SiteScope: send alert once

14.12 User management preferences

You can manage SiteScope user accounts from the User Management Preferences page. This page enables you to administer the users that are allowed access to SiteScope.

1. To access, select **Preferences context** >**User Management Preferences**.
2. In the **User Management Preferences** page, click the arrow next to the **New User** button and select New User.
3. In the **Main Settings** panel, enter the user name, login name, and password, and select the groups that can be accessed by this user profile.
4. Select the permissions to be granted to this user from the **Permissions** panel, or use the default permissions.

   All permissions are granted except the **Add, Edit** or **Delete** user permissions.
5. Click **OK**.

   The new user profile is added to the **User Management Preferences** list.

The following table provides UI descriptions of the **User Management Preferences** page.

UI Element	Description
	New. Click the arrow next to the button, and select: • **New User.** Creates a new user profile. • **New User Role.** Creates a new user role profile.
	Edit. Enables editing the selected user or user role profile.
	Delete User/User Role. Deletes the selected user or user role profiles.
	Copy to User Role. Enables coping an existing SiteScope user's permissions to a new user role. **Note:**SiteScope users still need to have a user login and a security group assigned to them on the LDAP server. (LDAP users have their own LDAP user name and password for logging on to SiteScope.)
	Select All. Selects all listed user and user role profiles.
	Clear Selection. Clears the selection.

Table 25 User Management Preferences

Note

Only an administrator in SiteScope or a user with add, edit, or delete user preferences permissions can create or make changes to user settings and permissions for the current user or for other users.

By default, a regular user does not have Add, edit or delete user preferences permissions. This means that they can view their own user properties only.

14.13 Managing certificates

When monitoring a remote server, if the target server uses a self-signed certificate, the certificate must be added to a trusted keystore. If you are monitoring a URL, a VMware-based server, a WebSphere Application Server, or a secure connection, you can manage self-signed certificates from the **Certificate Management** page.

Use the following procedure to import certificates:

1. Select **Preferences context** > **Certificate Management**.
2. To add certificates, click **the Import Certificates** button.

   The **Import Certificates** dialog box appears.
3. Select **File** or **Host** and enter the details of the source server.
4. From the **Loaded Certificates** table, select the server certificates that you want to import and click **Import**.

   The imported certificates are listed on the Certificate Management page.
5. To view certificate details, double-click a certificate.

Note

To view the **Certificate Management** page, you must be an administrator in SiteScope or a user granted with View certificates list permissions.

Chapter 15

Extending Action capabilities using UCA-Automation

The correlation and autonomous actions correspond to correlating traps received from NFV source and taking autonomous actions based on the NFV topology. Correlation is possible when the collection event is received from various NFV sources at the UCA EBC value pack. The event collection to UCA EBC is possible depending on the generic SNMP channel adaptor configuration.

15.1 Correlation and autonomous process

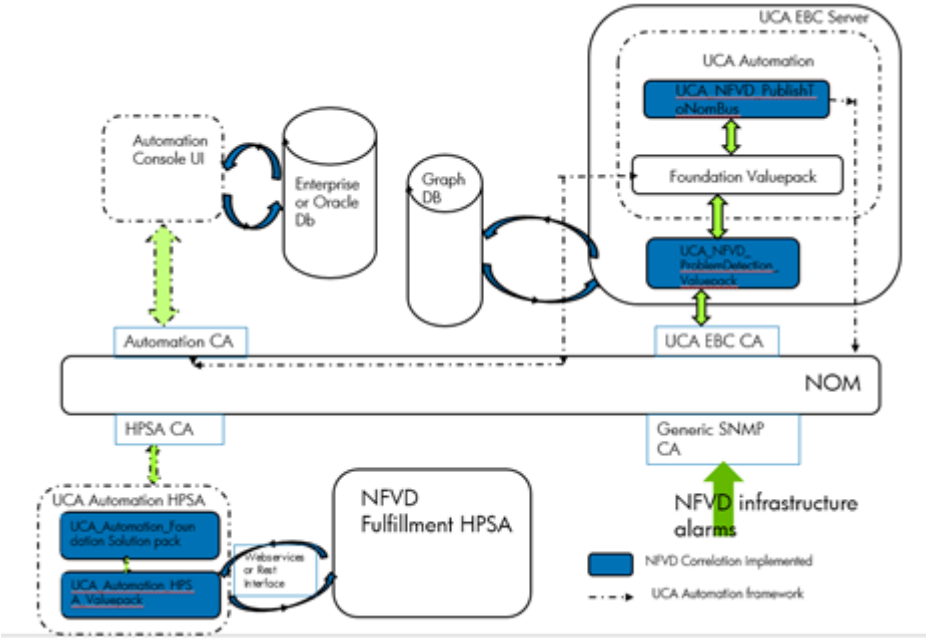


Figure 174 Correlation and autonomous action process diagram

Correlation and Autonomous actions can be categorized into the following:

- Alarm Enrichment
- Autonomous action
- Status of action and reporting

15.1.1 Alarm enrichment

The alarms received from the NFV source should be enriched with NFV topology information and parameters required for autonomous actions. The UCA NFVD Problem detection valuepack that is delivered with the NFV Director 3.0 enriches alarms. Depending on the action, it fetches the required parameters and finally publishes the alarms to Open Mediation.

The following is a sample of a raw alarm received from the NFV Director source.


```
<AlarmCreationInterface xmlns="http://hp.com/uca/expert/x733Alarm">
     <identifier>5:65d95213-00fd-4764-adfa-86b00af0881a</identifier>
<sourceIdentifier>NFVD Source</sourceIdentifier>
       <alarmRaisedTime>2014-06-23T11:55:00Z</alarmRaisedTime>
       <targetValuePack>SNMP-Customization-SiteScope-
FlowTarget</targetValuePack>
    <originatingManagedEntity>KVM_TestVM</originatingManagedEntity>
    <originatingManagedEntityStructure>
        <classInstance instance="SiteScope\HP\KVM VM CPU
Monitor\KVM_TestVM" clazz="Generic Hypervisor"/>
    </originatingManagedEntityStructure>
    <alarmType>QUALITY OF SERVICE ALARM</alarmType>
    <probableCause>calculatedCounterValue1 == 1 error</probableCause>
    <perceivedSeverity>WARNING</perceivedSeverity>
    <networkState>NOT CLEARED</networkState>
    <operatorState>NOT_ACKNOWLEDGED</operatorState>
    <problemState>NOT_HANDLED</problemState>
    <specificProblem>cpu usage medium: 1</specificProblem>
<additionalText>_customPropertiesValues=|_httpPort=8888|
         webserverAddress=15.154.72.226|
alertHelpURL=http://15.154.72.226:8080/SiteScope/sisdocs/doc lib/index.
htm?single=false&amp;context=system_avail&amp;topic=config_sis_alert|
        diagnosticTraceRoute=|errorOnly=|goodOnly=cpu usage high: 0
cpu usage low: 0|FullGroupId=HP: KVM VM CPU Monitor|group=KVM VM CPU
Monitor|
        groupdescription=CPU |
                                  groupID=201087530|
                                  id=&amp;lt;id&amp;gt;|
                                 mainStateProperties= groupID: 201087530
                 calculatedCounterValue1: 0
        calculatedCounterValue2: 1
        calculatedCounterValue3: 0|
monitorDrilldownUrl=http://ossvm8.ind.hp.com:8080/SiteScope/servlet/Mai
n?activeid=201087531&amp;activerighttop=dashboard&amp;view=new&amp;dash
board view=Details&amp;dashboard model=true&amp;sis silent login type=e
ncrypted&amp;login=%28sisp%29knjxbqDESkAn5mKcvgTmj%2FyFwHH5Ke3m&amp;pas
sword=%28sisp%29EzqXbIXEFD%2BJbE1N1T%2FZlELjja0DKaN7|

monitorServiceId=SiteScopeMonitor:201087530:201087531|
monitorTypeDisplayName=Generic Hypervisor|
monitorUUID=9a145576-cefb-483f-beaa-bd3bd6f9158f|
mountName=[/dev/mapper/VolGroup00-root_vol, /dev/mapper/VolGroup00-
root_vol (/), /dev/sda1, /dev/sda1 (/boot), /dev/mapper/VolGroup00-
home vol, /dev/mapper/VolGroup00-home vol (/home),
/dev/mapper/VolGroup00-opt_vol, /dev/mapper/VolGroup00-opt_vol (/opt),
/dev/mapper/VolGroup00-tmp_vol, /dev/mapper/VolGroup00-tmp_vol (/tmp),
/dev/mapper/VolGroup00-usr_vol, /dev/mapper/VolGroup00-usr_vol (/usr),
/dev/mapper/VolGroup00-var_vol, /dev/mapper/VolGroup00-var_vol (/var),
/dev/mapper/VolGroup00-var_crash_vol, /dev/mapper/VolGroup00-
var crash vol (/var/crash), /dev/mapper/VolGroup00-var log audit,
/dev/mapper/VolGroup00-var_log_audit (/var/log/audit)]|
```

```
multiViewUrl=http://15.154.72.226:8080/SiteScope/MultiView|
fullMonitorName=SiteScope\HP\KVM VM CPU Monitor\KVM_TestVM|
newSiteScopeURL=http://15.154.72.226:8080/SiteScope|sample=5|
SiteScopeBaseUrl=http://ossvm8.ind.hp.com:8080|
SiteScopeHost=ossvm8.ind.hp.com|
SiteScopeURL=http://15.154.72.226:8080/SiteScope|
SiteScopeuserurl=http://15.154.72.226:8080/SiteScope/userhtml/SiteScope
.html|
state=Virttop Management/Domains
                Information/KVM_TestVM/Performance/%CPU=0.1, ,
cpu_usage_high=0,
                cpu usage medium=1, cpu usage low=0|
tag=|targetHost=sheep.gre.hp.com|
targetIP=16.16.94.139|
targetIPVersion=IPV4|
templateDeployPath=HP/KVM VM CPU Monitor|
time=11:54 AM 6/23/14|
warningOnly= cpu usage medium: 1|customerId=&amp;lt;customerId&amp;gt;
     </additionalText>
</AlarmCreationInterface>
```

The following is an enriched alarm, after processing by UCA NFVD Problem Detection.

```
Valuepack.
- alarmRaisedTime                = 2014-05-05T20:41:00.848+05:30
    - sourceIdentifier               = NFVD_Source
    - originatingManagedEntity       = KVM TestVM
    - originatingManagedEntityStructure
            -> Host = ossvm1.ind.hp.com
    - alarmType                      = QUALITY OF SERVICE ALARM
    - probableCause                  = UtilizationPercentage
    - perceivedSeverity              = CRITICAL
    - networkState                   = NOT CLEARED
    - operatorState                  = NOT_ACKNOWLEDGED
    - problemState                   = NOT_HANDLED
    - problemInformation             = Attribute not available
    - specificProblem                = ERROR
    - additionalInformation          = null
    - additionalText                 = SiteScope
alarm|MONITOR.cpuMonitor-001|CONDITION=ERROR|group=KVM VM CPU
Monitor|groupdescription=CPU|groupID=201070542|id=1
    - proposedRepairActions          = null
    - notificationIdentifier         = 0
    - correlationNotificationIdentifiers = Attribute not available
    - timeInMilliseconds             = 1399302660848 [2014/05/05
20:41:00.848 +0530]
    - targetValuePack                = null
    - sourceScenarios                =
[com.hp.uca.expert.vp.pd.ProblemDetection]
    - passingFilters                 = [NfvdScenario]
    - passingFiltersTags             = {NfvdScenario=[Trigger,
ProblemAlarm, SubAlarm]}
    - passingFiltersParams           = {NfvdScenario={}}
    - hasParents                     = false
    - parentsNumber                  = 0
    - parents                        = null
    - hasChildren                    = false
    - childrenNumber                 = 0
    - children                       = null
    - justInjected                   = false
    - aboutToBeRetracted             = false
    - hasStateChanged                = false
```

```
    - stateChanges                    = none
    - hasAVCChanged                   = false
    - attributeValueChanges           = none
    - customFields
      -> userText = NFVD-PD
      -> NFVTopology =
<Start>
<VIRTUAL_MACHINE>
artifactCategory=GENERIC;
GENERAL.Description=A Virtual machine;
artifactId=KVMVM-2001;
GENERAL.Name=KVM_TestVM;
SERVICE_ACTION=CREATE;
templateId=template-512;
GENERAL.Type=VMtype;
SERVICE_TYPE=NFVD;
SERVICE_NAME=INSART;
artifactFamily=VIRTUAL_MACHINE;
GENERAL.hostname=KVM_TestVM;
GENERAL.Management_access=http://vm1.ind.com:8080;
</VIRTUAL_MACHINE>
<VNF_COMPONENT>
artifactCategory=GENERIC;
GENERAL.Description=This is VNFC component;
artifactId=VNFC-BLR1;
GENERAL.Name=vnfc-BNGALORE;
SERVICE_ACTION=CREATE;
templateId=template-8001;
lastUpdateTimestamp=15-04-2014 12:58;
creationTimestamp=15-04-2014 12:57;
SERVICE_TYPE=NFVD;
SERVICE_NAME=INSART;
artifactFamily=VNF_COMPONENT;
</VNF_COMPONENT>
<VIRTUAL_PORT>
artifactCategory=GENERIC;
artifactId=Ethe.1990;
SERVICE_ACTION=CREATE;
templateId=template-512;
lastUpdateTimestamp=15-04-2014 12:58;
SERVICE_TYPE=NFVD;
SERVICE_NAME=INSART;
artifactFamily=VIRTUAL_PORT;
INFO.Speed=10;
INFO.Name=EthernetPort-10GB;
INFO.ID=Ethe.1990;
INFO.Type=Port;
INFO.MAC=12:34:56:78:9A:BD;
</VIRTUAL_PORT>
<VIRTUAL_LUN>
artifactCategory=GENERIC;
artifactId=LUN-1001;
SERVICE_ACTION=CREATE;
templateId=template-512;
lastUpdateTimestamp=15-04-2014 12:58;
SERVICE_TYPE=NFVD;
SERVICE_NAME=INSART;
artifactFamily=VIRTUAL_LUN;
INFO.Name=LUN-1.2-BLR;
INFO.Amount=30;
INFO.ID=LUN-1001;
INFO.Type=Storage;
</VIRTUAL_LUN>
```

```
<VIRTUAL DISK>
artifactCategory=GENERIC;
artifactId=vDisk1;
SERVICE ACTION=CREATE;
templateId=template-512;
lastUpdateTimestamp=15-04-2014 12:58;
SERVICE TYPE=NFVD;
SERVICE_NAME=INSART;
artifactFamily=VIRTUAL_DISK;
INFO.Name=Seagate-500GB-SATA;
INFO.Amount=2;
INFO.ID=PSATA-500;
INFO.Type=Memory;
</VIRTUAL_DISK>
<VIRTUAL_MEMORY>
artifactCategory=GENERIC;
artifactId=RAM-4001;
SERVICE_ACTION=CREATE;
templateId=template-512;
lastUpdateTimestamp=15-04-2014 12:58;
creationTimestamp=15-04-2014 12:57;
SERVICE TYPE=NFVD;
SERVICE_NAME=INSART;
artifactFamily=VIRTUAL_MEMORY;
INFO.Name=DDR-RAM-8GB;
INFO.Amount=1;
INFO.ID=RAM-4001;
INFO.Type=Memory;
</VIRTUAL_MEMORY>
<VIRTUAL CORE>
artifactCategory=GENERIC;
artifactId=VCore-1001;
SERVICE ACTION=CREATE;
templateId=template-512;
lastUpdateTimestamp=15-04-2014 12:58;
SERVICE TYPE=NFVD;
SERVICE_NAME=INSART;
artifactFamily=VIRTUAL_CORE;
INFO.Speed=2233;
INFO.Name=VCore-I5;
INFO.Amount=3;
INFO.ID=Serial-VCore-1001;
INFO.Type=Virtual Core;
</VIRTUAL_CORE>
<MONITOR>
artifactCategory=GENERIC;
GENERAL.Description=Network Monitor;
artifactId=networkMonitor-004;
GENERAL.Name=Network;
SERVICE_ACTION=CREATE;
templateId=template-004;
lastUpdateTimestamp=30-04-2014 12:57;
creationTimestamp=30-04-2014 12:57;
SERVICE TYPE=NFVD;
SERVICE_NAME=INSART;
artifactFamily=MONITOR;
GENERAL.Frequency=30;
GENERAL.DeploymentPath=;
</MONITOR>
<MONITOR>
artifactCategory=GENERIC;
GENERAL.Description=Disk Monitor;
artifactId=diskMonitor-003;
```

```
GENERAL.Name=Disk;
SERVICE_ACTION=CREATE;
templateId=template-003;
lastUpdateTimestamp=30-04-2014 12:57;
creationTimestamp=30-04-2014 12:57;
SERVICE_TYPE=NFVD;
SERVICE NAME=INSART;
artifactFamily=MONITOR;
GENERAL.Frequency=30;
GENERAL.DeploymentPath=HP/NFVD/NS-Routing/VNF-K/VNFC-K;
</MONITOR>
<MONITOR>
artifactCategory=GENERIC;
GENERAL.Description=Memory Monitor;
artifactId=memoryMonitor-002;
GENERAL.Name=Memory;
SERVICE_ACTION=CREATE;
templateId=template-002;
lastUpdateTimestamp=30-04-2014 12:57;
creationTimestamp=30-04-2014 12:57;
SERVICE_TYPE=NFVD;
SERVICE NAME=INSART;
artifactFamily=MONITOR;
GENERAL.Frequency=30;
GENERAL.DeploymentPath=HP/NFVD/NS-Routing/VNF-Z/VNFC-Z;
</MONITOR>
<MONITOR>
artifactCategory=GENERIC;
GENERAL.Description=CPU Monitor;
artifactId=cpuMonitor-001;
GENERAL.Name=CPU;
SERVICE_ACTION=CREATE;
templateId=template-001;
lastUpdateTimestamp=30-04-2014 12:57;
creationTimestamp=30-04-2014 12:57;
SERVICE TYPE=NFVD;
SERVICE_NAME=INSART;
artifactFamily=MONITOR;
GENERAL.Frequency=30;
GENERAL.DeploymentPath=HP/NFVD/NS-Routing/VNF-Y/VNFC-Y;
</MONITOR>
<End>
      -> Evp = UCA_NFVD_PublishToNomBus
      -> Evpversion = 1.0
      -> Evpscenario = publishToNomBus
      -> Problem = NFVD:SCALE_OUT_CPU
      -> Parameternames = ArtifactInstanceId
      -> Parametervalues = KVMVM-2001
      -> Resourceinstance = KVMVM-2001
      -> Resourcetype = INVENTORY
      -> Actionpreset = true
      -> Action = SCALE_OUT
```

15.1.2 Autonomous action

The Autonomous action of NFV Director is based on UCA Automation Action Framework. For information on Automation framework, refer to the *HP UCA Automation - Administrator and User Interface Guide*.

The mandatory alarm attributes required for Autonomous action are provided in the following table.

Alarm Attribute Name	Alarm Attribute Value
Evp	UCA_NFVD_PublishToNomBus
Evpversion	1.0
Evpscenario	publishToNomBus
Problem	NFVD:<Problem Name as per UCA Automation>
Action	SCALE_IN | SCALE_OUT | SCALE_UP | SCALE_DOWN | SCRIPT If the Action parameter is not available, the Autonomous action is not triggered. The value is fetched from the Graph Database, depending on the raw alarm received.
Parameternames	< Value depending on Action Type>
Parametervalues	< Value depending on Action Type>
Resourcetype	INVENTORY
Actionpreset	TRUE

Table 26 Alarm attributes for Autonomous action

15.1.3 Status of Action and Reporting

The status of action and reporting is available in UCA Automation console.

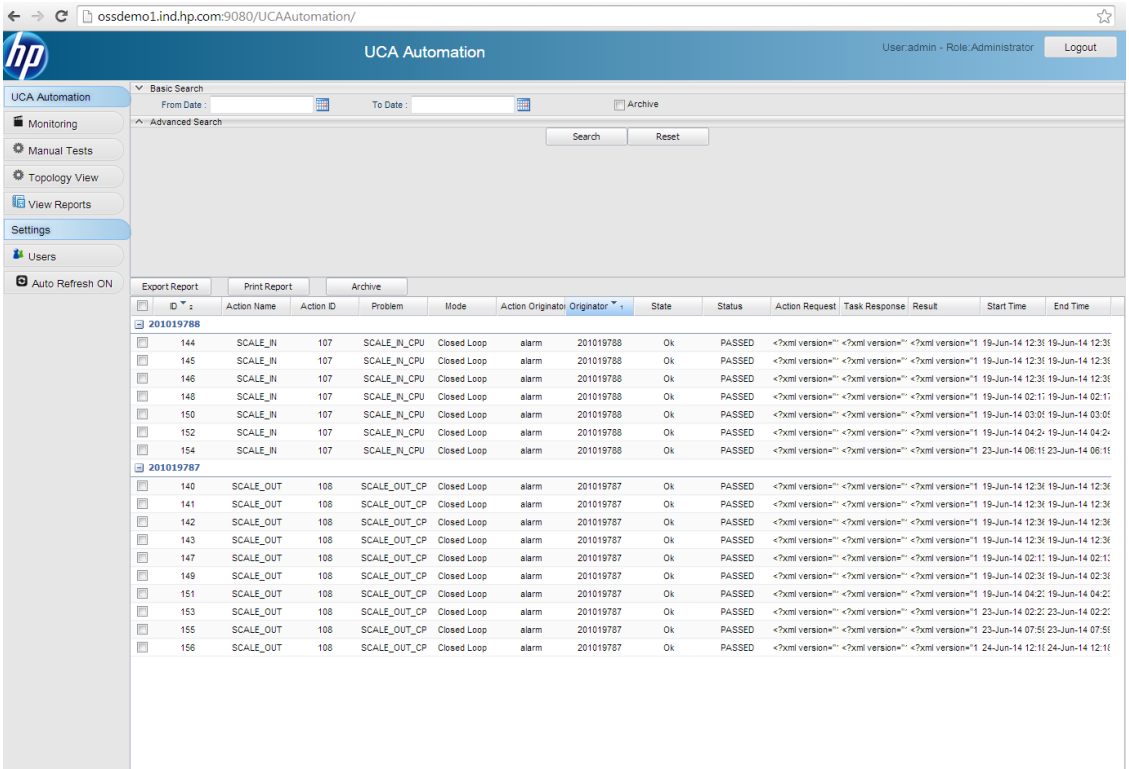
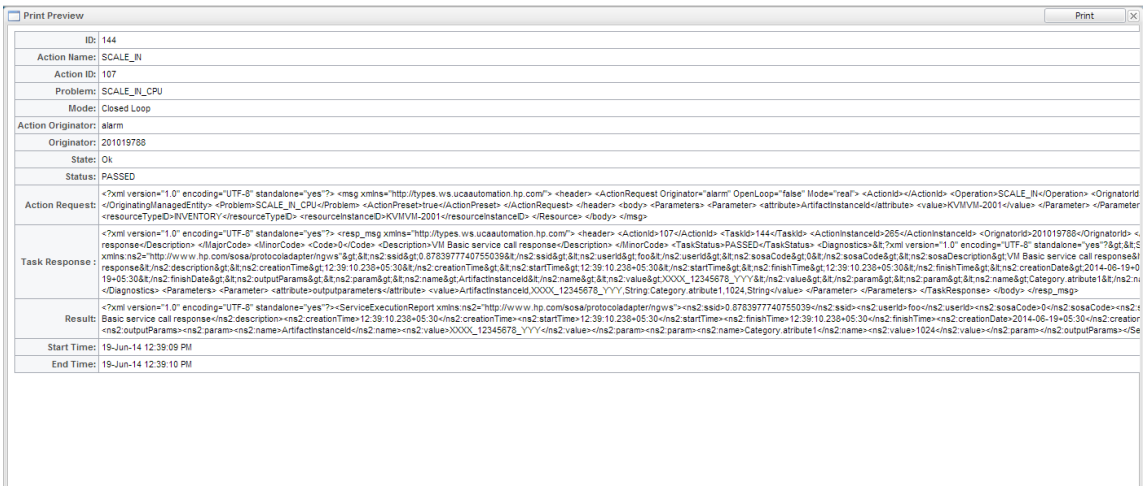


Figure 175 Snapshot of Status of action



Print Preview

Print

ID:	144
Action Name:	SCALE_IN
Action ID:	107
Problem:	SCALE_IN_CPU
Mode:	Closed Loop
Action Originator:	alarm
Originator:	201019788
State:	Ok
Status:	PASSED
Action Request:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <msg xmlns="http://types.ws.ucaautomation.hp.com/"> <header> <ActionRequest Originator="alarm" OpenLoop="false" Mode="real"> <ActionId></ActionId> <Operation>SCALE_IN</Operation> <OriginatorId />OriginatingManagedEntity> <Problem>SCALE_IN_CPU</Problem> <ActionPreset>true</ActionPreset> </ActionRequest> </header> <body> <Parameters> <Parameter> <attribute>ArtifactInstanceId</attribute> <value>KVMVM-2001</value> </Parameter> </Parameter <resourceTypeID>INVENTORY</resourceTypeID> <resourceInstanceID>KVMVM-2001</resourceInstanceID> </Resource> </body> </msg>
Task Response :	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <resp_msg xmlns="http://types.ws.ucaautomation.hp.com/"> <header> <ActionId>107</ActionId> <TaskId>144</TaskId> <ActionInstanceId>265</ActionInstanceId> <OriginatorId>201019788</OriginatorId> <. response</Description> </MajorCode> <MinorCode> <Code>0</Code> <Description>VM Basic service call response</Description> </MinorCode> <TaskStatus>PASSED</TaskStatus> <Diagnostics><?xml version="1.0" encoding="UTF-8" standalone="yes"?><S xmlns:ns2="http://www.hp.com/sosa/protocoladapter/ngws"><ns2:ssid>0.8783977740755039</ns2:ssid><ns2:userId>foo</ns2:userId><ns2:sosaCode>0</ns2:sosaCode><ns2:sosaDescription>VM Basic service call response< response</ns2:description><ns2:creationTime>12:39:10.238+05:30</ns2:creationTime><ns2:startTime>12:39:10.238+05:30</ns2:startTime><ns2:finishTime>12:39:10.238+05:30</ns2:finishTime><ns2:creationDate>2014-06-19+0 19+05:30</ns2:finishDate><ns2:outputParams><ns2:param><ns2:name>ArtifactInstanceId</ns2:name><ns2:value>XXXX_12345678_YYY</ns2:value></ns2:param><ns2:param><ns2:name>Category.atribute1</ns2:na </Diagnostics> <Parameters> <Parameter> <attribute>outputparameters</attribute> <value>ArtifactInstanceId,XXXX_12345678_YYY,String:Category.atribute1,1024,String</value> </Parameter> </Parameters> </TaskResponse> </body> </resp_msg>
Result:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?><ServiceExecutionReport.xmlns:ns2="http://www.hp.com/sosa/protocoladapter/ngws"><ns2:ssid>0.8783977740755039</ns2:ssid><ns2:userId>foo</ns2:userId><ns2:sosaCode>0</ns2:sosaCode><ns2: Basic service call response</ns2:description><ns2:creationTime>12:39:10.238+05:30</ns2:creationTime><ns2:startTime>12:39:10.238+05:30</ns2:startTime><ns2:finishTime>12:39:10.238+05:30</ns2:finishTime><ns2:creationDate>2014-06-19+05:30</ns2:creatior <ns2:outputParams><ns2:param><ns2:name>ArtifactInstanceId</ns2:name><ns2:value>XXXX_12345678_YYY</ns2:value></ns2:param><ns2:param><ns2:name>Category.atribute1</ns2:name><ns2:value>1024</ns2:value></ns2:param></ns2:outputParams></Se
Start Time:	19-Jun-14 12:39:09 PM
End Time:	19-Jun-14 12:39:10 PM

Figure 176 Snapshot of Reporting of action

Chapter 16

Discovery and Reconciliation

Resource tree can be either manually created or discovered from the VIMs in the infrastructure.

If we have the credentials to the VIM, the same can be modelled in the NFVD Artifact Instances to enable discovery.

16.1 Capacity Management

Capacity management is an independent component provides the resource tree information for NFV-Director, It consists of VIM specific channel adapter such as openstack-ca and a reconciliation channel adapter, that can be deployed on Open Mediation.

- Discovery Module : Interacts with VIM such as openstack and queries for resource information and parse the json response into artifact-relationship model
- Reconciliation Module : . Channel adapter will fetch Openstack VIM & Authentication artifact instances from fulfillment via REST API and trigger discovery module by passing the VIM credentials. Discovery module will send response in artifact-relationship instance model. Once the response is received, Reconciliation will query fulfillment to get the existing capacity information and then it formulates logic and builds delta information to reconcile. The final data will be prepared and persisted to fulfillment via REST API's

Below is the pictorial diagram that explains the design approach of Capacity Management

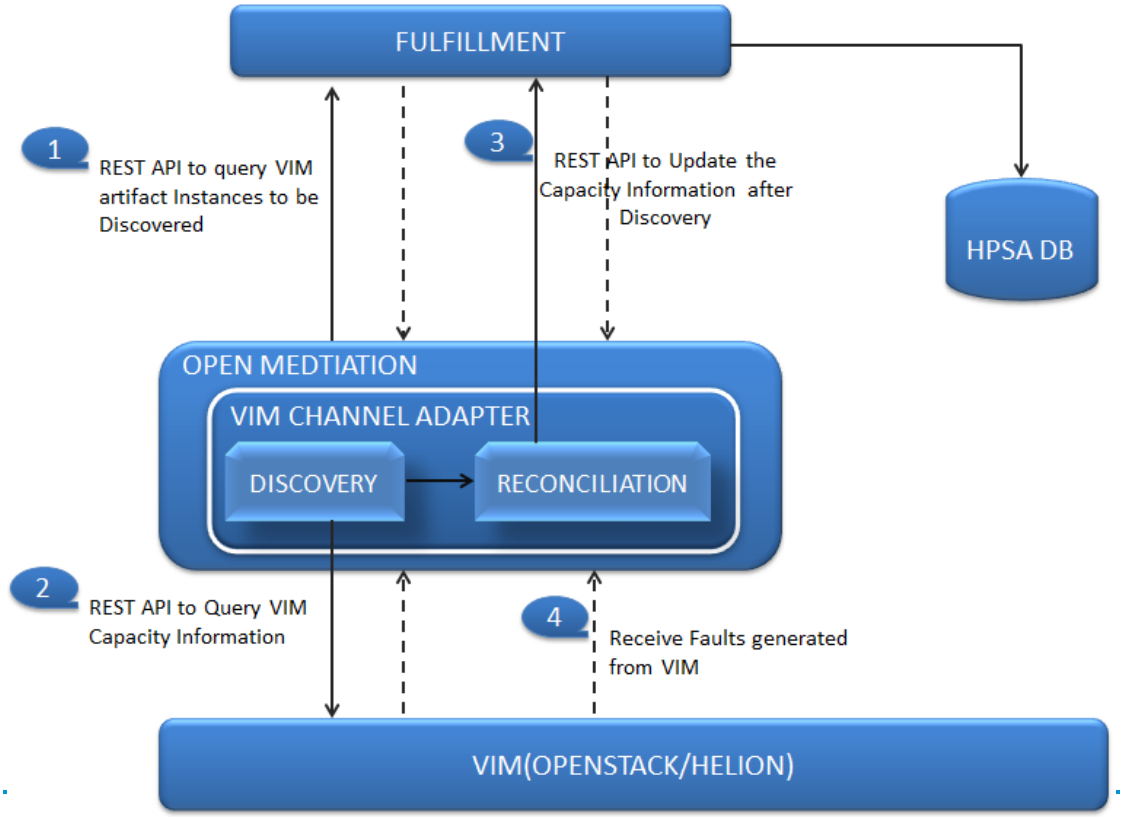


Figure 177 Capacity Management Design Diagram

Mandatory VIM configuration artifact instances has to be uploaded to fulfillment for capacity management to perform its task.

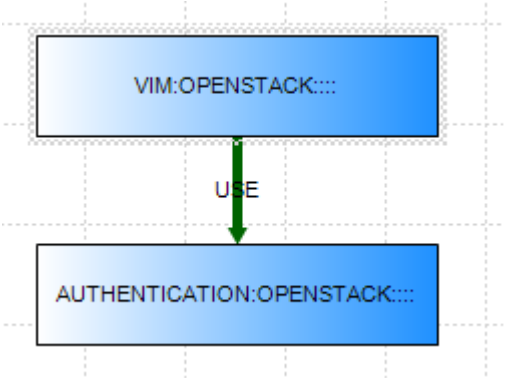


Figure 178 VIM and Authentication Artifact Instances

Attrtibutes requires for capacity management in VIM and AUTHENTICATION artifact as below.

1. `VIM > CAPACITY_DISCOVERY_PARAMS > NOM_LoadBalance_Instance`: Open Mediation instance number where the vim discovery should be handled.
2. `VIM > CAPACITY_DISCOVERY_PARAMS > Enable_Log_Entry: true/false`. Setting it to true captures the discovery information into log file.
3. `VIM > CAPACITY_DISCOVERY_PARAMS > Enable_REST_Call. true/false`. Setting it to true creates/updates/deletes fulfillment database directly through rest interface
4. AUTHENTICATION > CREDENTIALS > Url

   Example: An OpenStack URL: http://localhost:5000/v2.0/tokens
5. `AUTHENTICATION > CREDENTIALS > Login`. Openstack admin user name (Admin credentials are recommended to get physical information from openstack juno)
6. AUTHENTICATION > CREDENTIALS > Password. Openstack admin password
7. AUTHENTICATION > CREDENTIALS > TenantName
8. `AUTHENTICATION > CAPACITY > DiscoverTenants`. In case discovery module has to discover particular tenants only then comma separated tenant names has to be entered here for e.g: admin,demo, for all tenants leave it blank(Recommended leave it blank)

Sample VIM & Authentication artifact:

```
<instance xmlns="http://www.hp.com/nfvd">
<artifact-instances>
        <artifact-instance>
                <artifact-version>1</artifact-version>
```

```
                <artifact-definition>
                        <category>OPENSTACK</category>
                        <family>VIM</family>
                </artifact-definition>
                <status>
                        <enabled>true</enabled>
                        <label>ENABLED</label>
                        <visible-label>ENABLED</visible-label>
                </status>
                <categories>
                        <category>
                                <attributes>
                                        <attribute>
                                                <label>Login</label>
                                                <mandatory>true</mandatory>
                                                <order>1</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                                <value>admin</value>
                                        </attribute>
                                        <attribute>
                                                <label>Password</label>
                                                <mandatory>true</mandatory>
                                                <order>2</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                                <value>password</value>
                                        </attribute>

                                </attributes>
                                <label>CREDENTIALS</label>
                                <order>3</order>
                        </category>
                        <category>
                                <attributes>
                                        <attribute>

<label>NOM_LoadBalance_Instance</label>
                                                <mandatory>true</mandatory>
                                                <order>1</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                                <value>0</value>
                                        </attribute>
                                        <attribute>

<label>Enable_Log_Entry</label>
                                                <mandatory>false</mandatory>
                                                <order>3</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                                <value>TRUE</value>
                                        </attribute>
                                        <attribute>

<label>Enable_Rest_Call</label>
                                                <mandatory>false</mandatory>
                                                <order>2</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                                <value>TRUE</value>
                                        </attribute>
                                </attributes>
```

```
                                <label>CAPACITY_DISCOVERY_PARAMS</label>
                                <order>4</order>
                        </category>
                        <category>
                                <attributes>

                                        <attribute>
                                                <label>Url</label>
                                                <mandatory>false</mandatory>
                                                <order>3</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>

<value>http://localhost</value>
                                        </attribute>
                                        <attribute>
                                                <label>Name</label>
                                                <mandatory>true</mandatory>
                                                <order>1</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>

<value>MultiRegionSetup</value>
                                        </attribute>
                                        <attribute>
                                                <label>Type</label>
                                                <mandatory>false</mandatory>
                                                <order>2</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                        </attribute>
                                        <attribute>
                                                <label>Host</label>
                                                <mandatory>true</mandatory>
                                                <order>1</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                                <value>hostname</value>
                                        </attribute>
                                </attributes>
                                <label>GENERAL</label>
                                <order>1</order>
                        </category>
                        <category>
                                <attributes>
                                        <attribute>
                                                <label>additionalText</label>
                                                <description>additional
information about the status </description>
                                                <mandatory>false</mandatory>
                                                <order>6</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                        </attribute>
                                        <attribute>

<label>Operational_Status_Date</label>
                                                <description>Last status
update date (UTC)</description>
                                                <mandatory>false</mandatory>
                                                <order>2</order>
                                                <type>Date</type>
                                                <unit>Date</unit>
```

```
                                        </attribute>
                                        <attribute>

<label>Operational_Status</label>
                                                <description>Operational
Status</description>
                                                <mandatory>false</mandatory>
                                                <order>1</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                        </attribute>
                                        <attribute>

<label>LifeCycle_State_Date</label>
                                                <description>Last Life cycle
state change date (UTC)</description>
                                                <mandatory>false</mandatory>
                                                <order>5</order>
                                                <type>Date</type>
                                                <unit>Date</unit>
                                        </attribute>
                                        <attribute>
                                                <label>Source</label>
                                                <description>Denotes the
origin of state change</description>
                                                <mandatory>true</mandatory>
                                                <order>3</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                                <value>TEXT</value>
                                        </attribute>
                                        <attribute>
                                                <label>LifeCycle_State</label>
                                                <description>Life Cycle
State</description>
                                                <mandatory>false</mandatory>
                                                <order>4</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                        </attribute>
                                </attributes>
                                <label>STATUS</label>
                                <order>2</order>
                        </category>
                </categories>
                <creation-timestamp>2015-06-11T12:10:15.800+0200</creation-
timestamp>
                <id>vim-discovery-001</id>
                <identifier>vim-discovery-001</identifier>
                <physical>false</physical>
                <update-timestamp>2015-06-11T12:10:15.800+0200</update-
timestamp>
        </artifact-instance>
        <artifact-instance>
                <artifact-version>1</artifact-version>
                <artifact-definition>
                        <category>OPENSTACK</category>
                        <family>AUTHENTICATION</family>
                </artifact-definition>
                <status>
                        <enabled>true</enabled>
                        <label>ENABLED</label>
                        <visible-label>ENABLED</visible-label>
```

```
                    </status>
                    <categories>
                            <category>
                                    <attributes>
                                            <attribute>
                                                    <label>Url</label>
                                                    <mandatory>true</mandatory>
                                                    <order>1</order>
                                                    <type>TEXT</type>
                                                    <unit>TEXT</unit>

<value>http://localhost:5000/v2.0/tokens</value>
                                            </attribute>
                                            <attribute>
                                                    <label>Login</label>
                                                    <mandatory>true</mandatory>
                                                    <order>2</order>
                                                    <type>TEXT</type>
                                                    <unit>TEXT</unit>
                                                    <value>admin</value>
                                            </attribute>
                                            <attribute>
                                                    <label>Password</label>
                                                    <mandatory>true</mandatory>
                                                    <order>3</order>
                                                    <type>TEXT</type>
                                                    <unit>TEXT</unit>
                                                    <value>devstack</value>
                                            </attribute>
                                            <attribute>
                                                    <label>TenantName</label>
                                                    <order>3</order>
                                                    <mandatory>true</mandatory>
                                                    <value>admin</value>
                                                    <type>TEXT</type>
                                                    <unit>TEXT</unit>
                                            </attribute>
                                    </attributes>
                                    <label>CREDENTIALS</label>
                                    <order>2</order>
                            </category>
                            <category>
                                    <attributes>
                                            <attribute>
                                                    <label>Type</label>
                                                    <mandatory>false</mandatory>
                                                    <order>3</order>
                                                    <type>TEXT</type>
                                                    <unit>TEXT</unit>
                                                    <value>Type</value>
                                            </attribute>
                                            <attribute>
                                                    <label>Description</label>
                                                    <mandatory>false</mandatory>
                                                    <order>2</order>
                                                    <type>TEXT</type>
                                                    <unit>TEXT</unit>
                                                    <value>Description</value>
                                            </attribute>
                                            <attribute>
                                                    <label>Name</label>
                                                    <mandatory>false</mandatory>
                                                    <order>1</order>
```

```
                                                        <type>TEXT</type>
                                                        <unit>TEXT</unit>
                                                        <value>Name of the
artifact</value>
                                                </attribute>
                                        </attributes>
                                        <label>GENERAL</label>
                                        <order>1</order>
                                </category>
                                <category>
                                        <attributes>
                                                <attribute>
                                                        <label>DiscoverTenants</label>
                                                        <description>comma separated
values of tenant e.g: admin,demo ,leave it  blank for all tenants</description>
                                                        <mandatory>false</mandatory>
                                                        <order>1</order>
                                                        <type>TEXT</type>
                                                        <unit>TEXT</unit>
                                                </attribute>
                                        </attributes>
                                        <label>CAPACITY</label>
                                        <order>4</order>
                                </category>
                                <category>
                                        <attributes>
                                                <attribute>
                                                        <label>Source</label>
                                                        <mandatory>true</mandatory>
                                                        <order>3</order>
                                                        <type>TEXT</type>
                                                        <unit>TEXT</unit>
                                                        <value>internal</value>
                                                </attribute>
                                                <attribute>

<label>Operational_Status_Date</label>
                                                        <mandatory>false</mandatory>
                                                        <order>2</order>
                                                        <type>Date</type>
                                                        <unit>Date</unit>
                                                </attribute>
                                                <attribute>
                                                        <label>LifeCycle_State</label>
                                                        <mandatory>false</mandatory>
                                                        <order>4</order>
                                                        <type>TEXT</type>
                                                        <unit>TEXT</unit>
                                                </attribute>
                                                <attribute>
                                                        <label>additionalText</label>
                                                        <mandatory>false</mandatory>
                                                        <order>6</order>
                                                        <type>TEXT</type>
                                                        <unit>TEXT</unit>
                                                </attribute>
                                                <attribute>

<label>LifeCycle_State_Date</label>
                                                        <mandatory>false</mandatory>
                                                        <order>5</order>
                                                        <type>Date</type>
                                                        <unit>Date</unit>
```

```
                                        </attribute>
                                        <attribute>

<label>Operational_Status</label>
                                                <mandatory>false</mandatory>
                                                <order>1</order>
                                                <type>TEXT</type>
                                                <unit>TEXT</unit>
                                        </attribute>
                                </attributes>
                                <label>STATUS</label>
                                <order>3</order>
                        </category>
                </categories>
                <creation-timestamp>2015-06-11T12:10:15.587+0200</creation-
timestamp>
                <id>auth-discovery-001</id>
                <identifier>auth-discovery-001</identifier>
                <physical>false</physical>
                <update-timestamp>2015-06-11T12:10:15.987+0200</update-
timestamp>
        </artifact-instance>
</artifact-instances>
<relationship-instances xmlns="http://www.hp.com/nfvd">
        <relationship-instance>
                <categories/>
                <child-artifact-id>auth-discovery-001</child-artifact-id>
                <creation-timestamp>2015-06-12T11:20:37.312+0530</creation-
timestamp>
                <parent-artifact-id>vim-discovery-001</parent-artifact-id>
                <status>
                        <enabled>true</enabled>
                        <label>ENABLED</label>
                        <visible-label>ENABLED</visible-label>
                </status>
                <relationship-type>USE</relationship-type>
                <update-timestamp>2015-06-12T11:20:37.312+0530</update-
timestamp>
        </relationship-instance>
</relationship-instances>
</instance>
```


Reconciliation module can be triggered by a timer and the time interval is configured in rest.endpoint.polling.interval property inside nfvd-recon-internal.properties

Note

For each VIM discovery, a separate NOM container must be created, and Capacity Management CAs must be deployed in each container.

For each 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.

Reconciliation also exposes a REST Endpoint to receive a trigger, the REST message is as follows:

- Method: `POST`
- Payload: `startrecon`
- Path-uri: `/recon/endpoint/trigger`
- Endpoint: `http://<IPAddress>:18989`

For example in Curl:

```
curl --data "startrecon"
http://localhost:18989/recon/endpoint/trigger
```

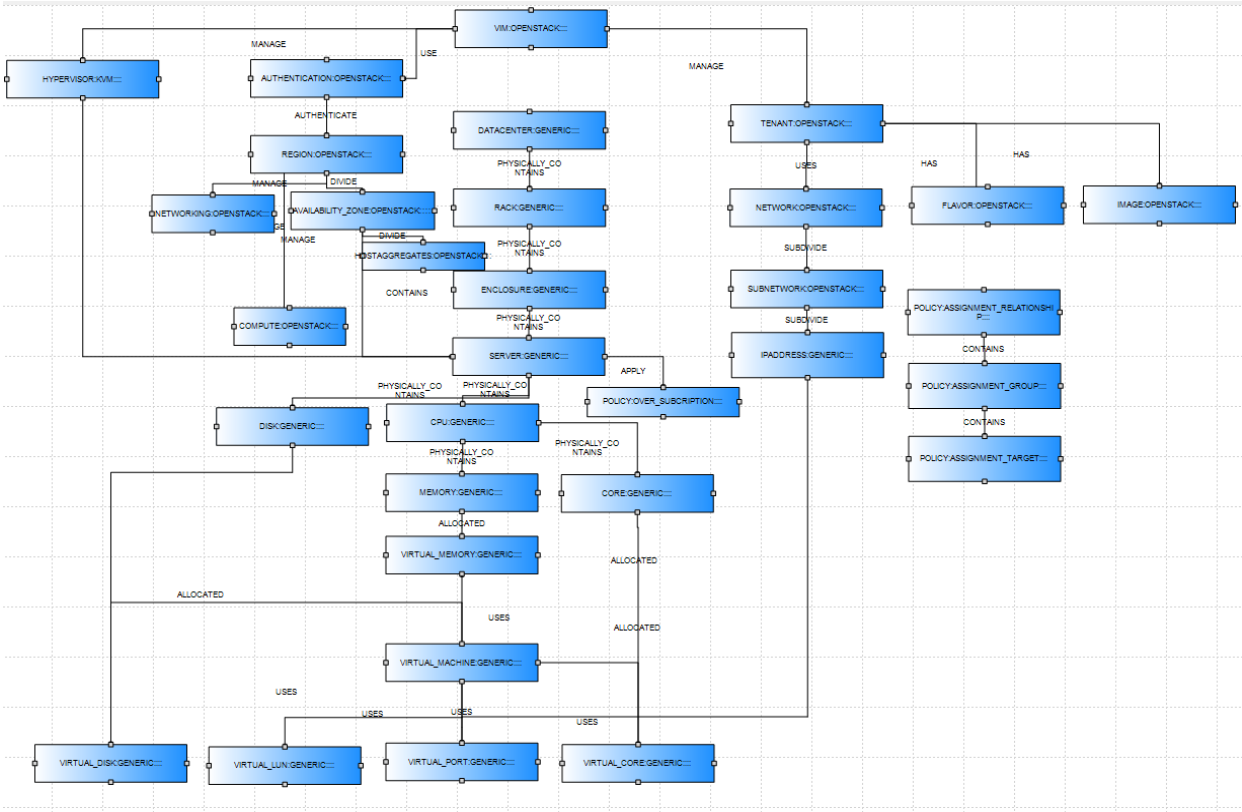


Figure 179 Discovered Artifacts

16.2 Exporting Topology and Metrics Information

16.2.1 Exporting Topology Information

Each notification for create, update and delete for artifacts are exported to topology file.

The topology files will be loaded in a folder that is configured in the `/var/opt/HP/nfvd/DataDirectory/Topology` folder.

The values in the file will be comma separated.

The format of the files are:
TOPOLOGY_<ARTIFACTFAMILY><ARTIFACTCATEGORY>_<HOST>_<TIMESTAMP>.csv.

Parameters in angular brackets which will be replaced are:

- ARTIFACT FAMILY : Name of the Artifact Family : eg : VNF,VIRTUAL_MACHINE etc
- ARTIFACTCATEGORY : Name of the Artifact Category eg : GENERIC
- HOST: Host machine were the file will be dumped
- TIMESTAMP: Timestamp of the file in UTC format.

The contents of the csv file will vary depending on the artifact Below is a sample of contents of a file

```
ACTIONS_ACTIONMODE,ACTIONS_APPROVED,ACTIONS_APPROVERCOMMENT,ACTIONS_CONDITION,ACTIONS_ENDTIMESTAMP
,ACTIONS_STARTIMESTAMP,ACTIONS_TIMEOFAPPROVAL,ACTIONS_USER,ACTIONS_OPERATION,CAPACITY.ALLOCATED_CO
RE,CAPACITY.ALLOCATED_DISK,CAPACITY.ALLOCATED_MEMORY,CAPACITY.AVAILABLE_CORE,CAPACITY.AVAILABLE_DI
SK,CAPACITY.AVAILABLE_MEMORY,CAPACITY.TOTAL_CORE,CAPACITY.TOTAL_DISK,CAPACITY.TOTAL_MEMORY,OPERATI
ON,TENANT.TENANTID,GENERAL.AssignmentPolicyID,GENERAL.Description,GENERAL.Name,GENERAL.Type,GENERA
L.Vendor,LAST_OPERATION.Operation,LAST_OPERATION.Result_code,LAST_OPERATION.Result_description,ORI
GIN_CREATION.RelationshipTreeID,ORIGIN_CREATION.ResourceTreeID,STATUS.LifeCycle_State,STATUS.LifeC
ycle_State_Date,STATUS.Operational_Status,STATUS.Operational_Status_Date,STATUS.Source,STATUS.addi
tionalText,artifactCategory,artifactFamily,artifactGroup,artifactId,artifactPhysical,artifactStatu
s,artifactStatusEnabled,artifactSubtype,artifactType,artifactVersion,artifactVisibleStatus,creatio
nTimestamp,lastUpdateTimestamp,templateId

null,null,null,null,null,null,null,null,null,0,0,0,0,0,0,0,0,0,create,c44b2d24-bf8c-4c31-ad5e-
b92514506747,,VNF Description,VNF_girish,VNF type,Vendor providing the VNF,,,,9c72dca5-a827-4f76-
bc40-49428a7f1d1c,d71dccba-360f-4a8c-ae74-2033ff112b79,,,,,,,GENERIC,VNF,,edbb7f5b-c752-42e1-9ca3-
30cfefda74e6,false,LOCKED,true,,,,LOCKED,2015-06-13T19:12:03.429+0530,2015-06-
13T19:12:03.429+0530,8fb6e22f-9505-4b22-8501-fb16bf4dc2e9 .
```

The CSV files contain the calculation for the capacity related artifacts which are:

- VNF
- VNF_COMPONENT
- VIRTUTAL_MACHINE
- SERVER
- RACK
- ENCLOSURE
- DATACENTER
- LOCATION
- AVAILABILITY_ZONE
- REGION

The attributes are:

1) TOTAL_CORE: Indicates the total core available for an Artifact.
2) ALLOCATED_CORE: Indicates the Allocated core.
3) AVAILABLE_CORE: Indicates Total_Core – ALLOCATED_CORE.
4) TOTAL_DISK: Indicates the total Disk available for an Artifact..
5) ALLOCATED_DISK: Indicates the Allocated disk.
6) AVAILABLE_DISK: Indicates Total_Disk – ALLOCATED_DISK.
7) TOTAL_MEMORY: Indicates the total memory available for an Artifact.
8) ALLOCATED_MEMORY: Indicates the Allocated memory.
9) AVAILABLE_MEMORY: Indicates Total_Memory – ALLOCATED_MEMORY.

For example: If a SERVER Artifact has 5 total cores and 1 are allocated to a vCore then: TOTAL_CORE=5,ALLOCATED_CORE=1 and AVAILABLE=4

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 . for example: export.VIRTUAL_MACHINE.GENERIC=false.

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

`export.<ARTIFACTFAMILY>.<ARTIFACTCATEGORY>=true`

For example : `export.MONITOR.GENERIC=true`

16.2.2 Exporting Metrics Information

Note : If a custom template is created in sitescope and kpi metrics needs to be exported to a file and metrics information should be displayed in GUI then user has to provide in the group description of sitescope template the value has %%groupDescription%%(counterName=<countnames>) . <countnames> : Names of the counter.
Eg: %%groupDescription%%(counterName=cpu_usage_average). If the counterName=<countnames> isn't provided in Sitescope and if there are multiple counters in the template then the order of the counters will not be maintained when it is exported to a file .

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 about how to configure Sitescope Instance to export KPI metrics.

The format of the file is `SiteScope_<HOST>_<TIMESTAMP>`.

The values in the angle brackets are:

- `HOST` = The hostname where the file will be generated
- `TIMESTAMP` = UTC time.

The columns of the files are:

FIELD	FIELD NAME	DESCRIPTION/COMMENT
1	COLLECTOR	Application collecting the data, which is always SiteScope
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	

Table 27 CSV File content

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_FAMILY,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_CATEGORY=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

Appendix A

Cloud System 8 Operations

A.1 Login

The user can enter the login credentials based on the permissions the user has, which ranges from a super-admin to tenant (project) user.

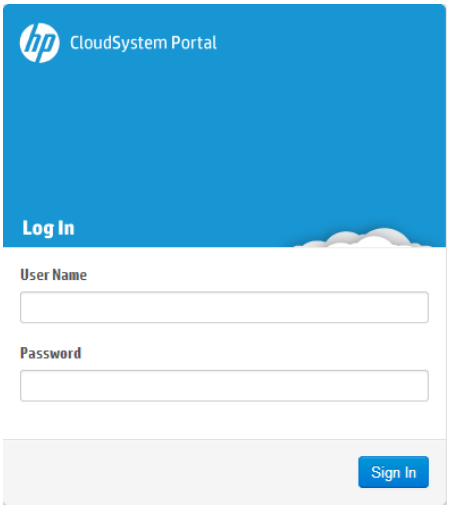


Figure 180 CloudSystem Login Portal

A.2 Overview

The following image shows the global status of the server, which includes allocated resources, active instances (usage summary), and the capacity of resources for an active tenant/user (limit summary).

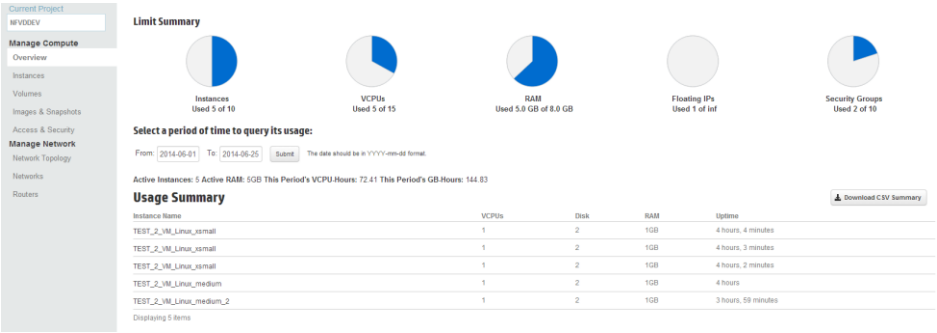


Figure 181 CloudSystem overview

A.3 Virtual machines (Instances)

This section provides information on the instances created on a server.

- Name—Name of the virtual machine.
- Image Name—Name of the image that was used to create the instance.
- IP Address—IP addresses of the virtual machines (external and internal).
- Size—Technical characteristics, flavor of the virtual machine (RAM, vCPU, disk).
- Status—Status of the VM (Active, Error, shut-down, and so on).

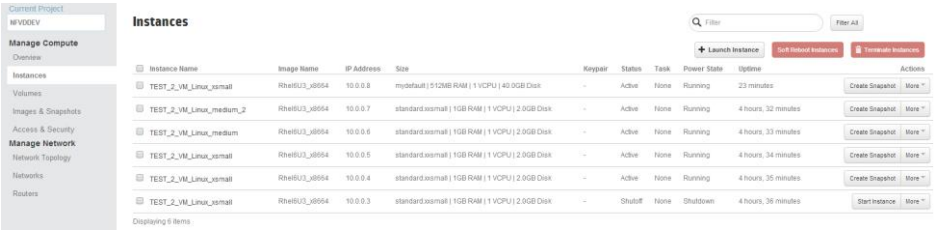


Figure 182 CloudSystem VM Instances

You can run the following VM operations from this section:

- Create a Virtual Machine
- Update a Virtual Machine
- Query for a Virtual Machine
- Delete (Terminate) a Virtual Machine
- Start / Stop a Virtual Machine

A.4 Images

This section provides information on the disk images allocated on a server.

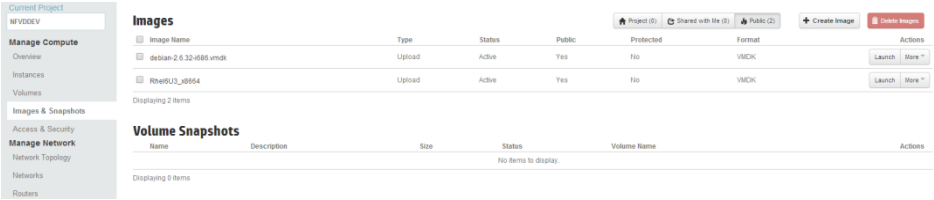


Figure 183 CloudSystem Images list

A.5 Networks

This section provides a list of available networks and subnets.

Figure 184 CloudSystem List of available networks and subnets

You can run the following operations from this screen:

- Create Network
- Create a Subnet from a Network
- Edit Network
- Edit Subnet
- Delete Network

Figure 185 CloudSystem Subnet

Appendix B

Relationships in NFV Director V3.0

Next pages show all the relationships in NFV Director version 3.0.

	Relationship	Relation type	Parent Artifact
1	ACTION:GENERIC:	HAS	ACTION_PARAMETER:GENERIC:
2	ACTION:GENERIC:	ASSOCIATED	NETWORK_SERVICE:GENERIC:
3	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_CORE:GENERIC:
4	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_DISK:GENERIC:
5	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_LUN:GENERIC:
6	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_MACHINE:GENERIC:
7	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_MACHINE:KVM:
8	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_MACHINE:VMWARE:
9	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_MEMORY:GENERIC:
10	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_PORT:BACKEND:
11	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_PORT:EXTERNAL:
12	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_PORT:FRONTEND:
13	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_PORT:GENERIC:
14	ACTION:GENERIC:	ASSOCIATED	VIRTUAL_PORT:INTERNAL:
15	ACTION:GENERIC:	ASSOCIATED	VNF:GENERIC:
16	ACTION:GENERIC:	ASSOCIATED	VNF_COMPONENT:GENERIC:
17	AUTHENTICATION:OPENSTACK:	AUTHENTICATE	REGION:OPENSTACK:
18	AVAILABILITY_ZONE:OPENSTACK:	DIVIDE	HOSTAGGREGATES:OPENSTACK:
19	AVAILABILITY_ZONE:OPENSTACK:	CONTAINS	SERVER:GENERIC:
20	CARD:GENERIC:	PHYSICALLY_CONTAINS	CARD:GENERIC:
21	CARD:GENERIC:	PHYSICALLY_CONTAINS	PORT:GENERIC:
22	CEILOMETER:OPENSTACK:	MEASURE	MONITOR:CUSTOM:
23	CINDER:OPENSTACK:	MEASURE	MONITOR:CUSTOM:
24	COMPUTE:OPENSTACK:	MEASURE	MONITOR:CUSTOM:
25	CONDITION:GENERIC:	TRIGGERS	ACTION:GENERIC:
26	CORE:GENERIC:	ALLOCATED	VIRTUAL_CORE:GENERIC:
27	CPU:GENERIC:	PHYSICALLY_CONTAINS	CORE:GENERIC:
28	CPU:GENERIC:	PHYSICALLY_CONTAINS	MEMORY:GENERIC:
29	CREDENTIALS:GENERIC:	USES	VNFM_API:GENERIC:
30	DATACENTER:GENERIC:	APPLY	POLICY:ENTITY_RANGE:
31	DATACENTER:GENERIC:	PHYSICALLY_CONTAINS	RACK:GENERIC:
32	DATACENTER:GENERIC:	PHYSICALLY_CONTAINS	SERVER:GENERIC:
33	DATACENTER:GENERIC:	CONTAINS	VIM:CS8:
34	DATACENTER:GENERIC:	CONTAINS	VIM:GENERIC:
35	DATACENTER:GENERIC:	CONTAINS	VIM:HELION:
36	DATACENTER:GENERIC:	CONTAINS	VIM:ICEHOUSE:
37	DATACENTER:GENERIC:	CONTAINS	VIM:OPENSTACK:
38	DATACENTER:GENERIC:	CONTAINS	VNF:GENERIC:
39	DATACENTER:GENERIC:	CONTAINS	ZONE:BACKEND:
40	DATACENTER:GENERIC:	CONTAINS	ZONE:CE:
41	DATACENTER:GENERIC:	CONTAINS	ZONE:EXTERNAL:
42	DATACENTER:GENERIC:	CONTAINS	ZONE:FIREWALL:
43	DATACENTER:GENERIC:	CONTAINS	ZONE:FRONTEND:
44	DATACENTER:GENERIC:	CONTAINS	ZONE:INTERNAL:
45	DESCRIPTOR:GENERIC:	CONTAINS	TEMPLATE:GENERIC:

46	DISK:GENERIC:	ALLOCATED	VIRTUAL_DISK:GENERIC:
47	ENCLOSURE:GENERIC:	PHYSICALLY_CONTAINS	PORT:GENERIC:
48	ENCLOSURE:GENERIC:	PHYSICALLY_CONTAINS	SERVER:GENERIC:
49	ENDPOINT:GENERIC:	CONNECTED	ENDPOINT:GENERIC:
50	ENDPOINT:GENERIC:	IS_COMPOSED_BY	INTERFACE:GENERIC:
51	ENDPOINT:GENERIC:	IMPLEMENTS	LINK:GENERIC:
52	ENDPOINT:GENERIC:	IS_COMPOSED_BY	PORT:GENERIC:
53	ENDPOINT:GENERIC:	IS_COMPOSED_BY	VIRTUAL_PORT:BACKEND:
54	ENDPOINT:GENERIC:	IS_COMPOSED_BY	VIRTUAL_PORT:EXTERNAL:
55	ENDPOINT:GENERIC:	IS_COMPOSED_BY	VIRTUAL_PORT:FRONTEND:
56	ENDPOINT:GENERIC:	IS_COMPOSED_BY	VIRTUAL_PORT:GENERIC:
57	ENDPOINT:GENERIC:	IS_COMPOSED_BY	VIRTUAL_PORT:INTERNAL:
58	ENDPOINT:GENERIC:	CONNECTED	ZONE:BACKEND:
59	ENDPOINT:GENERIC:	CONNECTED	ZONE:EXTERNAL:
60	ENDPOINT:GENERIC:	CONNECTED	ZONE:FRONTEND:
61	ENDPOINT:GENERIC:	CONNECTED	ZONE:INTERNAL:
62	EXECUTION_TASK:GENERIC:	EXECUTE	EXECUTION_TASK:GENERIC:
63	EXECUTION_TASK:GENERIC:	EXECUTE	EXECUTION_TASK_LIST:GENERIC:
64	EXECUTION_TASK_LIST:GENERIC:	EXECUTE	EXECUTION_TASK:GENERIC:
65	EXECUTION_TASK_LIST:GENERIC:	EXECUTE	EXECUTION_TASK_LIST:GENERIC:
66	FG_ENDPOINT:BIDIRECTIONAL:OPENSTACK	IMPLEMENTS	NETWORK:OPENSTACK:
67	FG_ENDPOINT:BIDIRECTIONAL:OPENSTACK	IMPLEMENTS	NETWORK:OPENSTACK:EXTERNAL
68	FG_ENDPOINT:BIDIRECTIONAL:OPENSTACK	IMPLEMENTS	NETWORK:OPENSTACK:PRIVATE
69	FG_ENDPOINT:BIDIRECTIONAL:OPENSTACK	IMPLEMENTS	NETWORK:OPENSTACK:PROVIDER
70	FG_ENDPOINT:BIDIRECTIONAL:OPENSTACK	CONNECT_THROUGH	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK
71	FG_ENDPOINT:BIDIRECTIONAL:OPENSTACK	CONNECTED	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK
72	FORWARDING_GRAPH:GENERIC:	BIDIRECTIONAL	FG_ENDPOINT:BIDIRECTIONAL:OPENSTACK
73	FORWARDING_GRAPH:GENERIC:	APPLY	FORWARDING_GRAPH:GENERIC:
74	GLANCE:OPENSTACK:	MEASURE	MONITOR:CUSTOM:
75	GROUP:GENERIC:	GENERIC	LOCATION:GENERIC:
76	GROUP:GENERIC:	GENERIC	NETWORK:BACKEND:
77	GROUP:GENERIC:	GENERIC	NETWORK:EXTERNAL:
78	GROUP:GENERIC:	GENERIC	NETWORK:FRONTEND:
79	GROUP:GENERIC:	GENERIC	NETWORK:GENERIC:
80	GROUP:GENERIC:	GENERIC	NETWORK:INTERNAL:
81	GROUP:GENERIC:	GENERIC	NETWORK:OPENSTACK:
82	GROUP:GENERIC:	GENERIC	NETWORK_SERVICE:GENERIC:
83	GROUP:GENERIC:	GENERIC	VNF:GENERIC:
84	HEAT:OPENSTACK:	MEASURE	MONITOR:CUSTOM:
85	HOSTAGGREGATES:OPENSTACK:	CONTAINS	SERVER:GENERIC:
86	HYPERVISOR:GENERIC:	MANAGE	SERVER:GENERIC:
87	HYPERVISOR:KVM:	MANAGE	SERVER:GENERIC:
88	HYPERVISOR:VCENTER:	MANAGE	SERVER:GENERIC:
89	HYPERVISOR:VMWARE:	MANAGE	SERVER:GENERIC:
90	IMAGE_STORAGE:OPENSTACK:	HAS	IMAGE:OPENSTACK:
91	IPADDRESS:GENERIC:	ALLOCATED	INTERFACE:GENERIC:
92	IPADDRESS:GENERIC:	ALLOCATED	PORT:GENERIC:
93	IPADDRESS:GENERIC:	ALLOCATED	VIRTUAL_PORT:BACKEND:
94	IPADDRESS:GENERIC:	ALLOCATED	VIRTUAL_PORT:EXTERNAL:
95	IPADDRESS:GENERIC:	ALLOCATED	VIRTUAL_PORT:FRONTEND:
96	IPADDRESS:GENERIC:	ALLOCATED	VIRTUAL_PORT:GENERIC:
97	IPADDRESS:GENERIC:	ALLOCATED	VIRTUAL_PORT:INTERNAL:
98	IPADDRESS_POOL:GENERIC:	ASSIGNED	INTERFACE:GENERIC:
99	IPADDRESS_POOL:GENERIC:	IS_COMPOSED_BY	INTERFACE:GENERIC:
100	KEYSTONE:OPENSTACK:	MEASURE	MONITOR:CUSTOM:
101	LINK:GENERIC:	CONNECTED	ENDPOINT:GENERIC:
102	LINK:GENERIC:	CONNECTED	NETWORK:BACKEND:
103	LINK:GENERIC:	CONNECTED	NETWORK:EXTERNAL:
104	LINK:GENERIC:	CONNECTED	NETWORK:FRONTEND:
105	LINK:GENERIC:	CONNECTED	NETWORK:GENERIC:

106	LINK:GENERIC:	CONNECTED	NETWORK:INTERNAL:
107	LINK:GENERIC:	CONNECTED	NETWORK:OPENSTACK:
108	LOCATION:GENERIC:	IS_LOCATED	DATACENTER:GENERIC:
109	LOCATION:GENERIC:	APPLY	POLICY:ENTITY_RANGE:
110	LOCATION:GENERIC:	GENERIC	SERVER:GENERIC:
111	LUN:GENERIC:	ALLOCATED	VIRTUAL_DISK:GENERIC:
112	LUN:GENERIC:	ALLOCATED	VIRTUAL_LUN:GENERIC:
113	MEMORY:GENERIC:	ALLOCATED	VIRTUAL_MEMORY:GENERIC:
114	MONITOR:CUSTOM:	TRANSGRESS	CONDITION:GENERIC:
115	MONITOR:CUSTOM:	USES	MONITOR_ARGUMENTS:ARGUMENTS:
116	MONITOR:CUSTOM:	USES	MONITOR_ARGUMENTS:CYPHER:
117	MONITOR:GENERIC:	TRANSGRESS	CONDITION:GENERIC:
118	MONITOR:GENERIC:	USES	MONITOR_ARGUMENTS:CYPHER:
119	MONITOR:SELF:	TRANSGRESS	CONDITION:GENERIC:
120	MONITOR:SELF:	USES	MONITOR_ARGUMENTS:CYPHER:
121	NE:FIREWALL:	PHYSICALLY_CONTAINS	CARD:GENERIC:
122	NE:LOAD_BALANCER:	PHYSICALLY_CONTAINS	CARD:GENERIC:
123	NE:ROUTER:	PHYSICALLY_CONTAINS	CARD:GENERIC:
124	NE:SWITCH:	PHYSICALLY_CONTAINS	CARD:GENERIC:
125	NETWORK:BACKEND:	CONNECTED	ENDPOINT:GENERIC:
126	NETWORK:BACKEND:	SUBDIVIDE	SUBNETWORK:GENERIC:
127	NETWORK:EXTERNAL:	CONNECTED	ENDPOINT:GENERIC:
128	NETWORK:EXTERNAL:	SUBDIVIDE	SUBNETWORK:GENERIC:
129	NETWORK:FRONTEND:	CONNECTED	ENDPOINT:GENERIC:
130	NETWORK:FRONTEND:	SUBDIVIDE	SUBNETWORK:GENERIC:
131	NETWORK:GENERIC:	CONNECTED	ENDPOINT:GENERIC:
132	NETWORK:GENERIC:	SUBDIVIDE	SUBNETWORK:GENERIC:
133	NETWORK:INTERNAL:	CONNECTED	ENDPOINT:GENERIC:
134	NETWORK:INTERNAL:	SUBDIVIDE	SUBNETWORK:GENERIC:
135	NETWORK:OPENSTACK:	CONNECTED	ENDPOINT:GENERIC:
136	NETWORK:OPENSTACK:	SUBDIVIDE	SUBNETWORK:OPENSTACK:
137	NETWORK:OPENSTACK:EXTERNAL	CONNECTED	ENDPOINT:GENERIC:
138	NETWORK:OPENSTACK:EXTERNAL	SUBDIVIDE	SUBNETWORK:OPENSTACK:
139	NETWORK:OPENSTACK:PRIVATE	CONNECTED	ENDPOINT:GENERIC:
140	NETWORK:OPENSTACK:PRIVATE	SUBDIVIDE	SUBNETWORK:OPENSTACK:
141	NETWORK:OPENSTACK:PROVIDER	CONNECTED	ENDPOINT:GENERIC:
142	NETWORK:OPENSTACK:PROVIDER	SUBDIVIDE	SUBNETWORK:OPENSTACK:
143	NETWORK_SERVICE:GENERIC:	ASSOCIATED	ACTION:GENERIC:
144	NETWORK_SERVICE:GENERIC:	INCLUDE	FORWARDING_GRAPH:GENERIC:
145	NETWORK_SERVICE:GENERIC:	USES	LINK:GENERIC:
146	NETWORK_SERVICE:GENERIC:	MEASURE	MONITOR:GENERIC:
147	NETWORK_SERVICE:GENERIC:	INCLUDE	NETWORK_SERVICE:GENERIC:
148	NETWORK_SERVICE:GENERIC:	APPLY	POLICY:POSTPRE_PROCESSING:
149	NETWORK_SERVICE:GENERIC:	STATUS_CHANGED_BY	PROPAGATION_RULE:GENERIC:
150	NETWORK_SERVICE:GENERIC:	INCLUDE	VIRTUAL_LINK:GENERIC:
151	NETWORK_SERVICE:GENERIC:	INCLUDE	VNF:GENERIC:
152	NETWORKING:OPENSTACK:	MEASURE	MONITOR:CUSTOM:
153	PHYSICAL_MACHINE:GENERIC:	MEASURE	MONITOR:CUSTOM:
154	PHYSICAL_MACHINE:GENERIC:	MEASURE	MONITOR:GENERIC:
155	PHYSICAL_MACHINE:GENERIC:	MEASURE	MONITOR:SELF:
156	PHYSICAL_MACHINE:GENERIC:	APPLY	POLICY:AFFINITY:
157	PHYSICAL_MACHINE:GENERIC:	APPLY	POLICY:ANTI_AFFINITY:
158	PHYSICAL_MACHINE:GENERIC:	APPLY	POLICY:ENTITY_RANGE:
159	PHYSICAL_MACHINE:GENERIC:	APPLY	POLICY:OVER_SUBCRIPTION:
160	PHYSICAL_MACHINE:GENERIC:	APPLY	POLICY:POSTPRE_PROCESSING:
161	PHYSICAL_MACHINE:GENERIC:	APPLY	POLICY:VALUE_GENERATION:
162	PHYSICAL_MACHINE:GENERIC:	APPLY	POLICY:VALUE_VALIDATION:
163	PHYSICAL_MACHINE:GENERIC:	STATUS_CHANGED_BY	PROPAGATION_RULE:GENERIC:
164	PHYSICAL_MACHINE:GENERIC:	USES	VIRTUAL_CORE:GENERIC:
165	PHYSICAL_MACHINE:GENERIC:	USES	VIRTUAL_DISK:GENERIC:

166	PHYSICAL_MACHINE:GENERIC:	USES	VIRTUAL_LUN:GENERIC:
167	PHYSICAL_MACHINE:GENERIC:	USES	VIRTUAL_MEMORY:GENERIC:
168	PHYSICAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:BACKEND:
169	PHYSICAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:EXTERNAL:
170	PHYSICAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:FRONTEND:
171	PHYSICAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:GENERIC:
172	PHYSICAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:INTERNAL:
173	POLICY:AFFINITY:	CONTAINS	POLICY:ASSIGNMENT_GROUP:
174	POLICY:AFFINITY:	APPLY	VIRTUAL_MACHINE:GENERIC:
175	POLICY:AFFINITY:	APPLY	VIRTUAL_MACHINE:KVM:
176	POLICY:AFFINITY:	APPLY	VIRTUAL_MACHINE:VMWARE:
177	POLICY:ANTI_AFFINITY:	CONTAINS	POLICY:ASSIGNMENT_GROUP:
178	POLICY:ANTI_AFFINITY:	APPLY	VIRTUAL_MACHINE:GENERIC:
179	POLICY:ANTI_AFFINITY:	APPLY	VIRTUAL_MACHINE:KVM:
180	POLICY:ANTI_AFFINITY:	APPLY	VIRTUAL_MACHINE:VMWARE:
181	POLICY:ASSIGNMENT_GROUP:	CONTAINS	POLICY:ASSIGNMENT_TARGET:
182	POLICY:ASSIGNMENT_RELATIONSHIP:	CONTAINS	POLICY:ASSIGNMENT_GROUP:
183	POLICY:ENTITY_RANGE:	OVER	DATACENTER:GENERIC:
184	POLICY:ENTITY_RANGE:	OVER	IPADDRESS:GENERIC:
185	POLICY:ENTITY_RANGE:	OVER	RACK:GENERIC:
186	POLICY:ENTITY_RANGE:	OVER	SERVER:GENERIC:
187	POLICY:ENTITY_RANGE:	OVER	VIM:GENERIC:
188	POLICY:ENTITY_RANGE:	OVER	VIM:HELION:
189	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_CORE:GENERIC:
190	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_DISK:GENERIC:
191	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_MACHINE:GENERIC:
192	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_MACHINE:KVM:
193	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_MACHINE:VMWARE:
194	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_MEMORY:GENERIC:
195	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_PORT:BACKEND:
196	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_PORT:EXTERNAL:
197	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_PORT:FRONTEND:
198	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_PORT:GENERIC:
199	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_PORT:INTERNAL:
200	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_PORT:PRIVATE:
201	POLICY:ENTITY_RANGE:	OVER	VIRTUAL_PORT:PROVIDER:
202	POLICY:ENTITY_RANGE:	OVER	VNF:GENERIC:
203	POLICY:ENTITY_RANGE:	OVER	VNF_COMPONENT:GENERIC:
204	POLICY:OVER_SUBCRIPTION:	APPLY	CORE:GENERIC:
205	POLICY:OVER_SUBCRIPTION:	APPLY	DISK:GENERIC:
206	POLICY:OVER_SUBCRIPTION:	APPLY	MEMORY:GENERIC:
207	POLICY:OVER_SUBCRIPTION:	APPLY	VIRTUAL_CORE:GENERIC:
208	POLICY:PROCESSING_TREE:	EXECUTE	POLICY:PROCESSING_ACTION:
209	PORT:GENERIC:	ALLOCATED	INTERFACE:GENERIC:
210	PORT:GENERIC:	CONFIGURED	INTERFACE:GENERIC:
211	PORT:GENERIC:	ALLOCATED	VIRTUAL_PORT:BACKEND:
212	PORT:GENERIC:	ALLOCATED	VIRTUAL_PORT:EXTERNAL:
213	PORT:GENERIC:	ALLOCATED	VIRTUAL_PORT:FRONTEND:
214	PORT:GENERIC:	ALLOCATED	VIRTUAL_PORT:GENERIC:
215	PORT:GENERIC:	ALLOCATED	VIRTUAL_PORT:INTERNAL:
216	QUOTA:GENERIC:	CONTAINS	QUOTA:VIRTUAL_CORE:
217	QUOTA:GENERIC:	CONTAINS	QUOTA:VIRTUAL_DISK:
218	QUOTA:GENERIC:	CONTAINS	QUOTA:VIRTUAL_LUN:
219	QUOTA:GENERIC:	CONTAINS	QUOTA:VIRTUAL_MEMORY:
220	QUOTA:VIRTUAL_CORE:	CONSUMED	VIRTUAL_CORE:GENERIC:
221	QUOTA:VIRTUAL_DISK:	CONSUMED	VIRTUAL_DISK:GENERIC:
222	QUOTA:VIRTUAL_MEMORY:	CONSUMED	VIRTUAL_MEMORY:GENERIC:
223	RACK:GENERIC:	PHYSICALLY_CONTAINS	ENCLOSURE:GENERIC:
224	RACK:GENERIC:	PHYSICALLY_CONTAINS	NE:FIREWALL:
225	RACK:GENERIC:	PHYSICALLY_CONTAINS	NE:LOAD_BALANCER:

226	RACK:GENERIC:	PHYSICALLY_CONTAINS	NE:ROUTER:
227	RACK:GENERIC:	PHYSICALLY_CONTAINS	NE:SWITCH:
228	RACK:GENERIC:	PHYSICALLY_CONTAINS	SERVER:GENERIC:
229	REGION:OPENSTACK:	DIVIDE	AVAILABILITY_ZONE:OPENSTACK:
230	REGION:OPENSTACK:	MANAGE	COMPUTE:OPENSTACK:
231	REGION:OPENSTACK:	MANAGE	IMAGE_STORAGE:OPENSTACK:
232	REGION:OPENSTACK:	MANAGE	NETWORKING:OPENSTACK:
233	REGION:OPENSTACK:	MANAGE	STORAGE:OPENSTACK:
234	REPO:IMAGE:LOCAL	HAS	IMAGE:GENERIC:
235	RESOURCE_POOL:GENERIC:	PROVIDES	BANDWIDTH:GENERIC:
236	RESOURCE_POOL:GENERIC:	CONTAINS	DATACENTER:GENERIC:
237	RESOURCE_POOL:GENERIC:	PROVIDES	IPADDRESS_POOL:GENERIC:
238	RESOURCE_POOL:GENERIC:	PROVIDES	VIRTUAL_CORE:GENERIC:
239	RESOURCE_POOL:GENERIC:	PROVIDES	VIRTUAL_DISK:GENERIC:
240	RESOURCE_POOL:GENERIC:	PROVIDES	VIRTUAL_MEMORY:GENERIC:
241	SECURITY_GROUP:OPENSTACK:	ATTACHED	SECURITY_GROUP_RULE:OPENSTACK:
242	SERVER:GENERIC:	PHYSICALLY_CONTAINS	CARD:GENERIC:
243	SERVER:GENERIC:	PHYSICALLY_CONTAINS	CPU:GENERIC:
244	SERVER:GENERIC:	PHYSICALLY_CONTAINS	DISK:GENERIC:
245	SERVER:GENERIC:	CONNECTED	LUN:GENERIC:
246	SERVER:GENERIC:	MEASURE	MONITOR:GENERIC:
247	SERVER:GENERIC:	APPLY	POLICY:OVER_SUBCRIPTION:
248	SERVER:GENERIC:	STATUS_CHANGED_BY	PROPAGATION_RULE:GENERIC:
249	SUBNETWORK:GENERIC:	SUBDIVIDE	IPADDRESS:GENERIC:
250	SUBNETWORK:GENERIC:	APPLY	POLICY:ENTITY_RANGE:
251	SUBNETWORK:OPENSTACK:	SUBDIVIDE	IPADDRESS:FLOATING:
252	SUBNETWORK:OPENSTACK:	SUBDIVIDE	IPADDRESS:GENERIC:
253	SUBNETWORK:OPENSTACK:	APPLY	POLICY:ENTITY_RANGE:
254	SUBNETWORK:OPENSTACK:	SUBDIVIDE	VIRTUAL_IP:GENERIC:
255	SWIFT:OPENSTACK:	MEASURE	MONITOR:CUSTOM:
256	TASK_DEFINITION:GENERIC:	EXECUTE	TASK_DEFINITION:GENERIC:
257	TASK_DEFINITION:GENERIC:	EXECUTE	TASK_LIST_DEFINITION:GENERIC:
258	TASK_LIST_DEFINITION:GENERIC:	EXECUTE	TASK_DEFINITION:GENERIC:
259	TASK_LIST_DEFINITION:GENERIC:	EXECUTE	TASK_LIST_DEFINITION:GENERIC:
260	TEMPLATE:GENERIC:	USES	POLICY:ASSIGNMENT_RELATIONSHIP:
261	TENANT:GENERIC:	DEDICATED	BANDWIDTH:GENERIC:
262	TENANT:GENERIC:	SHARED	BANDWIDTH:GENERIC:
263	TENANT:GENERIC:	DEDICATED	IPADDRESS:GENERIC:
264	TENANT:GENERIC:	DEDICATED	NETWORK:BACKEND:
265	TENANT:GENERIC:	SHARED	NETWORK:BACKEND:
266	TENANT:GENERIC:	DEDICATED	NETWORK:EXTERNAL:
267	TENANT:GENERIC:	SHARED	NETWORK:EXTERNAL:
268	TENANT:GENERIC:	DEDICATED	NETWORK:FRONTEND:
269	TENANT:GENERIC:	SHARED	NETWORK:FRONTEND:
270	TENANT:GENERIC:	DEDICATED	NETWORK:GENERIC:
271	TENANT:GENERIC:	SHARED	NETWORK:GENERIC:
272	TENANT:GENERIC:	DEDICATED	NETWORK:INTERNAL:
273	TENANT:GENERIC:	SHARED	NETWORK:INTERNAL:
274	TENANT:GENERIC:	DEDICATED	NETWORK:OPENSTACK:
275	TENANT:GENERIC:	SHARED	NETWORK:OPENSTACK:
276	TENANT:GENERIC:	APPLY	QUOTA:GENERIC:
277	TENANT:GENERIC:	HAS	REPO:IMAGE:LOCAL
278	TENANT:GENERIC:	DEDICATED	RESOURCE_POOL:GENERIC:
279	TENANT:GENERIC:	SHARED	RESOURCE_POOL:GENERIC:
280	TENANT:GENERIC:	DEDICATED	VIRTUAL_CORE:GENERIC:
281	TENANT:GENERIC:	SHARED	VIRTUAL_CORE:GENERIC:
282	TENANT:GENERIC:	DEDICATED	VIRTUAL_DISK:GENERIC:
283	TENANT:GENERIC:	SHARED	VIRTUAL_DISK:GENERIC:
284	TENANT:GENERIC:	DEDICATED	VIRTUAL_MEMORY:GENERIC:
285	TENANT:GENERIC:	SHARED	VIRTUAL_MEMORY:GENERIC:

286	TENANT:GENERIC:	CONTAINS	VNF:GENERIC:
287	TENANT:OPENSTACK:	HAS	FLAVOR:OPENSTACK:
288	TENANT:OPENSTACK:	HAS	IMAGE:OPENSTACK:
289	TENANT:OPENSTACK:	USES	NETWORK:OPENSTACK:
290	TENANT:OPENSTACK:	USES	NETWORK:OPENSTACK:EXTERNAL
291	TENANT:OPENSTACK:	USES	NETWORK:OPENSTACK:PRIVATE
292	TENANT:OPENSTACK:	USES	NETWORK:OPENSTACK:PROVIDER
293	TENANT:OPENSTACK:	CONTAINS	SECURITY_GROUP:OPENSTACK:
294	TENANT:OPENSTACK:	CONTAINS	VIRTUAL_IP:GENERIC:
295	VIM:CS8:	USE	AUTHENTICATION:OPENSTACK:
296	VIM:CS8:	MANAGE	HYPERVISOR:GENERIC:
297	VIM:CS8:	MANAGE	HYPERVISOR:KVM:
298	VIM:CS8:	MANAGE	HYPERVISOR:VCENTER:
299	VIM:CS8:	MANAGE	HYPERVISOR:VMWARE:
300	VIM:CS8:	MANAGE	TENANT:OPENSTACK:
301	VIM:GENERIC:	USE	AUTHENTICATION:OPENSTACK:
302	VIM:GENERIC:	MANAGE	HYPERVISOR:GENERIC:
303	VIM:GENERIC:	MANAGE	HYPERVISOR:KVM:
304	VIM:GENERIC:	MANAGE	HYPERVISOR:VCENTER:
305	VIM:GENERIC:	MANAGE	HYPERVISOR:VMWARE:
306	VIM:GENERIC:	MANAGE	TENANT:OPENSTACK:
307	VIM:HELION:	USE	AUTHENTICATION:OPENSTACK:
308	VIM:HELION:	MANAGE	HYPERVISOR:GENERIC:
309	VIM:HELION:	MANAGE	HYPERVISOR:KVM:
310	VIM:HELION:	MANAGE	HYPERVISOR:VCENTER:
311	VIM:HELION:	MANAGE	HYPERVISOR:VMWARE:
312	VIM:HELION:	MANAGE	TENANT:OPENSTACK:
313	VIM:ICEHOUSE:	USE	AUTHENTICATION:OPENSTACK:
314	VIM:ICEHOUSE:	MANAGE	HYPERVISOR:GENERIC:
315	VIM:ICEHOUSE:	MANAGE	HYPERVISOR:KVM:
316	VIM:ICEHOUSE:	MANAGE	HYPERVISOR:VCENTER:
317	VIM:ICEHOUSE:	MANAGE	HYPERVISOR:VMWARE:
318	VIM:ICEHOUSE:	MANAGE	TENANT:OPENSTACK:
319	VIM:OPENSTACK:	USE	AUTHENTICATION:OPENSTACK:
320	VIM:OPENSTACK:	MANAGE	HYPERVISOR:GENERIC:
321	VIM:OPENSTACK:	MANAGE	HYPERVISOR:KVM:
322	VIM:OPENSTACK:	MANAGE	HYPERVISOR:VCENTER:
323	VIM:OPENSTACK:	MANAGE	HYPERVISOR:VMWARE:
324	VIM:OPENSTACK:	MANAGE	TENANT:OPENSTACK:
325	VIRTUAL_CORE:GENERIC:	ASSOCIATED	ACTION:GENERIC:
326	VIRTUAL_CORE:GENERIC:	APPLY	POLICY:ENTITY_SCALE:
327	VIRTUAL_DATACENTER:GENERIC:	ALLOCATE	PHYSICAL_MACHINE:GENERIC:
328	VIRTUAL_DATACENTER:GENERIC:	ALLOCATE	VIRTUAL_MACHINE:GENERIC:
329	VIRTUAL_DATACENTER:GENERIC:	ALLOCATE	VIRTUAL_MACHINE:KVM:
330	VIRTUAL_DATACENTER:GENERIC:	ALLOCATE	VIRTUAL_MACHINE:VMWARE:
331	VIRTUAL_DATACENTER:GENERIC:	CONTAINS	VNF:GENERIC:
332	VIRTUAL_DATACENTER:GENERIC:	CONTAINS	ZONE:BACKEND:
333	VIRTUAL_DATACENTER:GENERIC:	CONTAINS	ZONE:CE:
334	VIRTUAL_DATACENTER:GENERIC:	CONTAINS	ZONE:EXTERNAL:
335	VIRTUAL_DATACENTER:GENERIC:	CONTAINS	ZONE:FIREWALL:
336	VIRTUAL_DATACENTER:GENERIC:	CONTAINS	ZONE:FRONTEND:
337	VIRTUAL_DATACENTER:GENERIC:	CONTAINS	ZONE:INTERNAL:
338	VIRTUAL_DISK:GENERIC:	ASSOCIATED	ACTION:GENERIC:
339	VIRTUAL_DISK:GENERIC:	APPLY	POLICY:ENTITY_SCALE:
340	VIRTUAL_IP:GENERIC:	ASSIGNED	VIRTUAL_MACHINE:GENERIC:
341	VIRTUAL_IP:GENERIC:	ASSIGNED	VIRTUAL_PORT:GENERIC:
342	VIRTUAL_LINK:GENERIC:	APPLY	VIRTUAL_LINK:GENERIC:
343	VIRTUAL_LINK:GENERIC:	BIDIRECTIONAL	VL_ENDPOINT:BIDIRECTIONAL:OPENSTACK
344	VIRTUAL_LUN:GENERIC:	ASSOCIATED	ACTION:GENERIC:
345	VIRTUAL_LUN:GENERIC:	APPLY	POLICY:ENTITY_SCALE:

346	VIRTUAL_MACHINE:GENERIC:	ASSOCIATED	ACTION:GENERIC:
347	VIRTUAL_MACHINE:GENERIC:	MEASURE	MONITOR:CUSTOM:
348	VIRTUAL_MACHINE:GENERIC:	MEASURE	MONITOR:GENERIC:
349	VIRTUAL_MACHINE:GENERIC:	MEASURE	MONITOR:SELF:
350	VIRTUAL_MACHINE:GENERIC:	APPLY	POLICY:AFFINITY:
351	VIRTUAL_MACHINE:GENERIC:	APPLY	POLICY:ANTI_AFFINITY:
352	VIRTUAL_MACHINE:GENERIC:	APPLY	POLICY:ENTITY_ASSIGN:
353	VIRTUAL_MACHINE:GENERIC:	APPLY	POLICY:ENTITY_RANGE:
354	VIRTUAL_MACHINE:GENERIC:	APPLY	POLICY:OVER_SUBCRIPTION:
355	VIRTUAL_MACHINE:GENERIC:	APPLY	POLICY:POSTPRE_PROCESSING:
356	VIRTUAL_MACHINE:GENERIC:	APPLY	POLICY:VALUE_GENERATION:
357	VIRTUAL_MACHINE:GENERIC:	APPLY	POLICY:VALUE_VALIDATION:
358	VIRTUAL_MACHINE:GENERIC:	STATUS_CHANGED_BY	PROPAGATION_RULE:GENERIC:
359	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_CORE:GENERIC:
360	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_DISK:GENERIC:
361	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_LUN:GENERIC:
362	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_MEMORY:GENERIC:
363	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:BACKEND:
364	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:EXTERNAL:
365	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:FRONTEND:
366	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:GENERIC:
367	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:INTERNAL:
368	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:MANAGEMENT:
369	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:PRIVATE:
370	VIRTUAL_MACHINE:GENERIC:	USES	VIRTUAL_PORT:PROVIDER:
371	VIRTUAL_MACHINE:KVM:	ASSOCIATED	ACTION:GENERIC:
372	VIRTUAL_MACHINE:KVM:	MEASURE	MONITOR:GENERIC:
373	VIRTUAL_MACHINE:KVM:	APPLY	POLICY:AFFINITY:
374	VIRTUAL_MACHINE:KVM:	APPLY	POLICY:ANTI_AFFINITY:
375	VIRTUAL_MACHINE:KVM:	APPLY	POLICY:ENTITY_ASSIGN:
376	VIRTUAL_MACHINE:KVM:	APPLY	POLICY:ENTITY_RANGE:
377	VIRTUAL_MACHINE:KVM:	APPLY	POLICY:OVER_SUBCRIPTION:
378	VIRTUAL_MACHINE:KVM:	APPLY	POLICY:POSTPRE_PROCESSING:
379	VIRTUAL_MACHINE:KVM:	APPLY	POLICY:VALUE_GENERATION:
380	VIRTUAL_MACHINE:KVM:	APPLY	POLICY:VALUE_VALIDATION:
381	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_CORE:GENERIC:
382	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_DISK:GENERIC:
383	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_LUN:GENERIC:
384	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_MEMORY:GENERIC:
385	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_PORT:BACKEND:
386	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_PORT:EXTERNAL:
387	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_PORT:FRONTEND:
388	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_PORT:GENERIC:
389	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_PORT:INTERNAL:
390	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_PORT:PRIVATE:
391	VIRTUAL_MACHINE:KVM:	USES	VIRTUAL_PORT:PROVIDER:
392	VIRTUAL_MACHINE:VMWARE:	ASSOCIATED	ACTION:GENERIC:
393	VIRTUAL_MACHINE:VMWARE:	MEASURE	MONITOR:GENERIC:
394	VIRTUAL_MACHINE:VMWARE:	APPLY	POLICY:AFFINITY:
395	VIRTUAL_MACHINE:VMWARE:	APPLY	POLICY:ANTI_AFFINITY:
396	VIRTUAL_MACHINE:VMWARE:	APPLY	POLICY:ENTITY_ASSIGN:
397	VIRTUAL_MACHINE:VMWARE:	APPLY	POLICY:ENTITY_RANGE:
398	VIRTUAL_MACHINE:VMWARE:	APPLY	POLICY:OVER_SUBCRIPTION:
399	VIRTUAL_MACHINE:VMWARE:	APPLY	POLICY:POSTPRE_PROCESSING:
400	VIRTUAL_MACHINE:VMWARE:	APPLY	POLICY:VALUE_GENERATION:
401	VIRTUAL_MACHINE:VMWARE:	APPLY	POLICY:VALUE_VALIDATION:
402	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_CORE:GENERIC:
403	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_DISK:GENERIC:
404	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_LUN:GENERIC:
405	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_MEMORY:GENERIC:

406	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_PORT:BACKEND:
407	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_PORT:EXTERNAL:
408	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_PORT:FRONTEND:
409	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_PORT:GENERIC:
410	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_PORT:INTERNAL:
411	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_PORT:PRIVATE:
412	VIRTUAL_MACHINE:VMWARE:	USES	VIRTUAL_PORT:PROVIDER:
413	VIRTUAL_MEMORY:GENERIC:	ASSOCIATED	ACTION:GENERIC:
414	VIRTUAL_MEMORY:GENERIC:	APPLY	POLICY:ENTITY_SCALE:
415	VIRTUAL_PORT:BACKEND:	ASSOCIATED	ACTION:GENERIC:
416	VIRTUAL_PORT:BACKEND:	CONFIGURED	INTERFACE:GENERIC:
417	VIRTUAL_PORT:EXTERNAL:	ASSOCIATED	ACTION:GENERIC:
418	VIRTUAL_PORT:EXTERNAL:	CONFIGURED	INTERFACE:GENERIC:
419	VIRTUAL_PORT:FRONTEND:	ASSOCIATED	ACTION:GENERIC:
420	VIRTUAL_PORT:FRONTEND:	CONFIGURED	INTERFACE:GENERIC:
421	VIRTUAL_PORT:GENERIC:	ASSOCIATED	ACTION:GENERIC:
422	VIRTUAL_PORT:GENERIC:	CONFIGURED	INTERFACE:GENERIC:
423	VIRTUAL_PORT:INTERNAL:	ASSOCIATED	ACTION:GENERIC:
424	VIRTUAL_PORT:INTERNAL:	CONFIGURED	INTERFACE:GENERIC:
425	VL_ENDPOINT:BIDIRECTIONAL:OPENSTACK	IMPLEMENTS	NETWORK:OPENSTACK:
426	VL_ENDPOINT:BIDIRECTIONAL:OPENSTACK	IMPLEMENTS	NETWORK:OPENSTACK:EXTERNAL
427	VL_ENDPOINT:BIDIRECTIONAL:OPENSTACK	IMPLEMENTS	NETWORK:OPENSTACK:PRIVATE
428	VL_ENDPOINT:BIDIRECTIONAL:OPENSTACK	IMPLEMENTS	NETWORK:OPENSTACK:PROVIDER
429	VL_ENDPOINT:BIDIRECTIONAL:OPENSTACK	CONNECT_THROUGH	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK
430	VL_ENDPOINT:BIDIRECTIONAL:OPENSTACK	CONNECTED	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK
431	VNF:GENERIC:	ASSOCIATED	ACTION:GENERIC:
432	VNF:GENERIC:	MEASURE	CONDITION:GENERIC:
433	VNF:GENERIC:	USES	LINK:GENERIC:
434	VNF:GENERIC:	MEASURE	MONITOR:GENERIC:
435	VNF:GENERIC:	USES	NETWORK:BACKEND:
436	VNF:GENERIC:	USES	NETWORK:EXTERNAL:
437	VNF:GENERIC:	USES	NETWORK:FRONTEND:
438	VNF:GENERIC:	USES	NETWORK:GENERIC:
439	VNF:GENERIC:	USES	NETWORK:INTERNAL:
440	VNF:GENERIC:	USES	NETWORK:OPENSTACK:
441	VNF:GENERIC:	USES	NETWORK:OPENSTACK:EXTERNAL
442	VNF:GENERIC:	USES	NETWORK:OPENSTACK:PRIVATE
443	VNF:GENERIC:	USES	NETWORK:OPENSTACK:PROVIDER
444	VNF:GENERIC:	APPLY	POLICY:AFFINITY:
445	VNF:GENERIC:	APPLY	POLICY:ANTI_AFFINITY:
446	VNF:GENERIC:	APPLY	POLICY:ENTITY_RANGE:
447	VNF:GENERIC:	APPLY	POLICY:OVER_SUBCRIPTION:
448	VNF:GENERIC:	APPLY	POLICY:POSTPRE_PROCESSING:
449	VNF:GENERIC:	APPLY	POLICY:VALUE_GENERATION:
450	VNF:GENERIC:	APPLY	POLICY:VALUE_VALIDATION:
451	VNF:GENERIC:	STATUS_CHANGED_BY	PROPAGATION_RULE:GENERIC:
452	VNF:GENERIC:	INCLUDE	VNF:GENERIC:
453	VNF:GENERIC:	INCLUDE	VNF_COMPONENT:GENERIC:
454	VNF:GENERIC:	INCLUDE	VNF_COMPONENT:SITESCOPE:
455	VNF:GENERIC:	EXPOSE	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK
456	VNF:NFVD:	INCLUDE	VNF_COMPONENT:ASSURANCE_GATEWAY:
457	VNF:NFVD:	INCLUDE	VNF_COMPONENT:ECP:
458	VNF:NFVD:	INCLUDE	VNF_COMPONENT:HPSA:
459	VNF:NFVD:	INCLUDE	VNF_COMPONENT:LOCK_MANAGER:
460	VNF:NFVD:	INCLUDE	VNF_COMPONENT:NEO4J:
461	VNF:NFVD:	INCLUDE	VNF_COMPONENT:OPEN_MEDIATION:
462	VNF:NFVD:	INCLUDE	VNF_COMPONENT:ORACLE:
463	VNF:NFVD:	INCLUDE	VNF_COMPONENT:POSTGRES:
464	VNF:NFVD:	INCLUDE	VNF_COMPONENT:SITESCOPE:
465	VNF:NFVD:	INCLUDE	VNF_COMPONENT:SOSA:

466	VNF:NFVD:	INCLUDE	VNF_COMPONENT:UCA:
467	VNF_COMPONENT:ASSURANCE_GATEWAY:	MEASURE	MONITOR:SELF:
468	VNF_COMPONENT:ASSURANCE_GATEWAY:	HA	VNF_COMPONENT:ASSURANCE_GATEWAY:
469	VNF_COMPONENT:ASSURANCE_GATEWAY:	CONNECTED_TO	VNF_COMPONENT:ECP:
470	VNF_COMPONENT:ASSURANCE_GATEWAY:	CONNECTED_TO	VNF_COMPONENT:HPSA:
471	VNF_COMPONENT:ASSURANCE_GATEWAY:	CONNECTED_TO	VNF_COMPONENT:LOCK_MANAGER:
472	VNF_COMPONENT:ASSURANCE_GATEWAY:	READS	VNF_COMPONENT:NEO4J:
473	VNF_COMPONENT:ASSURANCE_GATEWAY:	WRITES	VNF_COMPONENT:NEO4J:
474	VNF_COMPONENT:ASSURANCE_GATEWAY:	CONNECTED_TO	VNF_COMPONENT:OPEN_MEDIATION:
475	VNF_COMPONENT:ASSURANCE_GATEWAY:	CONNECTED_TO	VNF_COMPONENT:SITESCOPE:
476	VNF_COMPONENT:ASSURANCE_GATEWAY:	CONNECTED_TO	VNF_COMPONENT:SOSA:
477	VNF_COMPONENT:ASSURANCE_GATEWAY:	CONNECTED_TO	VNF_COMPONENT:UCA:
478	VNF_COMPONENT:ECP:	MEASURE	MONITOR:SELF:
479	VNF_COMPONENT:ECP:	CONNECTED_TO	VNF_COMPONENT:ASSURANCE_GATEWAY:
480	VNF_COMPONENT:ECP:	HA	VNF_COMPONENT:ECP:
481	VNF_COMPONENT:GENERIC:	ASSOCIATED	ACTION:GENERIC:
482	VNF_COMPONENT:GENERIC:	MEASURE	CONDITION:GENERIC:
483	VNF_COMPONENT:GENERIC:	MEASURE	MONITOR:CUSTOM:
484	VNF_COMPONENT:GENERIC:	MEASURE	MONITOR:GENERIC:
485	VNF_COMPONENT:GENERIC:	MEASURE	MONITOR:SELF:
486	VNF_COMPONENT:GENERIC:	USES	NETWORK:BACKEND:
487	VNF_COMPONENT:GENERIC:	USES	NETWORK:EXTERNAL:
488	VNF_COMPONENT:GENERIC:	USES	NETWORK:FRONTEND:
489	VNF_COMPONENT:GENERIC:	USES	NETWORK:GENERIC:
490	VNF_COMPONENT:GENERIC:	USES	NETWORK:INTERNAL:
491	VNF_COMPONENT:GENERIC:	USES	NETWORK:OPENSTACK:
492	VNF_COMPONENT:GENERIC:	INCLUDE	PHYSICAL_MACHINE:GENERIC:
493	VNF_COMPONENT:GENERIC:	APPLY	POLICY:AFFINITY:
494	VNF_COMPONENT:GENERIC:	APPLY	POLICY:ANTI_AFFINITY:
495	VNF_COMPONENT:GENERIC:	APPLY	POLICY:ENTITY_RANGE:
496	VNF_COMPONENT:GENERIC:	APPLY	POLICY:OVER_SUBCRIPTION:
497	VNF_COMPONENT:GENERIC:	APPLY	POLICY:POSTPRE_PROCESSING:
498	VNF_COMPONENT:GENERIC:	APPLY	POLICY:VALUE_GENERATION:
499	VNF_COMPONENT:GENERIC:	APPLY	POLICY:VALUE_VALIDATION:
500	VNF_COMPONENT:GENERIC:	STATUS_CHANGED_BY	PROPAGATION_RULE:GENERIC:
501	VNF_COMPONENT:GENERIC:	INCLUDE	VIRTUAL_MACHINE:GENERIC:
502	VNF_COMPONENT:GENERIC:	INCLUDE	VIRTUAL_MACHINE:KVM:
503	VNF_COMPONENT:GENERIC:	INCLUDE	VIRTUAL_MACHINE:VMWARE:
504	VNF_COMPONENT:GENERIC:	INCLUDE	VNF_COMPONENT:GENERIC:
505	VNF_COMPONENT:HPSA:	MEASURE	MONITOR:SELF:
506	VNF_COMPONENT:HPSA:	CONNECTED_TO	VNF_COMPONENT:ASSURANCE_GATEWAY:
507	VNF_COMPONENT:HPSA:	CONNECTED_TO	VNF_COMPONENT:ECP:
508	VNF_COMPONENT:HPSA:	HA	VNF_COMPONENT:HPSA:
509	VNF_COMPONENT:HPSA:	CONNECTED_TO	VNF_COMPONENT:LOCK_MANAGER:
510	VNF_COMPONENT:HPSA:	CONNECTED_TO	VNF_COMPONENT:OPEN_MEDIATION:
511	VNF_COMPONENT:HPSA:	CONNECTED_TO	VNF_COMPONENT:ORACLE:
512	VNF_COMPONENT:HPSA:	CONNECTED_TO	VNF_COMPONENT:POSTGRES:
513	VNF_COMPONENT:HPSA:	CONNECTED_TO	VNF_COMPONENT:SOSA:
514	VNF_COMPONENT:LOCK_MANAGER:	MEASURE	MONITOR:SELF:
515	VNF_COMPONENT:LOCK_MANAGER:	CONNECTED_TO	VNF_COMPONENT:ASSURANCE_GATEWAY:
516	VNF_COMPONENT:LOCK_MANAGER:	CONNECTED_TO	VNF_COMPONENT:HPSA:
517	VNF_COMPONENT:LOCK_MANAGER:	HA	VNF_COMPONENT:LOCK_MANAGER:
518	VNF_COMPONENT:NEO4J:	MEASURE	MONITOR:SELF:
519	VNF_COMPONENT:NEO4J:	HA	VNF_COMPONENT:NEO4J:
520	VNF_COMPONENT:OPEN_MEDIATION:	MEASURE	MONITOR:SELF:
521	VNF_COMPONENT:OPEN_MEDIATION:	HA	VNF_COMPONENT:OPEN_MEDIATION:
522	VNF_COMPONENT:ORACLE:	MEASURE	MONITOR:SELF:
523	VNF_COMPONENT:ORACLE:	HA	VNF_COMPONENT:ORACLE:
524	VNF_COMPONENT:POSTGRES:	MEASURE	MONITOR:SELF:
525	VNF_COMPONENT:POSTGRES:	HA	VNF_COMPONENT:POSTGRES:

526	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	AVAILABILITY_ZONE:OPENSTACK:
527	VNF_COMPONENT:SITESCOPE:	MEASURE	CONDITION:GENERIC:
528	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	DATACENTER:GENERIC:
529	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	ENCLOSURE:GENERIC:
530	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	HYPERVISOR:GENERIC:
531	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	HYPERVISOR:KVM:
532	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	HYPERVISOR:VCENTER:
533	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	HYPERVISOR:VMWARE:
534	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	LOCATION:GENERIC:
535	VNF_COMPONENT:SITESCOPE:	MEASURE	MONITOR:CUSTOM:
536	VNF_COMPONENT:SITESCOPE:	MEASURE	MONITOR:SELF:
537	VNF_COMPONENT:SITESCOPE:	INCLUDE	PHYSICAL_MACHINE:GENERIC:
538	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	RACK:GENERIC:
539	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	REGION:OPENSTACK:
540	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	SERVER:GENERIC:
541	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	TENANT:GENERIC:
542	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	VIM:CS8:
543	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	VIM:GENERIC:
544	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	VIM:HELION:
545	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	VIM:ICEHOUSE:
546	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	VIM:OPENSTACK:
547	VNF_COMPONENT:SITESCOPE:	INCLUDE	VIRTUAL_MACHINE:GENERIC:
548	VNF_COMPONENT:SITESCOPE:	INCLUDE	VIRTUAL_MACHINE:KVM:
549	VNF_COMPONENT:SITESCOPE:	INCLUDE	VIRTUAL_MACHINE:VMWARE:
550	VNF_COMPONENT:SITESCOPE:	CONNECTED_TO	VNF_COMPONENT:OPEN_MEDIATION:
551	VNF_COMPONENT:SITESCOPE:	HA	VNF_COMPONENT:SITESCOPE:
552	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	ZONE:BACKEND:
553	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	ZONE:CE:
554	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	ZONE:EXTERNAL:
555	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	ZONE:FIREWALL:
556	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	ZONE:FRONTEND:
557	VNF_COMPONENT:SITESCOPE:	MONITORED_BY	ZONE:INTERNAL:
558	VNF_COMPONENT:SOSA:	MEASURE	MONITOR:SELF:
559	VNF_COMPONENT:SOSA:	CONNECTED_TO	VNF_COMPONENT:ASSURANCE_GATEWAY:
560	VNF_COMPONENT:SOSA:	CONNECTED_TO	VNF_COMPONENT:HPSA:
561	VNF_COMPONENT:SOSA:	HA	VNF_COMPONENT:SOSA:
562	VNF_COMPONENT:UCA:	MEASURE	MONITOR:SELF:
563	VNF_COMPONENT:UCA:	CONNECTED_TO	VNF_COMPONENT:NEO4J:
564	VNF_COMPONENT:UCA:	CONNECTED_TO	VNF_COMPONENT:OPEN_MEDIATION:
565	VNF_COMPONENT:UCA:	CONNECTED_TO	VNF_COMPONENT:ORACLE:
566	VNF_COMPONENT:UCA:	CONNECTED_TO	VNF_COMPONENT:POSTGRES:
567	VNF_COMPONENT:UCA:	HA	VNF_COMPONENT:UCA:
568	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK	IMPLEMENTS	VIRTUAL_MACHINE:GENERIC:
569	VNF_ENDPOINT:BIDIRECTIONAL:OPENSTACK	DEPLOY	VIRTUAL_PORT:GENERIC:
570	VNFMANAGER:GENERIC:	USES	CREDENTIALS:GENERIC:
571	VNFMANAGER:GENERIC:	CONTAINS	DESCRIPTOR:GENERIC:
572	VNFMANAGER:GENERIC:	USES	RESOURCE_POOL:GENERIC:
573	VNFMANAGER:GENERIC:	USES	TENANT:GENERIC:
574	VNFMANAGER:GENERIC:	IMPLEMENTS	VNF:GENERIC:
575	VNFMANAGER:GENERIC:	MANAGE	VNF:GENERIC:
576	ZONE:BACKEND:	CONNECTED	ENDPOINT:GENERIC:
577	ZONE:BACKEND:	CONTAINS	NETWORK:BACKEND:
578	ZONE:BACKEND:	CONTAINS	NETWORK:GENERIC:
579	ZONE:BACKEND:	CONTAINS	NETWORK:OPENSTACK:
580	ZONE:CE:	CONTAINS	ENDPOINT:GENERIC:
581	ZONE:EXTERNAL:	CONNECTED	ENDPOINT:GENERIC:
582	ZONE:EXTERNAL:	CONTAINS	NETWORK:EXTERNAL:
583	ZONE:EXTERNAL:	CONTAINS	NETWORK:GENERIC:
584	ZONE:EXTERNAL:	CONTAINS	NETWORK:OPENSTACK:
585	ZONE:FIREWALL:	CONTAINS	ENDPOINT:GENERIC:

586	ZONE:FRONTEND:	CONNECTED	ENDPOINT:GENERIC:
587	ZONE:FRONTEND:	CONTAINS	NETWORK:FRONTEND:
588	ZONE:FRONTEND:	CONTAINS	NETWORK:GENERIC:
589	ZONE:FRONTEND:	CONTAINS	NETWORK:OPENSTACK:
590	ZONE:FRONTEND:	CONTAINS	ZONE:BACKEND:
591	ZONE:INTERNAL:	CONNECTED	ENDPOINT:GENERIC:
592	ZONE:INTERNAL:	CONTAINS	NETWORK:GENERIC:
593	ZONE:INTERNAL:	CONTAINS	NETWORK:INTERNAL: