NOKIA

NSP
Network Services Platform

Network Functions Manager - Packet (NFM-P)

Release 17.3

NFV Solutions Guide

3HE-11999-AAAA-TQZZA

Issue 1

March 2017

Legal notice

Contents

About this document

Purpose

The *NSP NFM-P NFV Solutions Guide* describes the network function virtualization (NFV) solution and provides information and workflows for managing and monitoring virtualized network functions with the NFM-P.

Safety information

For your safety, this document contains safety statements. Safety statements are given at points where risks of damage to personnel, equipment, and operation may exist. Failure to follow the directions in a safety statement may result in serious consequences.

Document support

Customer documentation and product support URLs:

- Customer Documentation Welcome Page
- Technical support

How to comment

Documentation feedback

- Documentation feedback

Part I: Getting started

Overview

Purpose

This part provides information on the NFV features added in the most recent releases of the NFM-P and provides an overview of the NFV solutions.

Contents

1 Safety information

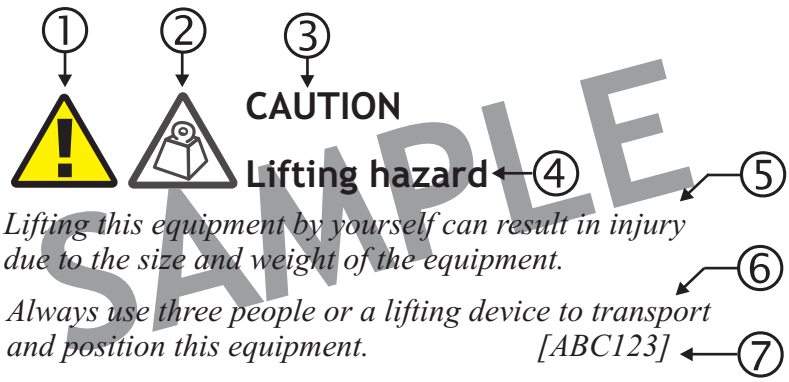
1.1 Structure of safety statements

1.1.1 Overview

This topic describes the components of safety statements that appear in this document.

1.1.2 General structure

Safety statements include the following structural elements:



Item	Structure element	Purpose
1	Safety alert symbol	Indicates the potential for personal injury (optional)
2	Safety symbol	Indicates hazard type (optional)
3	Signal word	Indicates the severity of the hazard
4	Hazard type	Describes the source of the risk of damage or injury
5	Safety message	Consequences if protective measures fail
6	Avoidance message	Protective measures to take to avoid the hazard
7	Identifier	The reference ID of the safety statement (optional)

1.1.3 Signal words

The signal words identify the hazard severity levels as follows:

Signal word	Meaning
DANGER	Indicates an extremely hazardous situation which, if not avoided, will result in death or serious injury.
WARNING	Indicates a hazardous situation which, if not avoided, could result in death or serious injury.
CAUTION	Indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a hazardous situation not related to personal injury.

2 What's new?

2.1 What's new in NFM-P Release 17.3?

2.1.1 Overview

This section highlights new NFV management features for NFM-P Release 17.3 and provides pointers into the documentation for information about using the features. Feature lists and high-level feature descriptions are also available in the *NSP NFM-P Release Description*.

2.1.2 Maintenance releases

Some releases may not be listed in this section, either because no new features are introduced or the features introduced do not require documentation.

2.1.3 What's new in NFM-P 17.3

Table 1 NFM-P 17.3 features

Feature	Description and Reference
SAM-90537 SAM interface to CBAM (based on VE-VNFM-EM reference point)	The Network Supervision application provides an interface with the CloudBand Application Manager (CBAM) for VNF management. See Chapter 4, "CBAM as VNF manager". See Chapter 5, "Network Supervision web application". See Chapter 6, "VNF threshold policy templates".

3 NFV solution overview

3.1 Network function virtualization overview

3.1.1 General information

Network function virtualization (NFV) allows network administrators to uncouple network functions from underlay hardware NEs so that the functions can run as software images. These network functions include load balancers, firewalls, and NAT. The purpose of NFV is to provide a simpler way to deliver and manage the network components required for a virtualized infrastructure. Network administrators are able to dynamically deploy network elements and services without needing to physically provision the underlying routers. The virtualized network element that represents the physical node is called a virtualized network function (VNF).

The NFM-P currently provides two separate solutions for NFV. The solutions exist independently and do not need to be used in conjunction with one another.

The two solutions are differentiated by where the VNF manager component resides:

- CBAM as VNF manager — the VNF manager component resides within the external application CBAM to which the NFM-P provides an interface using the Network Supervision application
- NFM-P as VNF manager — the VNF manager component resides within the NFM-P architecture and can be accessed using the VNF Manager application

In future releases of the NFM-P, CBAM will be used as the primary VNF manager application for the NFM-P solution.

Part II: CBAM as VNF manager solution

Overview

Purpose

This part provides information on the CBAM as VNF manager solution

Contents

4 CBAM as VNF manager

4.1 CBAM NFV solution

4.1.1 Overview

The Nokia CloudBand Application Manager (CBAM) is a VNF manager that automates VNF lifecycle management and cloud resource management. CBAM has standards-based APIs to allow it to work with any vendor VNF, EMS, VIM, or NFV Orchestrator (NFVO).

The NFM-P provides an interface with CBAM, which acts as the VNF manager in this solution. The NFM-P uses the Ve-Vnfm-Em reference point to exchange notifications on VNF lifecycle changes and monitor virtual resources with CBAM. The Network Supervision application allows you to configure a CBAM access point to monitor managed VNFs and execute lifecycle management actions.

The NFM-P provides the following functions when interfacing with CBAM:

- element management for VNFs
- VNF assurance, alarm monitoring, and status tracking
- VNF KPI monitoring
- lifecycle management proxy actions
- policy-based lifecycle changes

Note: When using CBAM as the VNF manager, the NFM-P VNF Manager application is not required. Only the Network Supervision application is needed for the solution.

4.1.2 Supported VNFs

The CBAM as VNF manager solution currently supports the following VNFs:

- CMG
- CMM

4.1.3 CBAM access point

The NFM-P supports CBAM integration through a CBAM access point created in the Network Supervision application. You must input CBAM and NFM-P login credentials in order to create the access point. You must also specify an NFM-P discovery rule to enable VNF automatic discovery. This procedure requires administrator or nfvMgmt access privileges.

Once the access point is successfully created, you can view a list of CBAM access points associated with the NFM-P. From the list of CBAM access points, you can cross-launch to the CBAM GUI, rescan CBAM VNFs, or open the Details tab for an access point. The Details tab allows you to view the associated discovery rule and the connection status for the specified CBAM access point. The connection status is verified by the NFM-P once every two minutes.

The following are prerequisites before creating a CBAM access point:

- Ensure the specified CBAM login credentials have access to the CBAM APIs via ReST. See the *Installing CloudBand Application Manager Guide* for more information.
- Ensure the NFM-P SSL certificates are installed on CBAM. See the *CloudBand Application Manager Administrator Guide* for more information.

See Chapter 5, “Network Supervision web application” for more information about CBAM access point creation.

4.1.4 VNF discovery

You cannot instantiate VNFs using a CBAM interface. You can only discover VNFs from CBAM using an NFM-P discovery rule specified during access point creation. The NFM-P requires that the VNFD template for discovered VNFs have post-instantiation scripts to enable SNMP and configure other protocols necessary for automatic node discovery. When the NFM-P discovers VNFs from a CBAM access point, it adds them as a rule element for the associated discovery rule.

The Unmanaged VNFs tab lists the VNFs managed by the CBAM instance that were not discovered by the specified NFM-P discovery rule. The list is updated once every two minutes.

4.1.5 Lifecycle change notifications

Lifecycle change notifications (LCNs) are messages sent from CBAM to the NFM-P with details on VNF lifecycle updates. When the CBAM access point is created, the NFM-P requests two different LCN subscriptions for the CMG and CMM. LCNs are used to inform the NFM-P of changes related to VNF instantiation, termination, scaling, healing, or variable modifications. When the NFM-P receives an LCN, it scans the VNF information from the CBAM access point and updates its VNF database accordingly.

Regardless of LCNs, the NFM-P automatically polls the CBAM access point for VNF updates once every hour.

Note: In a redundant deployment scenario, the CBAM LCN subscription fails after main server switchover takes place. To resubscribe, you must restart the Network Supervision application and open the CBAM access point.

4.1.6 VNF lifecycle management

Certain VNF lifecycle changes can be initiated from CBAM or the Network Supervision application. Whenever a lifecycle change is triggered in CBAM, it informs the NFM-P via an LCN.

VNFs can be instantiated in CBAM and advertised to the NFM-P via an LCN. When the NFM-P receives information on a newly instantiated VNF, it creates an associated VNF object and attaches a discovery rule to that object automatically. When the NFM-P discovers a VNF, it retrieves information related to supported operations, scaling and healing templates, extensions, and compute resources.

VNFs can be terminated in CBAM and advertised to the NFM-P via an LCN. When the NFM-P receives information on a terminated VNF, it unmanages the VNF object and removes all associated VNFCs.

VNFs can be deleted in CBAM and advertised to the NFM-P via an LCN. When the NFM-P receives information on a deleted VNF, it removes the VNF from its database and unmanages the associated network element.

VNFs can be healed to trigger a reboot in CBAM or the Network Supervision application. Healing must be enabled in the CBAM VNFD before this operation can be performed in the GUI. If the VNFD requires additional parameters for VNF healing, the parameters are visible in the Network Supervision application.

VNFs can be scaled in or scaled out in CBAM or the Network Supervision application. Scaling must be enabled in the CBAM VNFD before this operation can be performed in the GUI. When performing a scaling operation, you must specify a scaling aspect and a new level. The scaling level cannot exceed the maximum scaling level specified in the CBAM VNFD. If the VNFD requires additional parameters for VNF scaling, the parameters are visible in the Network Supervision application.

4.1.7 VNF threshold policies

You can assign a threshold policy to a VNF to allow the NFM-P to trigger automatic lifecycle management operations based on defined KPIs or alarms. A threshold policy allows you to monitor a set of pre-defined KPIs and create rules to define when the application indicates an overload, underload, or healing condition. The policy also allows you define an automatic triggered action to be performed when any of these conditions is met. These corrective actions include performing a scaling operation, performing a healing operation, or raising an alarm. When a lifecycle management action is triggered, the NFM-P automatically sends a lifecycle change notification to CBAM.

You can create a CMM or CMG template to define a list of conditions and specify an action to be automatically performed when those conditions are met. The template can be used to create a VNF threshold policy, but you can modify the default conditions and actions imported from the template each time you create a new policy.

See Chapter 5, “Network Supervision web application” for more information about configuring and using VNF threshold policies in the Network Supervision application. See Chapter 6, “VNF threshold policy templates” for more information about templates.

4.1.8 Network Supervision application

The primary NFM-P interface for using CBAM as a VNF manager is the Network Supervision application. See Chapter 5, “Network Supervision web application” for more information.

4.2 CBAM VNFD requirements

4.2.1 Overview

The VNF Descriptor is a package that describes the configuration of the VNF network. It consists of OpenStack Heat templates which define VNF specifications. This section describes the configuration requirements for the CBAM VNFD to allow the NFM-P to discover and manage CBAM VNFs.

For more information on CBAM VNFD template creation, see the CBAM documentation suite.

4.2.2 Template requirements

Ensure the CBAM VNFD templates meet the following requirements:

- The template must populate the **`vnfProductName`** parameter. This parameter allows the NFM-P to determine which VNFs it is not currently managing.
- The VNF instantiation workflow should include the application startup. This ensures that CBAM sends LCNs to the NFM-P only when the VNF application is up. If the application startup is not included in the VNF instantiation workflow, the discovery of the VNF into the NFM-P will be delayed.
- The ansible workflow should push initial configuration on the VNF. This configuration entails the prerequisite configuration requirements for a network element to be discovered by the NFM-P. The system interface should be configured based on the value defined in the **`systemIpAddr`** extension. This configuration minimizes the error situation where the actual system interface IPAddress is different from its definition in the extension.
- The VNFC healing workflow, if implemented, should include the additionalParam **`vnfcToHeal`** configured with a resourceId. The template must not us a UUID as the parameter value, as is already in use as an OpenStack term. The VNFD should be VIM agnostic and should use only CBAM-specific information.
- The template must include required parameters that should be pushed to the VNF during deployment. The following parameters should be available to the NFM-P as VNF extensions or VNFC resource metadata:
  - The **`systemIpAddr`** parameter must be available as a VNF extension.

— Each VNFC Resource must contain slotId information that uniquely identifies the card object. For VNFs that do no support cards, there must be information to uniquely identify the object that the NFM-P creates.

— The OAM/CPM VNFC must include the mgmtIP. The key for this metadata should be **nokia_vnf_ipAddr**. For a CMG, the IP from that vnfcResource will be used for the mgmtIP with a **nokia_vnf_slotId** of A.

— Each CMG aspect defined in the template must include an extension that defines the type of card to which is corresponds. This extension helps the domain make decisions during preScale.

The following sample shows resource metadata configured for NFM-P discovery and management.

—

```
server1:
    type: OS::Nova::Server
    depends_on: [ lb_port_int1, lb_port_data1 ]
    metadata: {"nokia_vnf_slotId": { get_param: lbSlot }, "nokia_vnf_cardType":
"LB" }
    properties:
        name: { list_join: [ "-", [ { get_param: prefix }, lb, { get_param:
lbSlot } ] ] }
            image: { get_param: imageId }
            flavor: { get_param: flavorId }
            config_drive: "true"
            user_data:
                str_replace:
                template: "TiMOS: slot=$slot chassis=VSR card=iom-v mda/1=m20-v
features=795"
                params:
                    $slot: { get_param: lbSlot }
        user_data_format: "RAW"
        networks:
          - port: { get_resource: lb_port_int1 }
          - port: { get_resource: lb_port_data1 }
```

5 Network Supervision web application

5.1 Network Supervision web application

5.1.1 General information

The Network Supervision application is the interface for the CBAM as VNF manager solution. The application provides a dashboard for assessing the overall health of your network. You can use the Network Supervision application to monitor groups of VNFs, which are automatically updated as VNFs are added or removed. Each group can be assessed based on KPIs and affecting alarms. You can also use the application to assign VNF objects to a special watch list and to perform alarm management tasks. If you create a CBAM access point, you can use the application to perform manual and automatic lifecycle management operations,

Click on the application menu and view Start Tour and How To? for more information about features and workflows in the Network Supervision application. This chapter discusses only the Network Supervision features that are specific to the NFV solution integrating with CBAM. Refer to the documentation in the application help menu for more general information on using the application.

5.1.2 Product help tours

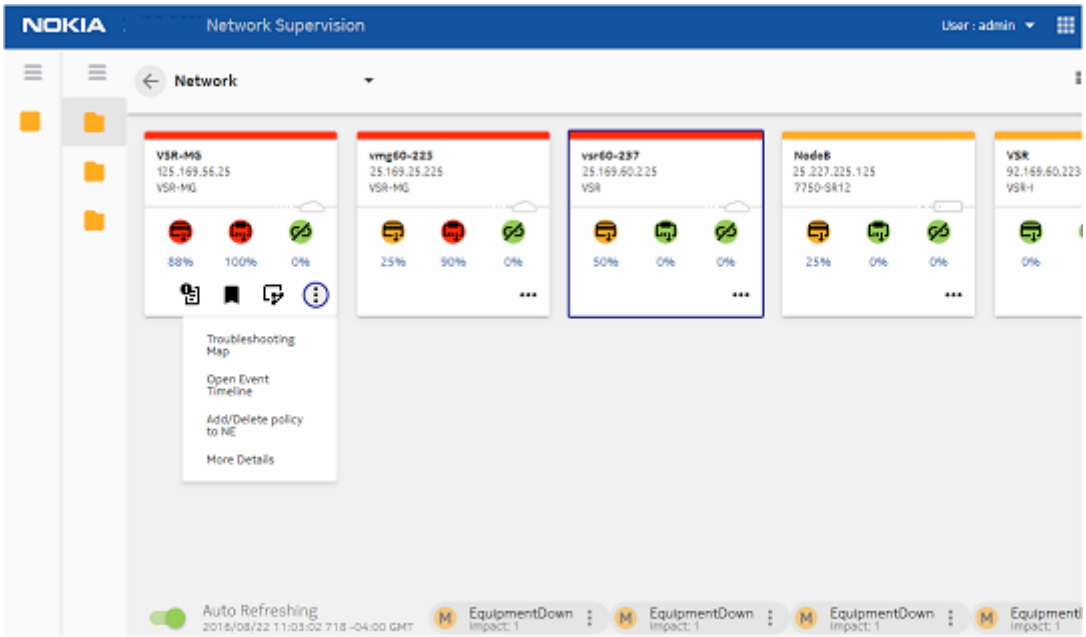
You can click on the menu in the application toolbar or on some panel toolbars to view a list of help tours. These tours are designed to explain the application features and provide workflows for completing management and monitoring tasks.

5.1.3 Application interface

The Network Supervision application interface includes a watch view, summary view, and group matrix. A user with administrator privileges must create supervision groups and summary views using the NFM-P Java UI.

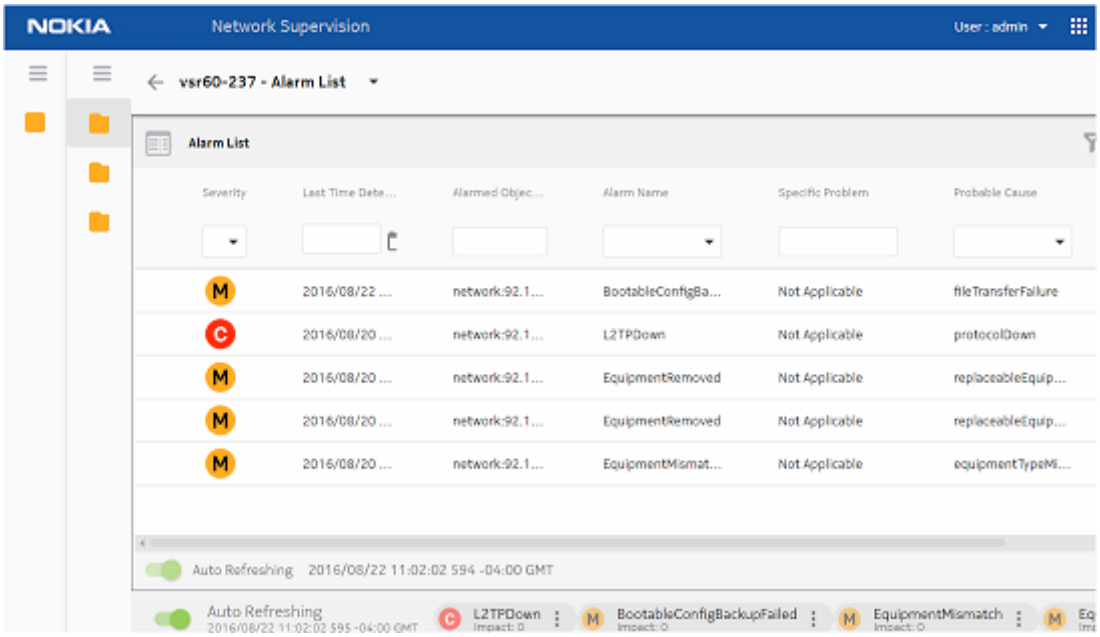
The following figure shows the Network Supervision application with a supervision group selected.

Figure 1 Network Supervision UI



You can use the group matrix to drill down to VNF and VNFC lists and associated alarm lists. The following figure shows the Network Supervision application with an alarm list displayed.

Figure 2 Network Supervision alarm list

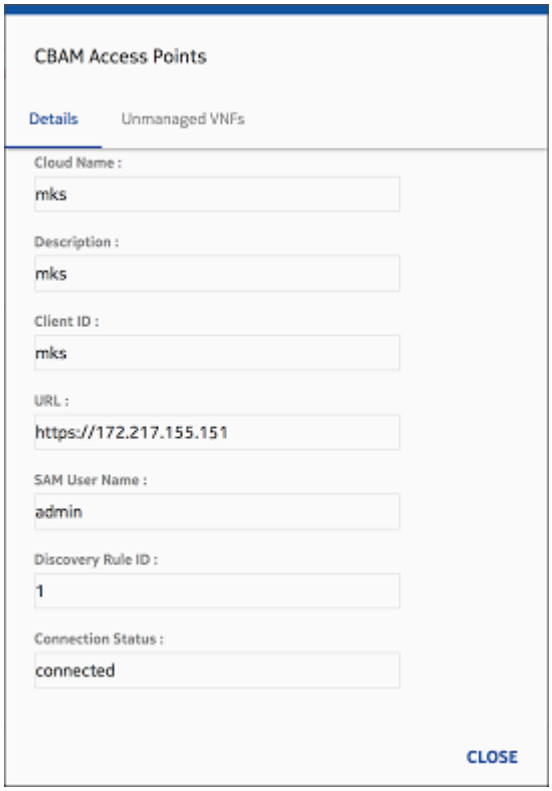


5.1.4 CBAM access point creation

Before you can perform manual and automatic lifecycle management tasks through the Network Supervision application, you must configure a CBAM access point. You must input CBAM and NFM-P login credentials in order to create the access point. You must also specify an NFM-P discovery rule to enable VNF automatic discovery. This procedure requires administrator or nfvMgmt access privileges.

Once the access point is successfully created, you can view a list of CBAM access points associated with the NFM-P. From the list of CBAM access points, you can cross-launch to the CBAM GUI, rescan CBAM NFVs, or open the Details tab for an access point. The Details tab allows you to view the associated discovery rule and the connection status for the specified CBAM access point. The connection status is verified by the NFM-P once every two minutes.

Figure 3 CBAM access point details tab



5.1.5 VNF monitoring and management

The matrix view provides a graphical representation of all the VNFs in a supervision group, or VNFCs if you have drilled down to the VNF components from a VNF. Each tile represents a VNF or VNFC. The tile color of a VNF provides an indication of how many VNFCs are currently in an affected state.

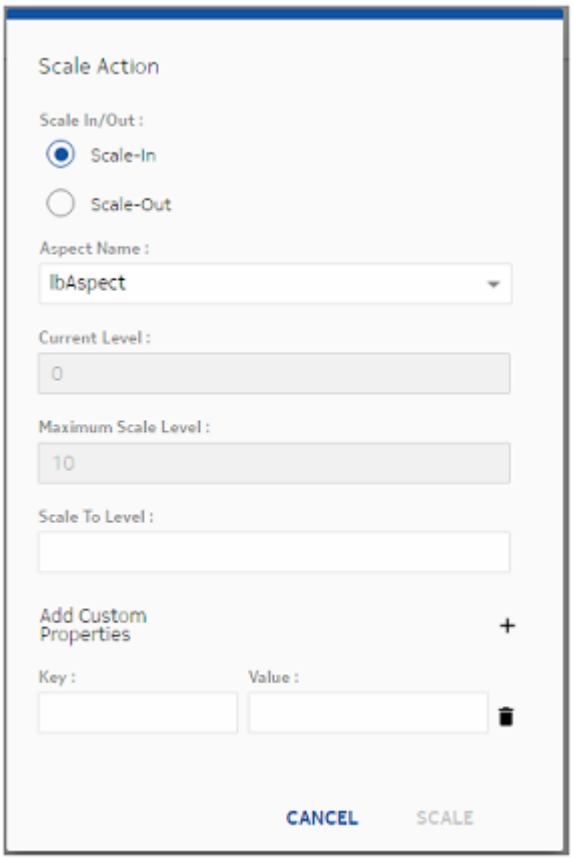
You can expand a tile in the matrix view to see more information about the VNF. Depending on the type of VNF, different KPIs are displayed. For example, the CMM tile displays subscriber count and capacity utilizations statistics. These statistics can be used to indicate an overload or underload condition. These conditions can trigger automatic lifecycle management operations using a VNF threshold policy.

Figure 4 CMM matrix view tile

You can perform a manual scaling operation from the matrix view. Perform a scale-In operation to reduce the processing capacity of the VNF. Perform a scale-Out operation to increase it. Configure the scaling aspect and level to specify the type and number of VMs to be scaled. The scaling level cannot exceed the maximum scaling level specified in the CBAM VNFD. If the templates in the VNF descriptor specify additional custom properties, you can configure them before executing the scaling operation.

On the CMM, only the CPPS VM is supported for scaling operations.

Figure 5 Manual scaling operation



You can perform a manual healing operation from the drilled-down VNFC view in the matrix view. Healing is supported for CMG VMs. A healing operation attempts to reboot a failed VNFC by attempting a soft reboot followed by a hard reboot. Healing must be enabled in the CBAM VNFD before this operation can be performed. If the templates in the VNF descriptor specify additional custom properties, you can configure them before executing the scaling operation.

5.1.6 VNF threshold policies

You can assign a threshold policy to a VNF to allow the NFM-P to trigger automatic lifecycle management operations based on defined KPIs or alarms. A threshold policy allows you to monitor a set of pre-defined KPIs and create rules to define when the application indicates an overload, underload, or healing condition. The policy also allows you define an automatic triggered action to be performed when any of these conditions is met. These corrective actions include performing a scaling operation, performing a healing operation, or raising an alarm. When a lifecycle management action is triggered,

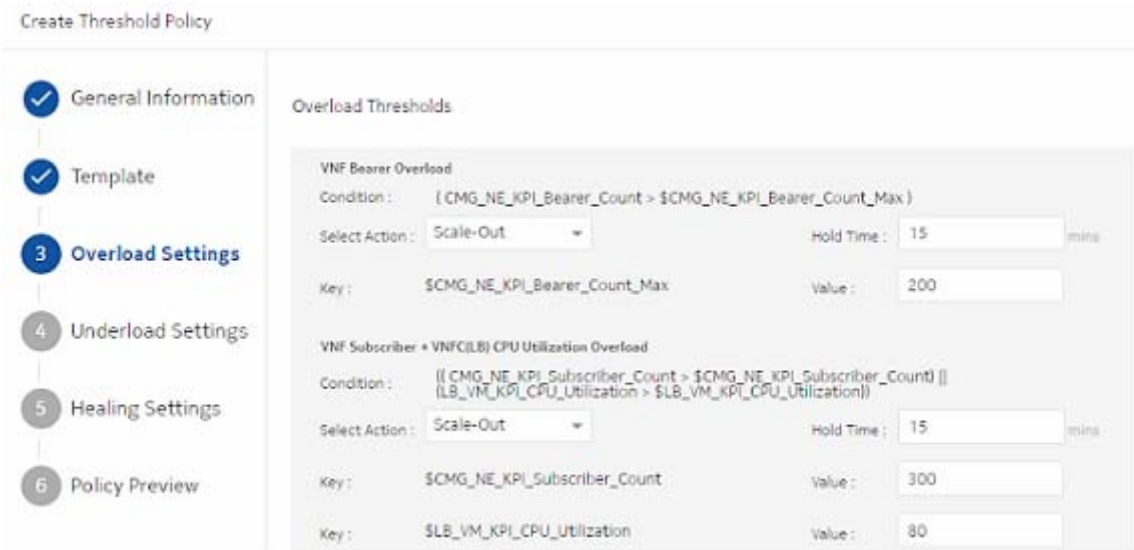
the NFM-P automatically sends a lifecycle change notification to CBAM. You can assign only one VNF threshold policy to a VNF.

You can assign a CMM or CMG template to define a list of conditions and specify an action to be automatically performed when those conditions are met. The template can be used to create a VNF threshold policy, but you can modify the default conditions and actions imported from the template each time you create a new policy.

Use the policy agent in the Network Supervision application to create a VNF threshold policy. The policy agent includes default policy templates for the CMM and CMG. After you select a policy template, you can review the overload, underload, and healing settings to select the actions for each condition. You can also configure the hold time to specify how long the NFM-P should wait before performing the specified action.

See Chapter 6, “VNF threshold policy templates” for more information about templates.

Figure 6 VNF threshold policy creation



KPI monitoring window

When you select a template, you should define a monitoring window and sampling frequency. These parameters ensure the quality of the KPI data. The monitoring window specifies the window of time that the NFM-P monitors the specified KPIs. A monitoring window is a sliding window of time. The sampling frequency specifies how frequently the KPI data is retrieved within the specified monitoring window.

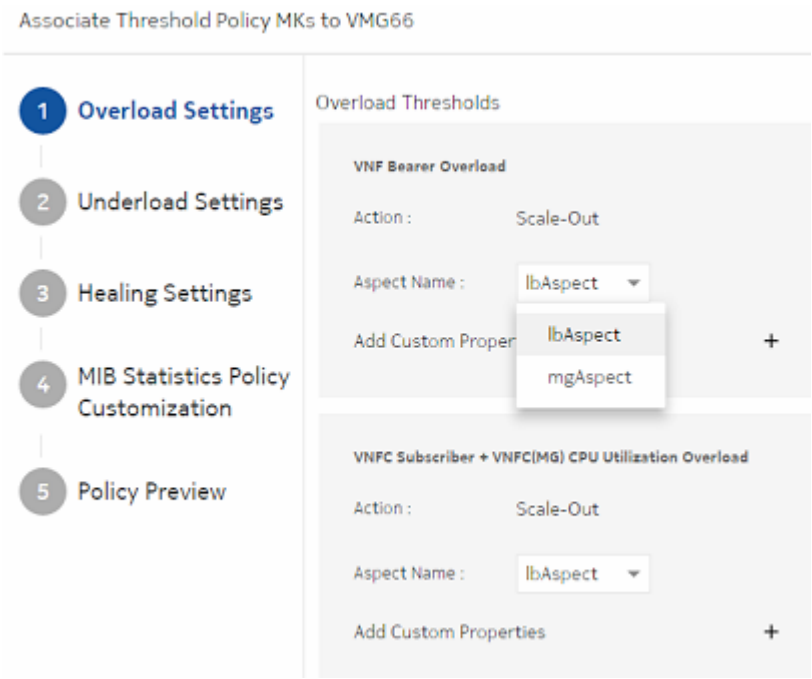
Sampling frequency is applicable to the CMG only. The CMM polls the KPI data every two minutes.

Applying a VNF threshold policy

Once you have created a VNF threshold policy, you can select Add/Delete Policy to NE from the VNF matrix view tile menu to apply the policy to the VNF. If any of the conditions defined in the policy have scaling or healing operations specified, you are presented with a step form when applying the policy to the VNF. You must configure the Aspect Name parameter for each overload and underload threshold rule. The Aspect Name parameter specifies aspect type to be scaled when the scaling operation is triggered. You can click the + button to add custom properties for overload, underload, and healing threshold rules.

When applying a VNF threshold policy to a CMG, you can choose to add a custom MIB statistics policy to the VNF. This option is enabled by default and ensures that statistics collection for the statistics associated with the KPIs in the VNF threshold policy is enabled. If you choose not to add a custom MIB statistics policy, you should use the NFM-P Java GUI to ensure there is a MIB statistics policy associated with the VNF. The policy must enable statistics collection for the statistics associated with the KPIs in the VNF threshold policy. See “Condition syntax” (p. 36) for a table that shows the mapping between VNF KPIs and NFM-P statistics.

Figure 7 VNF threshold policy application



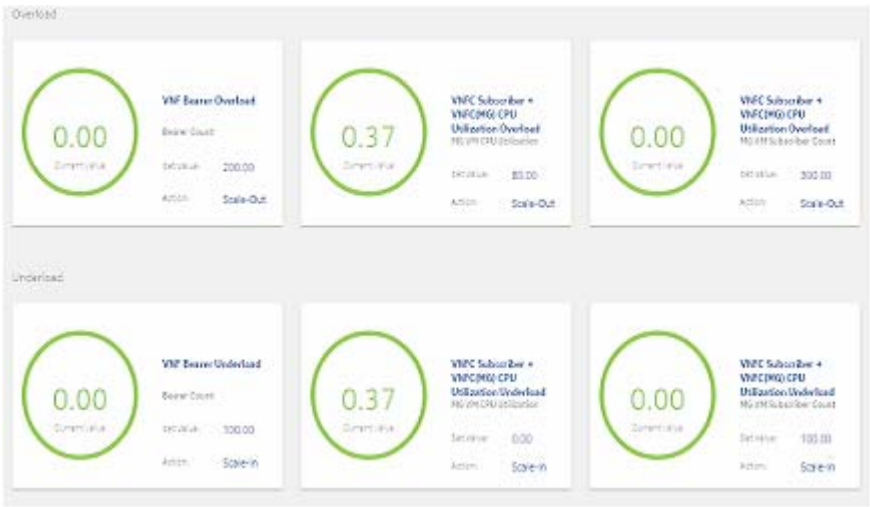
Note: When applying a VNF threshold policy with only alarm actions (as opposed to healing or scaling actions) to a CMM, no policy application configuration steps are required.

When applying a VNF threshold policy with only alarm actions (as opposed to healing or scaling actions) to a CMG with no associated MIB statistics policy, no policy application configuration steps are required.

Monitoring a VNF threshold policy

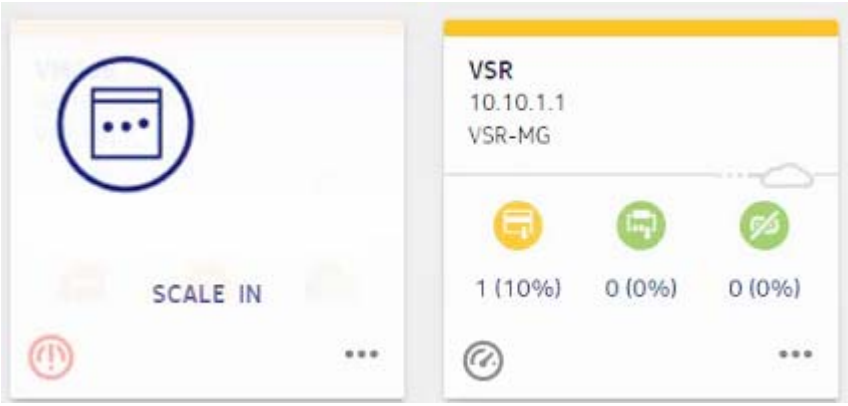
When a policy has been applied to a VNF, the policy monitoring icon appears on the VNF tile in the matrix view. You can click on the policy monitoring icon to view the status of each of the overload, underload, and healing threshold rules specified in the policy. The policy info view shows the conditions, thresholds, and actions for each rule. It also shows the current value of the condition, allowing you to assess how close the condition is to reaching the specified threshold.

Figure 8 VNF threshold policy status



When a threshold is crossed and the VNF threshold policy triggers a scaling or healing operation, the policy monitoring icon changes to red and the operation is shown in-progress on the VNF tile. While the operation is being performed, you can disable to policy to cancel the action.

Figure 9 VNF threshold policy tiles



6 VNF threshold policy templates

6.1 Policy template design

6.1.1 Overview

VNF threshold policies specify the KPIs used to indicate whether a VNF or VNFC is in an overload, underload, or healing condition. The policy can be used to automatically trigger lifecycle management operations such as scaling or healing. To create a VNF threshold policy, you must specify a template that details the default KPIs and actions for each condition. The NFM-P includes sample templates for the CMM and CMG, but you can also create a new template. This chapter describes the format and syntax requirements for creating a new VNF threshold policy template.

6.1.2 YAML

OpenStack templates are written using the YAML markup language. This chapter describes only the formatting and syntax requirements to create a *.yaml* file for VNF threshold policy templates. For more information about YAML, see the CBAM or OpenStack documentation suites.

6.1.3 Workflow to create a VNF threshold policy template

The following workflow describes the steps to create a VNF threshold policy template. See 6.1.4 “Template format” (p. 34) for information about the overall template format and keyword syntax.

1

Define the template name, NE type, monitoring window, and sampling frequency at the top of the *.yaml* file. See 6.1.4 “Template format” (p. 34) and 6.1.5 “Template keywords” (p. 35).

2

Note:
At least one of the following three steps is required for template creation.

If required, create an Overload_Condition_Criteria section and define rules elements. See 6.1.6 “Rules” (p. 35) and 6.1.7 “Rule keywords” (p. 35).

3

If required, create an Underload_Condition_Criteria section and define rules elements. See 6.1.6 “Rules” (p. 35) and 6.1.7 “Rule keywords” (p. 35).

4

If required, create a Healing_Condition_Criteria section and define rules elements. See 6.1.6 “Rules” (p. 35) and 6.1.7 “Rule keywords” (p. 35).

5

Review the template for any syntax errors.

6

Format the template to ensure the Network Supervision application user can read and understand contents clearly.

7

Save the *.yaml* file with a filename that indicates the purpose of the template.

8

Upload the template to the template directory in the NSP NFM-P file system. See 6.1.8 “Template directory” (p. 38).

6.1.4 Template format

The policy template must follow the format below. The template can exclude any of the sections for overload, underload, or healing condition criteria, but it must include at least one of them. The instances of **<string>**, **<integer>**, and **<rule #>** should be replaced with values to be determined by the user.

-

```
Name: <string>
NE_Type: <string>
Monitoring_Window: <integer>
Sampling_Frequency: <integer>
Overload_Condition_Criteria:
    Rules:
    - <rule 1>
    - <rule 2>
    - <rule n>
Underload_Condition_Criteria:
    Rules:
    - <rule 1>
    - <rule 2>
    - <rule n>
Healing_Condition_Criteria:
    Rules:
    - <rule 1>
```

```
- <rule 2>
- <rule n>
```

6.1.5 Template keywords

You must specify values for the following keywords at the top of the template:

- **Name** — identifies the policy as it will appear in the Network Supervision application policy agent
- **NE_Type** — specifies whether the template is for the CMM or CMG; specify “cmm” or “cmg”
- **Monitoring_Window** — specifies the window of time that the NFM-P monitors the specified KPIs; specify a time between 1 and 120 minutes
- **Sampling_Frequency** — specifies how frequently the KPI data is retrieved within the specified monitoring window; specify 1, 5, 10, 15, 30, 45, or 60 minutes

Note: **Sampling_Frequency** is not required for the CMM. By default, the sampling window for the CMM is 2.

6.1.6 Rules

If the policy template includes a section for overload, underload, or healing criteria, it must include at least one rule in the section. Rules must follow the format below. The instances of variables such as **<string>** should be replaced with values to be determined by the user.

—

```
- Name : <string>
 Condition : (<KPI_Value> <operator> $<Threshold_Value>)
 Action : <action>
 Hold_Time: <integer>
 Values : {$<Threshold_Value>: '<integer>'}
```

—

Note: **$** is a prefix used to identify variables.

6.1.7 Rule keywords

You must specify values for the following keywords within a template rule:

- **Name** — identifies the rule name as it will appear in the Network Supervision application
- **Condition** — specifies a logical statement including a KPI value, operator, and threshold value

- **`Action`** — specifies the action to perform if the condition is satisfied; specify “scaleOut”, “scaleIn”, “heal”, or “alarm”
- **`Hold_Time`** — specifies the time, in minutes, before waiting to trigger the action again if the condition is still satisfied
- **`Values`** — defines values for variables used in the logical conditions

Condition syntax

The Condition line is a logical statement that needs to be satisfied to perform the specified action. It includes at least one KPI value, logical operator, and threshold value. Multiple conditions can be grouped using logical operators. You must include a space between each KPI, operator, and variable.

The threshold value is a user-defined variable used to compare with the specified KPI value. The threshold value name has no syntax requirements, but it is recommended that you choose a name that indicates the KPI and type of threshold. For example, a threshold value that defines the maximum value for the KPI value CMG_NE_KPI_ Bearer_Count should be defined as CMG_NE_KPI_Bearer_Count_Max. Following this convention allows you to more easily compare the configured threshold value with the current value during VNF policy creation.

The following table lists the KPIs that can be used as part of a rule condition.

Table 2 VNF condition KPIs

KPI value	NFM-P class name	Property	MIB name
CMG NE level KPIs			
`CMG_NE_KPI_Subscriber_Count`	isa. PdnGwCardStats	ues	TIMETRA-MOBILE-PDN-MIB. tmnxMobPdnStatEntry
`CMG_NE_KPI_Session_Count`	isa. PdnGwCardStats	sessions	TIMETRA-MOBILE-PDN-MIB. tmnxMobPdnStatEntry
`CMG_NE_KPI_Bearer_Count`	isa. PdnGwCardStats	bearers	TIMETRA-MOBILE-PDN-MIB. tmnxMobPdnStatEntry
`CMG_NE_KPI_Combined_Session_ Count`	isa. PdnGwCardStats	combinedP-dnSessions	TIMETRA-MOBILE-PDN-MIB. tmnxMobPdnStatEntry
`CMG_NE_KPI_Combined_Bearer_ Count`	isa. PdnGwCardStats	combined-Bearers	TIMETRA-MOBILE-PDN-MIB. tmnxMobPdnStatEntry
CMG alarm KPI [1]			
`CMG_NE_ALARM_`<*alarm name*> For example, **`CMG_NE_ALARM_ VNFCHealingRequired`**	—	—	—
CMG VM level KPIs			

Table 2 VNF condition KPIs (continued)

KPI value	NFM-P class name	Property	MIB name
MG_VM_KPI_CPU_Utilization	equipment.CpuUtilizationStats	busyCoreUtil	TIMETRA-SYSTEM-MIB.tmnxCardCpuResMonitorEntry
MG_VM_KPI_Subscriber_Count	isa.PdnGwCardStats	ues	TIMETRA-MOBILE-PDN-MIB.tmnxMobPdnStatEntry
MG_VM_KPI_Session_Count	isa.PdnGwCardStats	sessions	TIMETRA-MOBILE-PDN-MIB.tmnxMobPdnStatEntry
MG_VM_KPI_Bearer_Count	isa.PdnGwCardStats	bearers	TIMETRA-MOBILE-PDN-MIB.tmnxMobPdnStatEntry
MG_VM_KPI_Combined_Session_Count	isa.PdnGwCardStats	combinedPdnSessions	TIMETRA-MOBILE-PDN-MIB.tmnxMobPdnStatEntry
MG_VM_KPI_Combined_Bearer_Count	isa.PdnGwCardStats	combinedBearers	TIMETRA-MOBILE-PDN-MIB.tmnxMobPdnStatEntry
CMG LB VM level KPI			
LB_VM_KPI_CPU_Utilization	equipment.CpuUtilizationStats	busyCoreUtil	TIMETRA-SYSTEM-MIB.tmnxCardCpuResMonitorEntry
CMG card count KPIs			
MG_VM_Count Card type: card_iom_mg_vsr	—	—	—
LB_VM_Count Card type: card_iom_vsr	—	—	—
CMM NE level KPIs			
CMM_NE_KPI_UE_Capacity	ltecmm.CMMInstanceCapacity	totalNumUes	—
CMM_NE_KPI_Percent_Utilization	ltecmm.CMMInstanceCapacity	percentUtilization	—
CMM alarm KPI [1]			
CMM_NE_ALARM_<*alarm name*> For example, **CMM_NE_ALARM_CmmLSS_hostReset**	—	—	—
CMM CPPS VM level KPIs			
CPPS_VM_KPI_UE_Capacity	ltecmm.CppsVNFCMemberCapacity	uesInCpps	—

Table 2 VNF condition KPIs (continued)

KPI value	NFM-P class name	Property	MIB name
CPPS_VM_KPI_Percent_ Utilization	ltecmm. CppsVNFCMem-berCapacity	percentUtiliza-tion	—

Notes:

1. In rule conditions, alarms must always be checked against the number of occurrences. For example, **CMM_NE_ALARM_CmmLSS_hostReset > $No_of_Occurrence**.

   The following table lists the operators that can be used in rule conditions.

Operator	Usage	Operator	Usage
$	Prefix used to identify variables	**%**	modulus
&&	and	**==**	equal to
||	or	**!=**	not equal to
!	not	**>**	greater than
+	addition	<	less than
-	subtraction	>=	greater than or equal to
*****	multiplication	<=	less than or equal to
/	division		

Note: The Network Supervision threshold policy framework may trigger multiple concurrent actions for the same policy. You should ensure that rule definitions in the threshold policy result in logical actions for each condition.

6.1.8 Template directory

Templates must be saved in the */opt/nsp/nfmp/KPITCA/TemplateLibrary* directory. The template directory includes the following sample templates:

- CMM_Rule_Template_Example_*n*.yaml
- CMG_Rule_Template_Example_*n*.yaml
- CMM_Rule_Template_Reference_Sample.yaml
- CMG_Rule_Template_Reference_Sample.yaml

Note: The Reference_Sample files are for reference only. Only the Template_Example files can be used.

The NFM-P supports redundancy for template files. Whenever an rsync operation is triggered, files in the */opt/nsp/nfmp/KPITCA/tca_rsync* directory with a *yaml*, *.sh, .txt.,* and *.json*. extension are duplicated into a standby file server. Changes in this directory on the active server are duplicated every 30 minutes by default.

6.2 Policy template sample

6.2.1 Overview

The policy template sample in section demonstrates the format and syntax required to create a valid policy template.

6.2.2 Sample

```
Name: Sample CMG Rules
NE_Type: cmg
Monitoring_Window: 15
Sampling_Frequency: 5

Overload_Condition_Criteria:
  Rules:
   - Name : VNF Bearer Overload
     Condition : ( CMG_NE_KPI_Bearer_Count > $CMG_NE_KPI_Bearer_Count_Max )
     Action :  scaleOut
     Hold_Time: 15
     Values : {$CMG_NE_KPI_Bearer_Count_Max: '200'}

   - Name : VNFC(MG) Sesion Overload
     Condition : ( MG_VM_KPI_Combined_Session_Count > $MG_VM_KPI_Combined_Session_
Count_Max )
     Action :  scaleOut
     Hold_Time: 15
     Values : {$MG_VM_KPI_Combined_Session_Count_Max: '200'}

   - Name : VNF Subscriber + VNFC(LB) CPU Utilization  Overload
     Condition : (( CMG_NE_KPI_Subscriber_Count > $CMG_NE_KPI_Subscriber_Count)
||
                  (LB_VM_KPI_CPU_Utilization > $LB_VM_KPI_CPU_Utilization))
     Action :  scaleOut
     Hold_Time: 15
     Values : {$CMG_NE_KPI_Subscriber_Count: '300',$LB_VM_KPI_CPU_Utilization:
'80'}

Underload_Condition_Criteria:
  Rules:
   - Name : VNF Bearer Underload
```

```
     Condition : ( CMG_NE_KPI_Bearer_Count > 0 && CMG_NE_KPI_Bearer_Count < $CMG_
NE_KPI_Bearer_Count_Min )
     Action :  scaleIn
     Hold_Time: 15
     Values : {$CMG_NE_KPI_Bearer_Count_Min: '100'}

   - Name : VNFC(MG) Sesion Underload
     Condition : ( MG_VM_KPI_Combined_Session_Count > 0 &&
                   MG_VM_KPI_Combined_Session_Count < $MG_VM_KPI_Combined_Session_
Count_Min )
     Action :  scaleIn
     Hold_Time: 15
     Values : {$MG_VM_KPI_Combined_Session_Count_Min: '50'}

   - Name : VNF Subscriber + VNFC(LB) CPU Utilization  Underload
     Condition : ((CMG_NE_KPI_Subscriber_Count >0 && CMG_NE_KPI_Subscriber_Count
< $CMG_NE_KPI_Subscriber_Count) || (LB_VM_KPI_CPU_Utilization > 0 && LB_VM_KPI_
CPU_Utilization < $LB_VM_KPI_CPU_Utilization))
     Action :  scaleIn
     Hold_Time: 15
     Values : {$CMG_NE_KPI_Subscriber_Count: '100',$LB_VM_KPI_CPU_Utilization:
'20'}

Healing_Condition_Criteria:
  Rules:
   - Name : VNFCDown Heal Alarm
     Condition : ( CMG_NE_ALARM_AutoHealRequired > $No_Of_Occurrence)
     Action :  heal
     Hold_Time: 15
     Values : {$No_Of_Occurrence: '1'}
```

Part III: NFM-P as VNF manager solution

Overview

Purpose

This part provides information on the NFM-P as VNF manager solution.

Contents

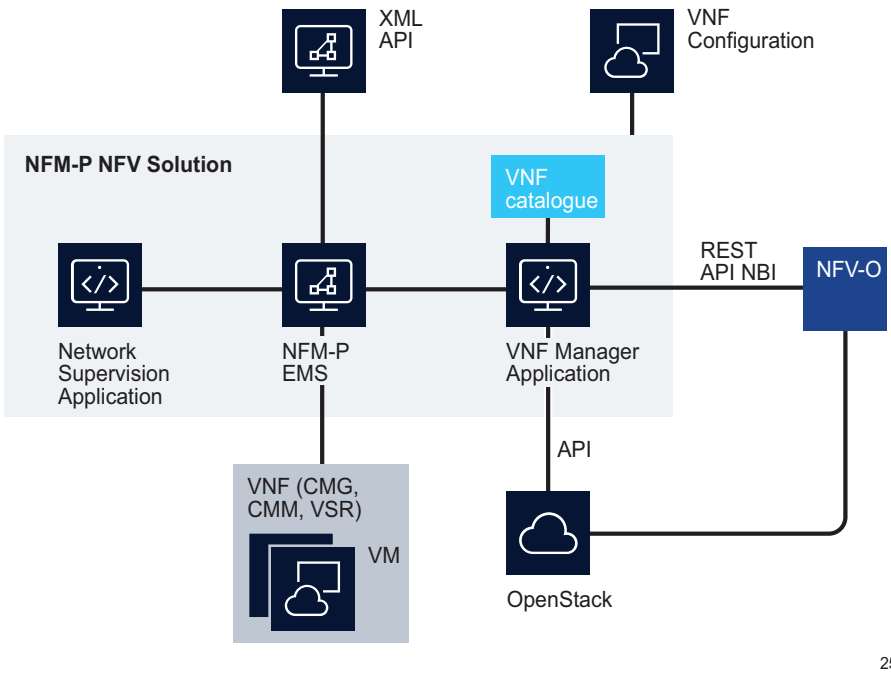
7 NFM-P as VNF manager

7.1 NFM-P NFV solution

7.1.1 Overview

The NFM-P can be used as the VNF manager for the NFV solution by using the VNF Manager application. The NFM-P provides an interface with OpenStack to enhance VNF management with more monitoring, assurance, and management functionality. This VNF-aware network management system (NMS) extends NFM-P existing network element management and assurance to VNFs and VNF lifecycle functions. The NFM-P NFV feature set is composed of two applications (VNF Manager and Network Supervision) as well as support for network function element management system (EMS) in the NFM-P Java GUI.

Figure 10 NFM-P as VNF manager solution



7.1.2 OpenStack

OpenStack is an open-source cloud management system (CMS) that can be used for NFV management. OpenStack provides the NFV infrastructure and orchestration components that can be used to perform lifecycle management tasks on managed VNFs and other virtualized network elements.

The NFM-P provides an interface with OpenStack through the VNF Manager application.

Orchestration

OpenStack orchestration allows the user to manage the lifecycle of VNFs and network infrastructure within the OpenStack cloud. The module used for orchestration is called Heat. The Heat orchestration engine is designed to start cloud applications based on template files that are specialized for different types of VNFs. These Heat orchestration template (HOT) files describe a list of OpenStack resources called a stack. Heat maps these resources to virtual machines (VMs) based on VNF requirements.

HOT files can be used to define the following VNF specifications:

- resource allocation
- lifecycle tasks, including deployment, rebooting, and scaling
- auto-scaling instructions

Cloud Management

In addition to Heat, OpenStack uses several other modules for VNF management. OpenStack cloud management programs include the following:

- Nova—used for compute resources
- Glance—discovery and provisioning for disk and server images
- Neutron—OpenStack networking
- Cinder—block-level storage
- Ceilometer—billing and telemetry
- Keystone—authentication

7.1.3 VNF application cross-launch

The NFM-P Java GUI supports cross-launching the VNF Manager and Network Supervision applications from the VNF or VNFC in the equipment tree. Cross-launching to the VNF Manager application allows you to perform manual lifecycle management tasks such as VNF instantiation or scaling. Cross-launching the Network Supervision application allows you to view the selected VNF or VNFC in the context of its supervision group. To cross-launch a VNF application from the equipment tree, right-click a network element or card slot and choose VNF Management to select an application to launch.

7.1.4 Licensing

The NFM-P allows you to view NFM-P NFV license statuses and counters in the Java GUI. The license status is listed as either valid, invalid, no license, or locked. You can view license counters to see how many NFM-P VNF and VNFC licenses of each type have been consumed. The NFM-P raises a node license alarm when the license is invalid and includes a UUID.

Note: After the NFM-P discovers the number of CMG VNFs allowed by the license, the NFM-P can no longer discover the CMG.

7.1.5 VNF assurance and monitoring

The NFM-P improves visibility of faults in the VNF network with alarm management. You can use the Network Supervision application to view all alarms in a VNF supervision group, and have the ability to drill down to a specific VNF or VNFC. The NFM-P reduces alarm volume and quickens fault resolution by correlating alarms between the VNF application and the virtual infrastructure. You can also view alarms for VNFs in the NFM-P Java GUI, which gives you access to more alarm management functionality, such as threshold-crossing alarms (TCAs).

The Network Supervision application provides a simplified dashboard view of VNFs in the network. The application allows you to track key performance indicators (KPIs) such as down VNFCs, to determine the overall health trend of your network. You can quickly assess network performance, outages, and VNF availability using the Watch List or Group Matrix. You can use this information to determine if a capacity change (scale-in or scale-out) is required.

You can configure the NFM-P to perform an automatic scale-out and automatic healing based on capacity thresholds or network reachability events.

7.1.6 VNF lifecycle management

The NFM-P provides an interface with OpenStack Heat to allow you to perform VNF lifecycle tasks from the VNF Manager application. You can perform the following lifecycle tasks directly from within the VNF Manager application:

- Instantiation—create a VNF instance from a specified cloud access point and a VNF catalog
- Deletion—delete a VNF instance
- Deployment—deploy a VNF instance to the cloud network
- Scaling—reduce or expand processing capacity by adding VMs to a VNF
- Healing—reboot a failing VNF component
- Sync—synchronize a VNF with the OpenStack tenant

Note: Automatic scale-in is not supported.

7.1.7 VNF Manager application

The primary NFM-P interface for this solution is the VNF Manager application. See 8.1 “VNF Manager web application” (p. 49) for more information.

7.2 Supported VNFs

7.2.1 Overview

The NFM-P extends basic EMS support to virtual network elements. Discovery, provisioning, and management workflows are the same as for physical nodes.

The NFM-P supports the following VNFs:

- CMG
- VMM
- VSR

See the *NSP NFM-P User Guide* for more information about NE discovery and management.

7.2.2 CMG

The CMG supports the following virtualized applications:

- Packet Data Network Gateway (VMG-PGW)
- Gateway GPRS Support Node (VMG-GGSN)
- Serving Gateway (VMG-SGW)
- Evolved Packet Data Gateway (VMG-ePDG)

These virtualized applications can be deployed as separate network functions or in combination (for example, VMG-SGW/PGW/GGSN or VMG-SGW/ePDG).

The CMG supports multiple mobile gateway functions including PGW, GGSN, SGW, and the combined SGW/PGW/GGSN. It also supports LTE profiles and policies, golden configuration, LTE reference points, EPS peers and paths, and statistics. These functions operate the same as they do on a physical 7750 MG.

The CMG consists of VMs operating as VNFCs. Each VM is dedicated to a specific set of functions that are replicated across other VMs. A group of VMs are represented as a single instance of an application. The VMs in the group operate in synchronization to support a network functionality that can scale horizontally as required.

The CMG is composed of the following VNFCs:

- operations, administration, and management VM (OAM-VM)—performs control plane functions including routing protocols, management interface functions such as CLI configuration, and VNF/VNFC management

- load balancer VM (LB-VM)—provides network connectivity to mobile gateway functions, load distribution across the MG-VMs, and forwarding of GTP-C/GTP-U and UE-addressed packets to the MG-VM
- mobile gateway VM (MG-VM)—manages services including 3GPP call processing (control and data plane), PCEF, and application assurance (PCEF-enhanced with ADC for application detection and control, and L7 service classification for policy charging control

You can configure automatic scale-out or automatic healing on the CMG. If automatic healing is enabled, the NFM-P will attempt to reboot a VNFC when it goes down. The NFM-P attempts a soft reboot first before attempting a hard reboot, if necessary.

See Chapter 9, “NFV use cases” for more information about automatic scale-out and automatic healing.

7.2.3 VMM

You can enable automatic scale-out on the VMM and configure an automatic scale-out threshold. The automatic scale-out threshold defines the point at which an automatic scale-out operation can be triggered, where the threshold is the total UE capacity of the VMM multiplied by the scale-out factor. When the threshold is reached and/or one or more MAFs is affected by a resource overload node alarm, the NFM-P automatically increases the processing capacity of the VMM by creating an additional virtual MAF on the NE.

The Perform Scale-out (VMM) parameter defines the conditions under which the automatic scale-out is triggered. You can specify AND or OR for the following conditions:

- The number of UEs on the WMM is over capacity.
- At least one MAF cards has a threshold crossing alarm against the number of UEs.

The Auto Scale-out Timer parameter prevents the NFM-P from triggering additional scale-out operations while a previous scale-out is still in progress. You should specify a length of time that exceeds the expected time for a virtual MAF to become operational.

You can also enable automatic healing on the VMM. If automatic healing is enabled, the NFM-P will attempt to reboot a VNFC when it goes down. The NFM-P attempts a soft reboot first before attempting a hard reboot, if necessary.

See Chapter 9, “NFV use cases” for more information about automatic scale-out and healing.

7.2.4 VSR

The Virtual Service Router (VSR) is a software-only version of the 7750 SR. VSR and VSR-I chassis types are supported.

The VSR consists of VMs operating as VNFCs. Each VM is dedicated to a specific set of functions that are replicated across other VMs. A group of VMs are represented as a single instance of an application. The VMs in the group operate in synchronization to support a network functionality that can scale horizontally as required.

The VSR is composed of the following VNFCs:

- IOM-V
- CPM-V

On the VSR-I, the IOM-V and CPM-V are deployed on a single VM.

8 VNF Manager web application

8.1 VNF Manager web application

8.1.1 Overview

The VNF Manager application is the interface used for the NFM-P as VNF manager solution. The application allows you to instantiate, maintain, and terminate VNFs managed by the NFM-P. The NFM-P provides an interface with the cloud management entity that provisions cloud resources. The NFM-P validates these resources and provides assurance and monitoring through the VNF Manager application and other features in the NFV solution.

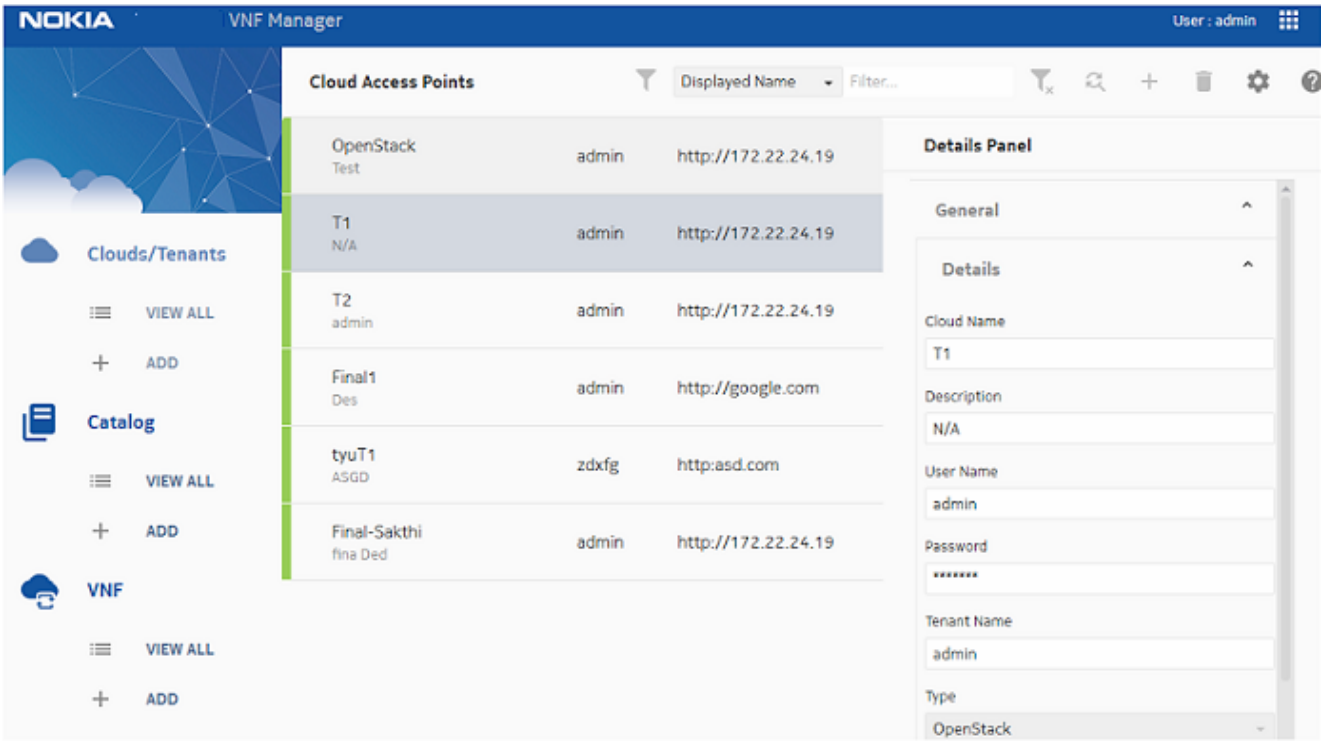
8.1.2 Product help tours

You can click on the menu in the application toolbar or on some panel toolbars to view a list of help tours. These tours are designed to explain the application features and provide workflows for completing management and monitoring tasks.

8.1.3 Application interface

The VNF Manager application interface is divided into three panels for managing catalogs, cloud access points, and VNFs.

Figure 11 VNF Manager application GUI



8.1.4 VNF catalogs

You can use the VNF Manager application to create and manage VNF catalogs. A VNF catalog is a collection of VNFs with a template that determines the type of VNF managed. The catalog includes deployment specifications, KPIs, and recipes for VNF management functions that are applicable to the VNF type. These become the default VNF settings that are defined when you instantiate a VNF using the catalog. You are able to configure these settings for specific VNFs during VNF instantiation.

You can create the following types of VNF catalogs:

- Generic
- VMG
- VMM
- VSR
- VSR-I

In addition to specifying the catalog type, you must also define a directory name. The directory name specifies the VNFD directory where the HOT files are located. The VNF

Manager automatically determines the catalog type and version based on the HOT files in the specified directory. You must configure a VNF catalog before you create a VNF object. See Chapter 10, “NFM-P VNF descriptor” for more information about template creation.

Generic VNF catalogs

While the VMG, VMM, and VSR catalog types include deployment settings and recipes specific to each NE type, the generic VNF catalog can be used for other NE types without node-specific settings. The onboarding, instantiation, and lifecycle management functions of generic VNF catalogs are the same as for the NE-specific catalogs. You must specify a VNFD directory and configure HOT template files for a generic VNF catalog.

8.1.5 Cloud access point

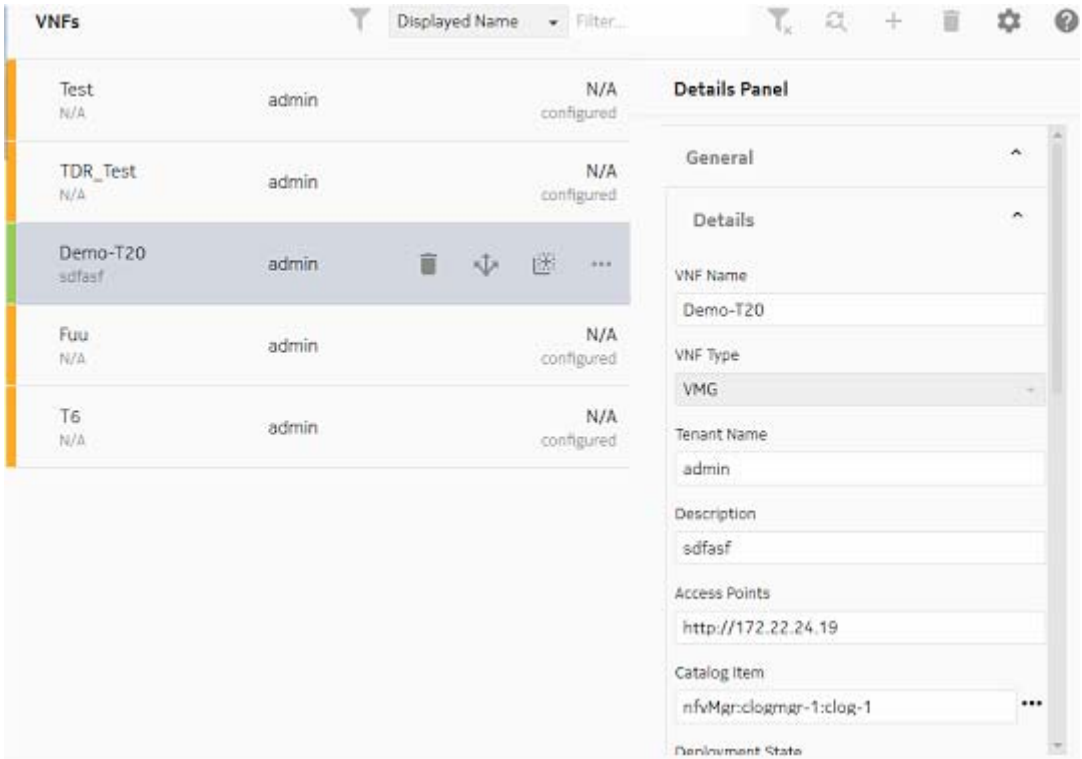
The VNF Manager application provides an interface with OpenStack for VNF management. You must define a cloud access point before you create a VNF object. You can use multiple cloud access points for VNFs under different tenants. For each cloud access point, you must provide login credentials for the tenant group and specify a Keystone URL. Specify an access URL, port number, and Keystone version in the format *http://127.227.135.61:5000/v2.0*.

The cloud access point defines parameters such as VNFC image/flavor and network /subnet ID, which are configurable during VNF instantiation.

8.1.6 VNF management

The VNF Manager application provides an at-a-glance view of VNFs and VNFCs in the network. VNF onboarding allows the application to archive, upload, and validate VNF software images with OpenStack Heat.

Figure 12 VNF Management



The following table describes the lifecycle management tasks that can be executed from the VNF Manager application.

Table 3 VNF lifecycle management tasks

Task	Description
Add	Instantiates a new VNF. You must choose a catalog and cloud access point when you create a new VNF.
Delete	Deletes a VNF.
Deploy	Deploys the VNF to OpenStack Heat. Newly instantiated VNFs are not automatically deployed. The Deployment State parameter shows whether the VNF has been deployed.

Table 3 VNF lifecycle management tasks (continued)

Task	Description
Scale-out	Increases processing capacity by adding a VM. You must specify the scaling template from the drop-down menu. The scale-out template defines the type and number of VNFCs to be added.
Scale-in	Decreases processing capacity by removing a VM. You must specify the scaling template from the drop-down menu. The scale-in template defines the type and number of VNFCs to be removed.
Sync	Synchronizes the VNF with OpenStack Heat.
Rescan Discovery Rule	Manually scan VNF discovery rules for new rule elements.
Reboot	Reboots the VNF component. The Reboot button is available from the VNF component list that opens when you select an VNF.

Manual scaling

CAUTION

Service Disruption

The manual scale-in function removes a VM without checking the node for subscribers. Nokia recommends you check the VM for subscribers and move them to another VM before using the scale-in function.

You can use the VNF Manager application to manually scale-in or scale-out to adjust the processing capacity of a VNF. When you use a scaling operation, you must select a scaling template to use. Each scaling template specifies a type and number of VNFCs to be scaled in or out. The scale-in and scale-out operations use the same scaling templates.

These scaling templates are defined by the *VNF_Type.userdef_chars*.grow.hot.yaml files in the grow sub-directory of the VNF catalog directory. The name of the scaling templates must specify the applicable VNF. A VMG scaling template must lead with *VMG* in the template name and a VMM scaling template must lead with *VMM*. For example, a scaling template designed to scale one load balancing VNF on a CMG might be named *CMG.1LB.grow.hot.yaml*. Nokia recommends that the template name specify the number and type of VNFCs to be scaled.

Note: The *userdef_chars* part of the template filename cannot contain a period.

For automatic scaling, the NFM-P looks for a file with the name VNF_ Type.default.grow.hot.yaml.

You can include a *grow_meta_data* entry in the meta file to specify which parameters are user-configurable during a VNF scaling operation in the VNF Manager application. For example, you can include *gw_subnet* in the *grow_meta_data* entry to allow the user to specify a subnet during VNF scale-out. See "VNF scaling parameterization" (p. 74).

8.1.7 VNF instantiation

The VNF Manager application uses VNF catalogs to instantiate VNFs on a cloud management system such as OpenStack. When you use the application to instantiate a VNF, you must specify a cloud access point which includes a cloud management system URL to which the NFM-P deploys the VNF. The VNF catalog selected during VNF instantiation defines the default settings that are required by the VNFD for lifecycle management operations. These settings are read from the catalog *.env.yaml* file. You can customize these settings for a specific VNF during instantiation, as required.

The cloud access point provides the list of available VNFC images and flavors. You can select the required images and flavors from drop-down menus during VNF instantiation.

You can also configure the System Address during VNF instantiation. These parameters uniquely identify the VNF. Alarms affecting the VNF or associated VNFCs list these parameters as System Name and Site ID.

Note: The instantiated VNF is not deployed until you click Deploy.

Note: The system address provided for the VNF should match the system address configured in the NE.

Initial configuration for CMG and VSR

During VNF instantiation, you can specify an initial configuration file for the CMG and VSR. Specify an FTP server address with the configuration file using the primary_config parameter. You must also specify the login credentials, system name, and SNMPv2 community string. The configuration file is uploaded to the file storage specified for the VNF instance. When the VNF is instantiated, it is automatically configured with the specifications in the initial configuration file.

Topology discovery rules

During VNF instantiation, you can specify a topology discovery rule for VSR, CMG, or VMM. If no discovery rules are defined, you can cross-launch the NFM-P Java GUI to create one. The NFM-P automatically creates a discovery rule element for the VNF once the VNF is successfully deployed. You can also manually trigger a discovery rule scan by clicking the Rescan Discovery Rule button. The ready-only Managed State identifies whether the VNF is managed by a discovery rule.

When the VNF is deleted, the NFM-P automatically removes the discovery rule element.

8.1.8 VNF component reboot

You can use the Reboot function to manually reboot a VNF component from the application GUI. To view a list of VNF components associated with a VNF, double-click the VNF. When you click the Reboot button, you can select a soft or hard reboot.

The read-only Task State parameter on the VNF component shows whether the component has any NFM-P component operations in progress. VNF component can be triggered only if the Task State parameter is Deployed. The read-only VM State parameter displays the OpenStack VM state. VNF component operations — such as reboot — can be performed only if the VM State parameter is Active, ShutOff, or Rescued.

8.1.9 VNF component evacuation

You can perform VNF component evacuation to move a VNF component to a new compute host. This functionality is useful if a VNF component goes down due to a compute host failure. This action does not reboot the VNF.

VNF component evacuation is available from the VNF Manager REST API only.

8.1.10 VNFD redundancy

The NFM-P supports redundancy for the HOT files that define the lifecycle management tasks performed by the VNF Manager application. These files are kept in the *<NFMP_INSTALL_DIR>/os* directory and require NFM-P administrator read/write privileges. Whenever an rsync operation is triggered, files in the directory with a *yaml, .txt, .sh,* and *.json* extension are duplicated into a standby file server. Changes in the *<NFMP_INSTALL_DIR>/os* directory on the active server are duplicated every 30 minutes by default.

8.1.11 REST API

The VNF Manager application publishes a set of URLs which point to resources, or web services, managed by them. The URLs that are available to users are documented. These URLs can be accessed through a browser by any authorized user, including OSSs which can use them to cross launch from their own application. To view the published URLs of a given application:

`http(s)://<host>/VNFManager/api-docs`

Where *host* is the hostname or IP address which hosts the application.

9 NFV use cases

9.1 Automatic scale-out and healing

9.1.1 Overview

You can configure the NFM-P to perform automatic scale-out and healing operations based on system-defined and user-defined processing thresholds and network events.

A scale-out operation increases the call processing capacity of a VNF. Processing capacity is increased by automatically creating and provisioning a VNFC. A healing operation reboots a VNFC. Automatic healing is triggered by the NFM-P when an alarm is raised against a VNFC. The NFM-P performs additional tasks based on the results of the attempted healing operation. You can enable automatic scale-out and automatic healing for the VMM or CMG. This configuration must be performed by a system administrator.

VMM automatic scale-out

You can enable automatic scale-out on the VMM and configure an automatic scale-out threshold. The automatic scale-out threshold defines the point at which an automatic scale-out operation can be triggered, where the threshold is the total UE capacity of the VMM multiplied by the scale-out factor. When the threshold is reached and/or one or more MAFs is affected by a resource overload node alarm, the NFM-P automatically increases the processing capacity of the VMM by creating an additional virtual MAF on the NE.

The Perform Scale-out (VMM) parameter defines the conditions under which the automatic scale-out is triggered. You can specify AND or OR for the following conditions:

- The number of UEs on the VMM is over capacity.
- At least one MAF cards has a threshold crossing alarm against the number of UEs.

The Auto Scale-out Timer parameter prevents the NFM-P from triggering additional scale-out operations while a previous scale-out is still in progress. You should specify a length of time that exceeds the expected time for a virtual MAF to become operational.

CMG automatic scale-out

Automatic scale-out on the CMG is triggered by user-defined thresholds. You can specify low and high threshold bearer limits by configuring a threshold group on the PDN gateway. When the bearer management threshold limit is reached, a threshold-crossing alarm is raised. This alarm triggers an automatic scale-out. The NFM-P automatically increases the processing capacity of the CMG by creating an additional auto-provisioned ISM MG Card on the NE.

CMG automatic healing

Automatic healing is triggered on the CMG when an LSS_hostReset or VNFC Down alarm is raised on a VNFC and the VM state is either Active, Shut Off, or Rescued. The NFM-P first attempts a soft reboot. If the alarm is not cleared, the NFM-P then attempts a hard reboot. If the alarm is still not cleared, the NFM-P raises a VNFCAutoHealFailed alarm on the VNFC.

The Auto Heal Timer specifies the length of time the NFM-P waits between reboot attempts before proceeding with the next action.

VMM automatic healing

Automatic healing is triggered on the VMM when an LSS_hostReset alarm is raised on a VNFC, the VM state is Active, and the deployed state is Deployed. The NFM-P first attempts a soft reboot. If the alarm is not cleared, the NFM-P then attempts a hard reboot. If the alarm is still not cleared, the NFM-P raises a VNFCAutoHealFailed alarm on the VNFC.

The Auto Heal Timer specifies the length of time the NFM-P waits between reboot attempts before proceeding with the next action.

Note: Automatic healing is not triggered if the LSS_hostReset alarm is raised on the active OAM VNFC.
Automatic healing is not triggered if the VMM is in MaintState,

Reboot status

You can see the status of the VNFC reboot from the VNF Manager application by viewing the Auto Heal State. The Auto Heal State is a real-time indicator that describes the current state of the automatic healing process. The Auto Heal State displays one of the following statuses:

- No Attempt — no automatic healing operation was attempted
- Soft Reboot — a soft reboot has resolved the alarm that triggered the automatic reboot
- Hard Reboot — a hard reboot has resolved the alarm that triggered the automatic reboot
- Reboot failed — the soft and hard reboot have failed to resolve the alarm that triggered the automatic reboot

9.2 To configure automatic scale-out or automatic healing

9.2.1 Purpose

Perform this procedure to configure automatic scale-out or automatic healing for the VMM or CMG. This procedure requires system administrator access rights.

9.2.2 Steps

Enable automatic scale-out or automatic healing

1

Choose Administration→System Preferences from the NFM-P main menu. The System Preferences form opens.

2

Click on the NFV tab and configure the required parameters for automatic scale-out or automatic healing.

3

If you are configuring automatic scale-out for the VMM, configure the scale-out Factor (VMM) (%), Auto scale-out Timer (VMM) (hours), and Perform Scale-out (VMM) parameters.

The scale-out Factor (VMM) (%) parameter defines the threshold that triggers the automatic scale-out operation, where the threshold is the UE capacity of the VMM multiplied by the scale-out factor.

The Auto scale-out Timer (VMM) (hours) parameter defines the minimum time that the NFM-P will wait before triggering an automatic scale-out operation while a previous scale-out operation is still in progress. You should specify a value that exceeds the expected time for a virtual MAF to become operational.

The Perform Scale-out (VMM) parameter defines the conditions under which the automatic scale-out is triggered. You can specify AND or OR for the following conditions:

- The number of UEs on the WMM is over capacity.
- At least one MAF cards has a threshold crossing alarm against the number of UEs.

4

If you are configuring automatic healing, configure the Auto Heal Timer (VMM) (min) and Auto Heal Timer (VMG) (min) parameters.

The Auto Heal Timer parameters specify the time that the NFM-P waits between reboot attempts before proceeding with the next action.

5

Save your changes and close the form.

For CMG automatic scaling, configure a threshold group

6

On the equipment tree, right-click on a CMG and choose Properties. The Network Element (Edit) form opens.

7

Choose a PGW and click Properties. The PDN Gateway (Edit) form opens.

8

Click on the Threshold Groups tab and click Create. The Threshold Group (Create) form opens.

9

Configure the required parameters.

Select Bearer Mgmt Limits for the Threshold Groups parameter.

Set the Administrative State parameter to Down before you create a threshold group counter.

10

Click on the Threshold Group Counters tab and click Create. The Threshold Group Counter (Create) form opens.

11

Configure the required parameters.

Select Number of Bearers for the Threshold Counter parameter.

12

Save your changes and close the forms.

END OF STEPS

10 NFM-P VNF descriptor

VNF descriptor

10.1 NFM-P NFV requirements

10.1.1 Overview

The VNF Descriptor is a package that describes the configuration of the VNF network. It consists of OpenStack Heat templates and a meta file. The Heat templates define VNF specifications, and the meta file creates custom parameters required by NFM-P NFV. This chapter describes the configuration requirements and recommendations for the Heat templates and meta file.

10.1.2 VNF Descriptor sample

You can view sample VNF Descriptor packages in the NFM-P server installation directory. The files are located in /opt/nsp/nfmp/server/nms/sample/NFV. There are sample VNFD sample packages for the CMG and VMM. Each sample package includes sample Heat templates and a sample meta file. You can use these samples to help build the templates and meta file with configurations required for your NFV network setup.

OpenStack Heat templates

10.2 Heat templates

10.2.1 Overview

The Heat orchestration engine is designed to start cloud applications based on template files that are specialized for different types of VNFs. These Heat orchestration template (HOT) files describe a list of OpenStack resources called a stack. OpenStack maps these resources to VMs based on VNF requirements. HOT files list the stack resources, and specify their type and configuration parameters. HOT files must update dynamically as the stack resources change due to lifecycle management operations.

The NFM-P uses HOT files to interact with OpenStack Heat. Whenever a stack resource is changed through the VNF Manager application, the NFM-P dynamically recreates the HOT file. The lifecycle management tasks that can be performed from the VNF manager application require specifications from pre-configured HOT files. Before you can use this functionality in the application, you must configure these HOT files with a format specific to NFM-P functionality. The default HOT file for a VNF is created from a generic node template. You can customize the deployment parameters using a Heat environment file.

This chapter describes the HOT file requirements for the VNF functions supported by the NFM-P.

Note: Basic Heat environment file requirements are not described in this chapter. This chapter describes only the additional requirements for NFM-P functionality. This chapter should be used in conjunction with OpenStack Heat template documentation.

10.2.2 NFM-P NFV template components

The NFM-P requires several HOT files to manage VNFs and communicate with OpenStack Heat. One of each of the following static template files are required per VNF type and release:

- base template
- custom resources templates
- sample environment
- growth unit template — required only for scaling operations

One of each of the following dynamic template files are required per VNF instance and lifecycle management event:

- deployment-specific base template
- custom resource template
- deployment-specific environment file
- meta file

10.2.3 Template archive

The NFM-P requires an archive for each VNF type (VMM, CMG, and VSR). This archive contains templates for initial deployment and, optionally, scaling.

For example, the *<NFMP_INSTALL_DIR>/os/VMG* archive for a VNF contains the following files:

- VMG.hot.yaml
- VMG.env.yaml
- VMG.env.meta.yaml
- VMG_LB.template.yaml
- VMG_OAMA.template.yaml
- VMG_MG.template.yaml
- VMG_OAMB.template.yaml
- *grow/*
  - VMG.1_LB_card + 1_VMG_card.grow.hot.yaml
  - VMG.2_VMG_card.grow.hot.yaml
  - VMG.default.grow.hot.yaml

The VNF folders in the *<NFMP_INSTALL_DIR>/os* must be created manually. The *.yaml* files must be in the directory that is specified in the VNF catalog.

Note: The *<NFMP_INSTALL_DIR>/os* directory and all template files must have nsp permissions.

10.3 Deployment template design

10.3.1 Overview

Each VNF must have one base template and a base environment file for each node release. These template files define VNF parameters and resources. If custom resources are defined, there must also be custom template files for each custom resource.

Base template — *x*.hot.yaml

The base template defines the Heat template version and description. It defines parameters that the user can configure during VNF instantiation. The base template lists basic OpenStack Heat resources and can optionally list custom resources.

Environment file — *x*.env.yaml

The environment file corresponds with the base template. It must contain all deployment-specific parameters that are defined in the base template. The environment file contains

a resource registry for custom resources, and includes file locations for the custom templates that define these resources. It must not contain any image or flavor names.

Custom resource template — *x*.template.yaml

The custom resource templates define custom resources. They must be located in the same file directory as the environment file that lists the custom resources under the resource registry. Custom resource templates should contain only native OpenStack Heat resource types.

10.3.2 Heat template version

The Heat template version must be specified at the top of the base and custom template files in the following format:

`heat_template_version:` *YYYY-MM-DD*

where *YYYY-MM-DD* is a Heat template version such as **`2014-10-16`**

10.3.3 Custom resources

Custom resources can be listed in the base template file. The resource types and file location must be included under the resource_registry in the environment file and then defined in separate template files.

Nokia recommends that all custom resource types follow the same naming convention, starting with *NOK::*. Custom resource types must be derived from basic OpenStack resources with no more than five levels of nested stacks (for example, *OS::Nova::Server*).

Consider the following when creating custom resources:

- Resource IDs must be unique. If two resources have the same name, OpenStack Heat will create only one of them. It will not produce any validation errors.
- Resource names for resources of type *OS::Nova::Server* must contain the Heat stack name as a prefix. This convention is required for some NFM-P monitoring functions.
- Resource names for resources of type *OS::Neutron::Port* and *OS::Cinder::Volume* should contain the Heat stack name as a prefix. This convention improves debugging and allows for quick association between resources and the Heat stack.

10.3.4 Parameters

The NFM-P parses the top level environment file that contains the deployment-specific parameters. Each parameter entry is translated into an input parameter that can be configured by an operator during VNF instantiation.

Consider the following when creating parameters:

- Parameters must be contained in a file with a *.env.yaml* extension.
- Parameter labels and comments appear in the VNF Manager application GUI during VNF instantiation. These appear to the operator as the parameter name and description. Use clear and descriptive labels and comments to explain the parameter functionality.
- Strings are preferred for operator input. Avoid using maps or lists, unless they are as simple as possible.
- Nokia recommends not defining image or flavor names as parameters in the environment file. Doing this will result in operators being required to have knowledge of image and flavor names. Image and flavor names should be defined in the template resources for the corresponding *OS::Nova::Server* resource. You can still achieve template reusability across VNF software releases by using the software version as a parameter in the environment file. You can then use image naming conventions that contain the prefix and the software version. See the following example.

—

```
image:
   str_replace
     template: LCP_MAF_$sw_version
  params:
     $sw_version: {get_param : sw_version}
```

Pre-defined parameter keywords

The NFM-P supports pre-defined keywords that can be used in parameter names. When these keywords are used, the NFM-P performs additional actions. These keywords can be used by any VNF type when defining parameter names in the environment file. The following table describes the pre-defined parameter keywords.

Keyword	Input type	NFM-P actions
auth_url	String	The NFM-P defines a read-only parameter with the authentication endpoint of the OpenStack tenant into which the VNF is being deployed.

Keyword	Input type	NFM-P actions
tenant_id	String	The NFM-P defines a read-only parameter with the tenant ID of the OpenStack tenant into which the VNF is being deployed.
tenant_name	String	The NFM-P defines a read-only parameter with the tenant name of the OpenStack tenant into which the VNF is being deployed.
username	String	The NFM-P defines a read-only parameter with the tenant username of the OpenStack tenant into which the VNF is being deployed.
password	String	The NFM-P defines a read-only parameter with the tenant password of the OpenStack tenant into which the VNF is being deployed.

10.3.5 VNF component slot ID

When the *OS::Nova::Server* is defined, you must include a *metadata* property. The *metadata* property tags the server with a slot and shelf ID for VNF components. This property stores arbitrary key/value server metadata and helps correlate domain-specific card alarms to VNF component alarms. See the following example.

—

```
type:
 OS::Nova::Serverproperties:
   metadata: {"nokia_vnf_slotId":{get_param:slot}}
```

10.4 Scaling template design

10.4.1 Overview

Scaling templates define the resources that are added or removed when a scaling operation is performed. For each VNF that supports scaling, there must be a scaling template. These templates define the number and type of VNF components to be scaled. They also describe all resources that are required for the scaling operation.

The same scaling template is used for multiple scaling operations. For example, a scale-out operation of three VNF components is derived from the scaling template for a single VNF component. The same scaling template is used for scale-in and scale-out operations.

The NFM-P manipulates the base deployment template to dynamically add or remove resources defined in the scaling template. There are some scaling template configuration requirements to facilitate this functionality.

10.4.2 Template naming

Scaling templates are defined by the *VNF_Type.userdef_chars*.grow.hot.yaml files in the grow sub-directory. The name of the scaling templates must specify the applicable VNF. A CMG scaling template must lead with *VMG* in the template name and a VMM scaling template must lead with *VMM*. Nokia also recommends that the template name specify the number and type of VNF components to be scaled. For example, a scaling template designed to scale one load balancing VNF on a CMG might be named *VMG.1LB.grow.hot.yaml*.

Note: The *userdef_chars* part of the template filename cannot contain a period.

For automatic scaling, the NFM-P looks for a file with the name VNF_ Type.default.grow.hot.yaml.

10.4.3 Template design

Consider the following when creating a scaling template:

- The template must include a resources section, as it would appear in the base deployment template.
- The template must not include any new parameter definitions. Any new parameters required must be defined in the environment file.
- Parameter and output sections are not required. The NFM-P is not able to dynamically update these sections, so they are not parsed.
- All resources that should be created together during a scale-in or healing operation (for example, server, ports, volume attachments, software deployments) must be grouped together in a single custom resource or in a *OS::Heat::ResourceGroup* resource.
- The NFM-P does not manipulate the outputs section of a scaling template.
- The Heat template version must be specified at the top of the base and custom template files in the following format:

  `heat_template_version: !!str` *YYYY-MM-DD*

  where *YYYY-MM-DD* is a Heat template version such as **`2014-10-16`**

The NFM-P supports some pre-defined properties that can be used as part of a custom resource in a scaling template. When these properties are used, the NFM-P performs additional actions.

resource_id

The resource_id property is a common prefix used for resources of the same type that are subject to horizontal growth. Each consecutive scaling operation adds a new instance of this resource. The NFM-P dynamically calculates the resource_id and name to be used when updating the Heat stack.

group_index

The group_index property is a list of properties defined in the scaling resource. The group_index is used to permit dynamic increments of a value of a common property for multiple stack resources. For example, group_index can be used for a VNF that requires a property called “Slot number” that uses an new increment with each scale-out operation.

When a scale-out operation is performed on a template where the group_index property is used, the NFM-P calculates the new incremental value of each property in the list based on the maximum value of the property with the same name among the resources already configured in the stack.

Strings in the group_index property must be listed in the following format:

[optional string][optional leading 0][number]

For example: *Card slot: 04*

group_index example

A stack contains *resourceA* of custom type *resA*. The custom type *resA* has a property called *myCardNumber* with a value of “05” in the initial deployment.

—

```
resources:
 resourceA:
   type: resA
  properties:
     deployment_prefix: {get_param: "OS::stack_name"}
     myCardNumber: "05"
```

—

If an operator requires a scale-out operation that adds one more resource of type *resA* with a new value for *myCardNumber*, the scaling template specifies the following:

—

```
resources:
 resourceA:
   type: resA
  properties:
     deployment_prefix: {get_param: "OS::stack_name"}
     group_index: ["myCardNumber"]
     myCardNumber: "anything"
```

—

When this scale-out operation is performed, the NFM-P dynamically creates the following HOT file. The resource name and *myCardNumber* value are incremented.

—

```
resources:
 resourceA:
   type: resA
  properties:
     deployment_prefix: {get_param: "OS::stack_name"}
     myCardNumber: "05"
 resourceA1:
   type: resA
  properties:
     deployment_prefix: {get_param: "OS::stack_name"}
     group_index: ["myCardNumber"]
     myCardNumber: "06"
```

—

dependency_group

The dependency_group property is a map of strings that defines related resources in the scaling template. A scaling template contains only the basic list of objects that need to be scaled, but the actual names of the resources being added depend on how many previous scaling operations there were in the lifecycle of the stack. The resource names are dynamically calculated during the scaling operation using the group_index property. The dependency_group property address the issue of a resource that refers to another resource during a scaling operating when the name of the resource is not yet known, having been updated due to previous scaling operations. When the dependency of a type is identified, the dependency_group property can be used to dynamically resolve resource names. See the following example:

—

```
dependency_group: {{propertyX: resourceY}}
```

—

For each entry in the dependency_group map, the NFM-P replaces the entry of *resourceY* with the actual calculated name of the resource with the prefix *resourceY* in the same scaling template.

dependency_group example

A stack has *resourceA* of custom type *resA* and *resourceB* of custom type *resB*. One of the properties of *resourceA* refers to *resourceB*.

—

```
resources:
 resourceA:
   type: resA
  properties:
     propertyXyz: {get_resources: resourceB}
 resourceB:
   type: resB
  properties:
    ...
```

—

After two scale-out operations, the NFM-P dynamically updates the resource names to accommodate for names that have been changed during scaling.

—

```
resources:
 resourceA1:
   type: resA
  properties:
     propertyXyz:
 {get_resources: resourceB1}
 resourceB1:
   type: resB
  properties:
    ...
```

—

```
resources:
 resourceA2:
   type: resA
  properties:
```

```
    propertyXyz: {get_resources: resourceB2}
 resourceB2:
   type: resB
  properties:
   ...
```

—

ResourceGroup example

All resources that should be created together during a scale-in or healing operation (for example, server, ports, volume attachments, software deployments) must be grouped together in a single custom resource or in a *OS::Heat::ResourceGroup* resource.

—

```
ALU-LCP-Pair%card%:
   type: OS::Heat::ResourceGroup
  dependency_group: {{property3: resource_idX}}
   properties:
    count: <<count>>
    resource_def:
     type: <customized resource type NOK::>
     properties:
      group_index: ["property1", "property2"]
 ...
   property1: somevalue1
  property2: somevalue2
  property3: {get_resource: resource_idX}
 ...
 <resource_idY>
   type: <customized resource type NOK::>
   properties:
    group_index: ["property1", "property2"]
 ...
   property1: somevalue1
  property2: somevalue2
  property3: {get_resource: resource_idX}
```

Meta file configuration and requirements

10.5 Meta file configuration

10.5.1 Overview

In order to improve usability and ensure OpenStack compatibility with certain NFM-P NFV functions, you should perform the following preconfiguration tasks while creating design templates.

- Create meta files for each VNF type.
- Define VNF management IDs.
- Define VNF component slot IDs.

10.5.2 Meta file design

A meta file is useful for the NFM-P to properly render the environment file and display the correct information to the user. The meta file improves usability and readability for VNF settings in the VNF Manager UI. For example, it allows you to display drop-down menus for properties such as flavor_map and subnets rather than plain text boxes.

You should create one meta file for each VNF type (VSR, VMM, and CMG). A meta file uses the extension *env.meta.yaml*.

The NFM-P retrieves the following resources from OpenStack resourcesduring VNF instantiation. Each of these resources should be described in a JSON format in the meta file. The nok_os_type uses one of these resources to retrieve the appropriate resources for the environment parameters.

- networks
- subnets
- images
- flavors
- volumes
- availabilityZoneInfo

Note: If a meta file is not present, parameter groupings and OpenStack resource retrieval are not available during VNF instantiation. The NFM-P displays the parameter keys and values as a simple label. Properties such as flavor and image maps appear as text boxes rather than drop-down menus.

Meta file keywords

The following keywords are specific to NFM-P NFV meta files:

- **nok_readwrite** — defines read/write attributes
- **nok_readonly** — defines read-only attributes

- **nok_os_type** — defines the key type in the environment file
- **nok_os_display_field** — specifies which OpenStack property is displayed to the NFM-P user
- **nok_os_ret_field** — specifies which OpenStack property is returned to the environment file
- **nok_hot_env_name** — defines the parameter name in the environment file

VNF scaling parameterization

You can include a *grow_meta_data* entry in the meta file to specify which OpenStack parameters can be configured by the user during a VNF scaling operation. For example, you can include *image* in the *grow_meta_data* entry to allow the user to select from a list of images during a VNF scale-out.

The following example shows a *grow_meta_data* entry including the *image* variable.

—

```
grow_meta_data:{
   image:{
      nok_os_type:images,
      nok_readwrite:{
         nok_os_display_field:name,
         nok_os_ret_field:id
      }
   }
}
```

—

Parameterization is supported for the following scale-out variables:

- flavor
- image
- network
- subnet
- availability zone

For each parameterized variable in the *grow_meta_data* entry, you must also include the variable in the resource template *.yaml* using the **$MKS_***variable***$** format. The following example shows the entry for an LB resource with parameterization for the *image* and *flavor* variables.

—

```
LB-%slot%:
    type: ALU::VMG::LB
```

```
    properties:
      stack_name: { get_param: "OS::stack_name" }
      instance: 1
      slot: 1
      group_index: [ "slot", "instance" ]
      gw_network:  { get_param: [EXTnet_info, gw_network, id] }
      gw_subnet:  { get_param: [EXTnet_info, gw_network, gw_
subnet, id] }
      internal_network: { get_param: [EXTnet_info, internal_
network, id] }
      mmeIf_network:  { get_param: [EXTnet_info, mmeIf_network,
id] }
      gwIf_network: { get_param: [EXTnet_info, gwIf_network,
id] }
      security_group: { get_param: security_group }
      image: $MKS_image$
      flavor: $MKS_flavor$
```

—

Meta file samples

The following example shows how to map a simple JSON object such as flavor_map where all of the keys present represent the same data type.

—

Environment file:

—

```
flavor_map: {
   "oam-flavor": m1.medium,
   "lb-flavor": m1.medium,
   "mg-flavor": m1.medium,
  }
imageA: " VMG_70S201"
```

—

Meta file:

—

```
flavor_map: {
    nok_os_type: flavors,
    nok_readwrite: {
```

```
      nok_os_display_field: name,
      nok_os_ret_field: id
    }
  }
imageA: {
    nok_os_type: images,
    nok_readwrite: {
      nok_os_display_field: name,
      nok_os_ret_field: id
    }
  }
```

—

The following example shows how to map networks and subnetworks into a meta file.

—

Environment file:

—

```
EXTnet_info: {
    pnf2net: {
      id: 52e6e28d-a7bf-4c57-a741-d42051011991,
      V4subnet1: {
        id: 99cf155f-6fb5-48a2-a4b4-552d0776c39c,
        cidr: 135.111.51.128/26,
        default_gateway: 135.111.51.190,
      },
   },
    snf3net: {
      id: 4e09c5bc-2290-4980-8d36-eec507f33e0e,
      V4subnet1: {
        id: 943fc247-dd9b-40ea-a869-de7d375a8f80,
        cidr: 172.16.0.0/16,
        default_gateway: 172.16.0.254,
      },
      V4subnet2: {
        id: a94b3443-9c81-4137-9127-6714d673aa28,
        cidr: 172.17.0.0/16,
        default_gateway: 172.17.0.254,
      },
  },
}
```

—

Meta file:

—

```
meta_data :
     env_meta_data: {
EXTnet_info: {
    nok_os_type: networks,
    nok_readwrite: [
      {
        nok_hot_env_name: id,
        nok_os_display_field: name,
        nok_os_ret_field: id
      }
    ],
    nok_subnet: {
      nok_os_type: subnets,
      nok_readonly: [
        {
          nok_hot_env_name: cidr,
          nok_os_display_field: cidr
        },
        {
          nok_hot_env_name: default_gateway,
          nok_os_display_field: gateway_ip
        }
      ],
      nok_readwrite: [
        {
          nok_hot_env_name: id,
          nok_os_display_field: name,
          nok_os_ret_field: id
        }
      ],
    }
  }
}
```

10.6 Other requirements

10.6.1 VNF management IP

For NFV functions, EMS functions, and alarm correlation, the NFM-P requires the management IP of the VNF when it is instantiated. The management IP must be provided in a key/value format in the parameters section of the environment file. See the following example.

—

```
parameters:
   oama_ip: 192.169.60.60
   oamb_ip: 192.169.60.61
```

—

In the above example, *oama_ip* and *oamb_ip* must be defined in the base template.

Note: If the VNF management IP is not defined, the VNF cannot be instantiated.

10.7 To enable OpenStack message logging

10.7.1 Purpose

You may need to view OpenStack request/response message logs to troubleshoot possible communication issues. To enable OpenStack message logging on the NFM-P, you must modify the nms-server.xml file.

After performing this procedure, OpenStack message logs can be found in the following folder:

/opt/nsp/nfmp/server/nms/log/NFV_debug

Other NFV messages are logged in the NFV.log file:

/opt/nsp/nfmp/server/nms/log/server/NFV

10.7.2 Steps

1

Log in to the main server station as the nsp user.

2

Open the /opt/nsp/nfmp/server/nms/config/nms-server.xml file using a plain-text

Note: Contact your Nokia technical support representative before you attempt to
modify the nms-server.xml file. Modifying the nms-server.xml file can have serious
consequences that can include service disruption.

3

Add the following line:

```
<systemNFVApiLog enabled="yes" path="../log/NFV_debug"/>
```

4

Run /opt/nsp/nfmp/server/nms/bin/nmsserver.bash read_config .

5

View message logs in the following folder:

/opt/nsp/nfmp/server/nms/log/NFV_debug

END OF STEPS