Monitoring Virtual Network Functions

- Monitoring the VNFs, page 1
- Monitoring Methods, page 7
- Monitoring a VM, page 8
- Monitoring the Health of ESC, page 9
- Monitoring Operations, page 11

Monitoring the VNFs

After deploying VNFs, they are monitored periodically to check their health and workload. Monitoring is based on the definition of metrics within the KPI section of the deployment data model. As described in the KPIs section the metric type determined not only the variable to monitor, but also the collector action to be executed. ESC allows you to define the metrics to be monitored and the actions that needs to be executed when the conditions are met. These metrics and actions are defined in the deployment datamodel. Several monitoring methods are used to monitor the VNFs. You can monitor the following:

- VM aliveness
- VM variables for Disk usage, Memory, CPU, Network throughput
- ICMP message on the VM monitoring interface

Pre-requisites for Monitoring

The following pre-requisites must be met for the VMs to be monitored by ESC:

- Monitoring is enabled for VMs that are successfully deployed. The deployed VMs must be alive.
- KPI must be configured in the data model with the monitoring parameters.

Monitoring and Action Execution Engine

Monitoring is based on the definition of metrics within the KPI section of the deployment datamodel. As described in the KPIs section the metric type determines not only the variable to monitor, but also the collector action to be executed. The monitoring engine comprises of metrics and actions.
1 Metrics
2 Actions

The metrics and actions <metadata> section describes the properties or entries controlling the programmable aspect of the engine.

**Metrics Section**

The metrics section is as follows:

```xml
<metrics>
  <metric>
    <name>{metric name}</name>
    <type>{metric type}</type>
    <metaData>
      <type>{monitoring engine action type}</type>
      <properties>
        <property>
          <name></name>
          <value></value>
        </property>
        : : : : :
      </properties>
    </metaData>
  </metric>
  : : : : :
</metrics>
```

**Table 1: Metric Section Description**

<table>
<thead>
<tr>
<th>Tag name</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A user defined metric name. The metric name must be unique.</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Dynamic mapping supported type.</td>
<td>MONITOR_SUCCESS_FAILURE MONITOR_THRESHOLD MONITOR_COMPUTE_THRESHOLD</td>
</tr>
</tbody>
</table>

**Metric Metadata Section**

The purpose of the metadata section is to provide information specific to the monitoring solution.

**Table 2: Metric Metadata Section**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>The action type, values are a one to one mapping with MONA supported actions.</td>
<td>custom_script custom_script_threshold snmp_get_threshold</td>
</tr>
</tbody>
</table>
Values

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>properties</td>
<td>A container for a list of properties (name/value) that will be passed to selected action. The properties are defined by the list of expected monitoring and actions attributes.</td>
<td>Properties are based on the selected action type.</td>
</tr>
</tbody>
</table>

### Actions Section

The actions section is as follows:

```xml
<actions>
  <action>
    <name>{action name}</name>
    <type>{action type}</type>
    <metaData>
      <type>{monitoring engine action type}</type>
      <properties>
        <property>
          <name></name>
          <value></value>
        </property>
        : : : : : :
      </properties/>
    </metaData>
  </action>
  : : : : : :
</actions>
```

### Table 3: Actions

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A user defined action name. The action name must be unique.</td>
<td>One of the main requirements is also to have the chosen name prefixed with TRUE or FALSE to allow mapping between ESC data model rule and dynamic actions, just for MONITOR_SUCCESS_FAILURE.</td>
</tr>
</tbody>
</table>

| type     | Supported type. | ESC_POST_EVENT SCRIPT CUSTOM_SCRIPT |

### Actions Metadata Section

The purpose of the metadata section is to provide information specific to the monitoring solution.
### Table 4: Action Metadata section

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>The action type, values are a one to one mapping with monitoring and actions engine supported actions.</td>
<td>icmp_ping, icmp4_ping, icmp6_ping, esc_post_event, script, custom_script, snmp_get, snmp_get_threshold</td>
</tr>
<tr>
<td>properties</td>
<td>A container for a list of properties (name/value) that will be passed to selected action. The properties are defined by the list of expected monitoring and action attributes.</td>
<td>Properties are based on the selected action type.</td>
</tr>
</tbody>
</table>

For more details see the KPIs, Rules and Dynamic Mapping APIs section.

### Table 5: Supported Action Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Properties and their description</th>
</tr>
</thead>
</table>
| icmp_ping | • ip_address  
            • enable_events_after_success: Boolean controlling when MONA will start forwarding events notifications. If set to true, notification will be forwarded only after the first transition to success.  
            • timeOut: Default set to 5 seconds |
| icmpv4_ping | • ip_address  
            • enable_events_after_success: Boolean controlling when MONA will start forwarding events notifications. If set to true, notification will be forwarded only after the first transition to success.  
            • timeOut: Default set to 5 seconds |
<table>
<thead>
<tr>
<th>Type</th>
<th>Properties and their description</th>
</tr>
</thead>
</table>
| icmpv6_ping        | • ip_address  
|                   | • enable_events_after_success: Boolean controlling when MONA will start forwarding events notifications. If set to true, notification will be forwarded only after the first transition to success.  
|                   | • timeOut: Default set to 5 seconds                                                                 |
| script            | • script_filename: Full path to the script to be executed (The script has to be located on the ESC VM).  
|                   | • wait_for_script: Boolean controlling if the action is waiting for the completion of the script. (Not actually exercised) |
| custom_script     | script_filename: Full path to the script to be executed (The script has to be located on the ESC Manager VM). |
| custom_script_threshold | • script_filename: Full path to the script to be executed (The script has to be located on the ESC Manager VM).  
|                   | • threshold                                                                                      |
| post_esc_event    | • esc_url  
|                   | • vm_external_id  
|                   | • vm_name  
|                   | • esc_event  
|                   | • event_name                                                                                     |
| snmp_get          | • target_oid, agent_address, IP Address of the SNMP agent (IPV4/IPV6 is supported)  
|                   | • agent_port: Port used by the SNMP Agent.  
|                   | • agent_protocol: Protocol used by the SNMP Agent (tcp/udp).  
<p>|                   | • Community: SNMP v2c community string used by the SNMP agent                                       |</p>
<table>
<thead>
<tr>
<th>Type</th>
<th>Properties and their description</th>
</tr>
</thead>
<tbody>
<tr>
<td>snmp_get_threshold</td>
<td>• target_oid: Object Identifier that will be used for the threshold comparison.</td>
</tr>
<tr>
<td></td>
<td>• agent_address: IP Address of the SNMP agent (IPV4/IPV6 is supported).</td>
</tr>
<tr>
<td></td>
<td>• agent_port: Port used by the SNMP Agent.</td>
</tr>
<tr>
<td></td>
<td>• agent_protocol: Protocol used by the SNMP Agent (tcp/udp).</td>
</tr>
<tr>
<td></td>
<td>• community: SNMP v2c community string used by the SNMP agent.</td>
</tr>
<tr>
<td>snmp_get_threshold_ratio</td>
<td>• oid_total_value: Object Identifier that will be used to represent the current for the ratio/percentage computation.</td>
</tr>
<tr>
<td></td>
<td>• oid_current_value: Object Identifier that will be used to represent the current for the ratio/percentage computation. We are currently supporting two algorithms: COMPUTE_TOTAL_CURRENT_BASED, COMPUTE_TOTAL_AVAILABILITY_BASED.</td>
</tr>
<tr>
<td></td>
<td>• agent_address: IP Address of the SNMP agent (IPV4/IPV6 is supported).</td>
</tr>
<tr>
<td></td>
<td>• agent_port: Port used by the SNMP Agent.</td>
</tr>
<tr>
<td></td>
<td>• agent_protocol: Protocol used by the SNMP Agent (tcp/udp).</td>
</tr>
<tr>
<td></td>
<td>• community: SNMP v2c community string used by the SNMP agent.</td>
</tr>
</tbody>
</table>

**Properties and Runtime Parameter Injection**

The properties list passed to the selected action type supports the capabilities to automatically inject runtime value for some selected parameters. For example, runtime value of the virtual machine ip_address or its name can be passed automatically as arguments to the selected action.

Following are some of the parameters that can be passed to the scripts at the time of execution. Parameter value is populated at runtime only if

- the parameter is a supported one, and
- its value is empty within the dynamic-mappings.xml file.

Otherwise, the value defined within the script is passed as is.
Table below shows the parameters passed during runtime.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>esc_url</td>
<td>The URL of the Elastic Services Controller.</td>
</tr>
<tr>
<td>vm_external_id</td>
<td>The external id of the managed VM.</td>
</tr>
<tr>
<td>vm_name</td>
<td>The name of the managed VM.</td>
</tr>
<tr>
<td>vm_mac_address</td>
<td>The mac address of the managed VM.</td>
</tr>
<tr>
<td>vm_external_host_id</td>
<td>The VM external host Identifier.</td>
</tr>
<tr>
<td>vm_external_host_name</td>
<td>The VM external host name.</td>
</tr>
<tr>
<td>vm_group_name</td>
<td>The VM group name.</td>
</tr>
<tr>
<td>ip_address</td>
<td>The VM IP Address.</td>
</tr>
<tr>
<td>event_name</td>
<td>The ESC event name.</td>
</tr>
</tbody>
</table>

The properties list passed to the selected action, is not bound by the parameters in the action type. A script designer can define its own parameters. However, the values have to be provided.

---

**Monitoring Methods**

ESC uses several monitoring methods to monitor the VNFs. You must configure the KPI data model for the monitoring methods.

**ICMP Ping Monitoring**

Ping monitoring assess the liveliness or reachability of a VNF.

If a VM is unreachable the healing of the VM is triggered. At every defined interval, ESC polls the metric value and sends alarms whenever needed. The number of polls, metric value, and other configuration are set in the KPI datamodel.

**SNMP Monitoring**

In SNMP Monitoring, load of the VM such as memory usage and CPU in a given period is monitored. The SNMP Get operation is used to assess the liveliness or reachability of a VNF. In this monitoring method, only the success or failure is monitored.

**SNMP Threshold Monitoring**

In SNMP threshold monitoring, you can set the upper and lower threshold levels in the kpi section of the data model. Actions are performed based on the upper and lower threshold levels.
**Custom Monitoring**

In ESC 2.1 or earlier, the Dynamic Mapping XML is required to map the actions and metrics defined in the datamodel to the valid actions and metrics available in the monitoring agent. The file is stored on the ESC VM and is modified using a text editor. This method is error prone and modification for an HA pair requires to take place on both the primary and secondary VMs. ESC 2.2 or later does not have an esc-dynamic-mapping directory and dynamic_mappings.xml file. The CRUD operations for mapping the actions and the metrics is now available through REST API in ESC. For more information, see KPIs, Rules and Dynamic Mapping APIs.

---

**Monitoring a VM**

Cisco Elastic Services Controller monitors the VM to detect any erroneous condition. ESC uses one of its monitoring methods to detect actions on a VM, and passes this information to the rules service for processing. The monitoring request comes from the northbound client along with VNF deployment requests.

There are two sections in the datamodel xml file which define the events and rules: KPI and Rule.

Based on the monitors and actions, rules are triggered.

```
<kpi>
  <event_name>VM_ALIVE</event_name>
  <metric_value>50</metric_value>
  <metric_type>UINT32</metric_type>
  <metric_occurrences_true>3</metric_occurrences_true>
  <metric_occurrences_false>3</metric_occurrences_false>
  <metric_collector>
    <type>ICMPPing</type>
    <nicid>0</nicid>
    <poll_frequency>15</poll_frequency>
    <polling_unit>seconds</polling_unit>
    <continuous_alarm>false</continuous_alarm>
  </metric_collector>
</kpi>
```

In the example above, an event is sent to check whether the VM is alive. The VM is pinged at regular intervals, and based on the result VM_ALIVE event is sent to the rules engine along with the details of the VM.

The rules engine receives events from the monitoring engine. The rules engine can handle simple to complex events. Based on the event received an action is triggered.

If the VM is not alive, based on the event the actions defined in the <rule> section are triggered. This can be found in the dep.xml datamodel.

```
<rules>
  <admin_rules>
    <rule>
      <event_name>VM_ALIVE</event_name>
      <action>ALWAYS log</action>
      <action>FALSE recover autohealing</action>
      <action>TRUE servicebooted.sh</action>
    </rule>
  </admin_rules>
</rules>
```

The rules section describes the actions to be executed once a monitoring event has been detected. The dynamic mapping API drives the rules based on keywords.

In the above example, the following actions are performed based on the given condition:

- **ALWAYS log**: Whether the event is pingable or not, the details are logged.
TRUE servicebooted.sh: The action identified by this keyword in the dynamic mapping API is triggered when the VM moves from a non-pingable to a pingable state. The serviceboot script informs ESC that the VM is “alive” allowing it to transition the VMs state.

FALSE recover autohealing: The action identified by this keyword will be triggered and the VM will be recovered without the administrator's intervention.

Monitoring log files for troubleshooting are available at /var/log/mona.

## Monitoring the Health of ESC

ESC provides REST API for any third party software to monitor the health of ESC and its services. Using the API, the third party software can query the health condition of ESC periodically to check whether ESC is in service. In response to the query, API provides status code and messages, see Table 6: ESC Health API Status Code and Messages, on page 9 for details. In an HA setup the virtual IP (VIP) must be used as the monitoring IP. The return value provides the overall condition of the ESC HA pairs. See the Table 7: Health API Status Messages for Standalone ESC and HA, on page 10 for details.

The REST API to monitor the health of ESC is as follows:

```
GET to https://<esc_vm_ip>:60000/esc/health
```

The monitoring health API response is as follows:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<esc_health_report>
  <status_code>2000</status_code>
  <message>ESC services are running</message>
</esc_health_report>
```

XML and JSON responses are also supported for the monitoring health API.

The status code and messages below provide the health condition of ESC. The status codes with 2000 series imply that the ESC is operational. The status codes with 5000 series imply that at least one ESC component is not in service.

### Table 6: ESC Health API Status Code and Messages

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>ESC services are running.</td>
</tr>
<tr>
<td>2010</td>
<td>ESC services are running, but ESC High Availability node is not reachable.</td>
</tr>
<tr>
<td>2020</td>
<td>ESC services are running. One or more VIM services (for example, keystone and nova) not reachable.</td>
</tr>
<tr>
<td>2030</td>
<td>ESC services are running, but VIM credentials are not provided.</td>
</tr>
</tbody>
</table>
ESC services are running, but ESC High-Availability node is not reachable. One or more VIM services (for example, nova) are not reachable.

ESC service, ESC_MANAGER is not running.

ESC service, CONFD is not running.

ESC service, MONA is not running.

ESC service, VIM_MANAGER is not running.

More than one ESC service (for example, confd and mona) are not running.

ESC HA mode refers to ESC HA in DRBD setup only. For more information on the ESC HA setup, see the Cisco Elastic Services Controller Install Guide.

The table below describes the status message for standalone ESC and HA with success and failure scenarios. For more information on ESC standalone and HA setup, see the Cisco Elastic Services Controller Install Guide.

**Table 7: Health API Status Messages for Standalone ESC and HA**

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2100</td>
<td>ESC services are running, but ESC High-Availability node is not reachable. One or more VIM services (for example, nova) are not reachable.</td>
</tr>
<tr>
<td>5010</td>
<td>ESC service, ESC_MANAGER is not running.</td>
</tr>
<tr>
<td>5020</td>
<td>ESC service, CONFD is not running.</td>
</tr>
<tr>
<td>5030</td>
<td>ESC service, MONA is not running.</td>
</tr>
<tr>
<td>5040</td>
<td>ESC service, VIM_MANAGER is not running.</td>
</tr>
<tr>
<td>5090</td>
<td>More than one ESC service (for example, confd and mona) are not running.</td>
</tr>
</tbody>
</table>

**Note**

The response is collected from the monitoring health API and the status code is 2000.

NA

The response is collected from the monitoring health API and the status code returned is in the 5000 series.

- Monitor cannot get the response from the monitoring health API.
- The response is collected from the monitoring health API and the status code returned is in the 5000 series.
### Monitoring Operations

You can set and unset monitoring of VMs using RESTful interface.

A payload is required to monitoring VMs:

```
POST ESCManager/v0/{internal_tenant_id}/deployments/vm/{vm_name}
```

Example,

```
<?xml version='1.0' encoding='UTF-8'?>
<vm_operation xmlns='urn:ietf:params:xml:ns:netconf:base:1.0'>
  <operation>enable_monitoring</operation>
  <force>false</force>
</vm_operation>
```
You must mention enable_monitoring to set VM monitoring, and disable_monitoring to unset VM monitoring in the operation field.