Healing Virtual Network Functions

Healing Virtual Network Functions Using ETSI API

As part of life cycle management, ESC heals the VNFs when there is a failure. The recovery policy specified during deployment controls the recovery. ESC supports recovery using the policy-driven framework, see Configuring a Recovery Policy Using the Policy-driven Framework in the Cisco Elastic Services Controller User Guide.

The healing parameters define the behavior that is monitored to trigger a notification to heal a VNF. These parameters are configured in the KPI section of each compute node in the VNFD along with rules. The rules define the action to be taken (including events that are triggered) as a result of these KPI conditions to heal a VNF.

ESC ETSI configures monitoring using the following two sections:

- kpi_data—defines the type of monitoring, events, polling interval and other parameters
- admin_rules—defines the actions when the KPI monitoring events are triggered

Example:

```
vdu1:
  type: cisco.nodes.nfv.Vdu.Compute
  properties:
    name: Example VDU1
    description: Example VDU
    ...
    kpi_data:
      VM_ALIVE-1:
        event_name: 'VM_ALIVE-1'
        metric_value: 1
        metric_cond: 'GT'
        metric_type: 'UINT32'
        metric_occurrences_true: 1
        metric_occurrences_false: 30
        metric_collector:
          type: 'ICMPPing'
          nicid: 1
          poll_frequency: 10
```

```
        polling_unit: 'seconds'
        continuous_alarm: false
    admin_rules:
      VM_ALIVE-1:
        event_name: 'VM_ALIVE-1'
        action:
          - 'ALWAYS log'
          - 'FALSE recover autohealing'
          - 'TRUE esc_vm_alive_notification'
    ...
```

This example shows the default KPI and rule to support the service alive notification required to complete the deployment in ESC. For more information on KPI, rules, and the underlying data model that is exposed in the VNFD, see KPIs, Rules and Metrics in the Cisco Elastic Services Controller User Guide.

There are three types of actions for recovery when an event denoting that an instance requires attention is received, a timer expires or a manual recovery request is received; the healing workflow will:

- REBOOT_THEN_REDEPLOY—first attempt to reboot the affected VNFCs; if this fails, then it attempts to redeploy the affected VNFCs (on the same host)
- REBOOT_ONLY—only attempt to reboot the VM
- REDEPLOY_ONLY—only attempt to redeploy the VM

The recovery policy is configured at a VNF-level, and applies to each VNFC contained within. The monitoring agent monitors each VNFC and when a recovery situation arises, the message is converted to an alarm and sent to any subscribed consumers (e.g. an NFVO or Element Manager).

If autoheal is *enabled* on the VNF instance, then ESC automatically attempts to recover the VNF based on the recovery policy configured on deployment. This may be configured in the VNFD or alternatively modified against the VNF instance prior to instantiation.

The recovery of the VNF is to request action against the affected VNFCs. If the service fails to deploy, then the lifecycle management operation fails, if ESC cannot manage to recover the service using the defined policy after the initial deployment operation times out.

To modify the autoheal flag (*isAutohealEnabled*) VNF instance resource, see Modifying Virtual Network Functions.

If autoheal is *not enabled*, only the alarm is dispatched to all the subscribers. The subscriber can initiate a manual HealVnfRequest. The data structures are available for any VNF specific actions. There are no mandatory parameters.

Example for *SOL003*:

```
Request Payload (ETSI data structure: HealVNFRequest)
POST /vnf_instances/{vnfInstanceId}/heal
{
    "cause": "b9909dde-e21e-45ec-9cc0-9e9ae413eee0",
}
```

Example for *SOL002*:

```
POST /vnf_instance/{vnfInstanceId}/heal
 {
   "vnfcInstanceId": ["b9909dde-e21e-45ec-9cc0-9e9ae413eee0"],
   "cause": "b9909dde-e21e-45ec-9cc0-9e9ae413eee0",
   "healScript": "REBOOT_ONLY"
 }
```

The healScript is implemented as an enumeration of the valid recovery policy names which allow the policy configured in the deployment data model to be overriden. The list of vnfcInstanceIds allow the required VNFCs to be affected, however the absence of this list means the request applies to the entire VNF.

Additional parameters can be used to specify an overriding recovery policy, regardless of the policy configured at the time of deployment.

The recovery policy can be specified at VNFC level using additional parameters. This will override the values set at the VNF level. If the recovery policy is not specified at VNFC level, then ESC will inherit the properties from the VNF level recovery policy.

An optional additional parameter is added to the cisco.datatypes.nfv.VnfcAdditionalConfigurableProperties data type to support VNFC level recovery.

```
cisco.datatypes.nfv.VnfcAdditionalConfigurableProperties:
  derived_from: tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties
  properties:
    ...
    is_vnfc_autoheal_enabled:
      type: boolean
     description: It permits to enable (TRUE)/disable (FALSE) the auto-healing functionality.
 If the properties is not present for configuring, then VNF-level property is used instead
      required: false
    recovery_action:
      type: string
      required: false
      constraints:
        - valid_values: [ REBOOT_THEN_REDEPLOY, REDEPLOY_ONLY, REBOOT_ONLY ]
```

For information on monitoring, see Monitoring Virtual Network Functions Using ETSI API.

Updating an Existing Deployment During Healing

After a deployment is created successfully, the resources within it can be updated. As part of deployment management, you can add or remove resources, or update the configuration of the existing resources. These updates can be carried out in a running deployment. The resources are updated as part of the recovery process.

You can update an existing deployment (provisioned through the ETSI NFV MANO API) during the healing workflow. During the Heal request, the existing image and Day-0 parameters are compared and updated to the new ones provided as part of a subsequent Heal request.

The healing workflow allows:

- Updating the deployment model with the new image and Day-0 configuration
- Re-applying new or existing configuration data to the VNFC when healing with an upgraded image

Note You must redeploy the VNF after any update to the data model *if* the change is not carried out directly on the VIM.

After supplying new *additionalParams* via the HealVnfRequest, if the Grant response (from the NFVO) also supplies a new image or new additionalParams, this would also trigger a service update.

If the NFVO determines that the deployment should be moved as part of a redeployment, then the Grant provides a new *zoneId* to reflect the new placement of the resources.

The recovery action takes place after the service update is complete. In the event of a redeploy, it considers the up-to-date deployment model to ensure that any deployed updates are not reverted.

The following example shows the details NFVO returns in the Grant to trigger a service update with new *additionalParams* and/or a new *vimSoftwareImageId*.

Example:

```
{
  "headers" : {
    "Content-Type" : [ "application/json" ],
    "Location" : [
"http://{nfvoApiRoot}/sol003/default/grant/v1/grants/38ba2103-dab3-450e-992b-ee85aad6c899"
 ],
    "Content-Length" : [ "22935" ],
  },
  "body" : {
    "id" : "38ba2103-dab3-450e-992b-ee85aad6c899",
    "vnfInstanceId" : "6aaf527c-0093-49c3-ba2e-49fc6d8a4f71",
    "vnfLcmOpOccId" : "cdc5d9b3-81a0-400b-a4d9-97d1b3e117d9",
    "_links" : {
      "self" : {
        "href" :
"http://{nfvoApiRoot}/sol003default/grant/v1/grants/38ba2103-dab3-450e-992b-ee85aad6c899"
      },
      "vnfLcmOpOcc" : {
        "href" :
"https://{vnfmApiRoot}/vnflcm/v1/vnf_lcm_op_occs/cdc5d9b3-81a0-400b-a4d9-97d1b3e117d9"
      },
      "vnfInstance" : {
        "href" :
"https://{vnfmApiRoot}/vnflcm/v1/vnf_instances/6aaf527c-0093-49c3-ba2e-49fc6d8a4f71"
      }
    },
    "vimConnections" : [ {
      "id" : "myVimConnection",
      "vimType" : "OPENSTACK_V3",
      "vimId" : "595b0bc2-8dad-4087-abdf-ebe3b0b14d96",
      "interfaceInfo" : {
        "endpoint" : "https://{vimApiRoot}/v3"
      },
      "accessInfo" : {
        "password" : "********",
        "project" : "cisco",
        "projectDomain" : "demo",
        "region" : "RegionOne",
        "userDomain" : "demo",
        "username" : "********"
      }
    } ],
    "zones" : [{
      "id" : "1773873a-ab15-4a7b-b024-bc338425ed24",
      "zoneId" : "nova"
    },{
      "id" : "1773873a-ab15-4a7b-b024-bc555555ed55",
      "zoneId" : "nova2"
    }],
    "addResources" : [{
      "resourceDefinitionId" : "res-a6252dbf-b418-4f88-b8a9-14d8f3942938",
      "vimConnectionId" : "myVimConnection",
```

```
      "zoneId" : "1773873a-ab15-4a7b-b024-bc555555ed55"
    }],
    "vimAssets" : {
      "softwareImages" : [ {
        "vnfdSoftwareImageId" : "s3",
        "vimSoftwareImageId" : "3a609da7-e2b2-4e27-91b6-7bcabe902820",
        "vimConnectionId" : "myVimConnection"
      }, {
        "vnfdSoftwareImageId" : "s4",
        "vimSoftwareImageId" : "3a609da7-e2b2-4e27-91b6-7bcabe902820",
        "vimConnectionId" : "myVimConnection"
      } ]
    }
  },
  "additionalParams": [
    ...
    /* changed additionalParams */
    "CF_VIP_ADDR": "10.123.23.4",
    "SF_VIP_ADDR": "10.123.24.4",
    ...
  ],
  "statusCode" : "CREATED",
  "statusCodeValue" : 201
}
```

For more information on healing, see Healing Virtual Network Functions Using ETSI API, on page 1.