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, for more information, 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 with rules. The rules define the action as a result of these KPI conditions to heal a VNF.

The ETSI VNFM 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
    ...
     configurable_properties:
      additional_vnfc_configurable_properties:
        vim_flavor: { get_input: VIM_FLAVOR }
        bootup_time: { get_input: BOOTUP_TIME }
        vm_name_override: { get_input: VDU1_VM_NAME}
        recovery_action: REBOOT_THEN_REDEPLOY
        recovery_wait_time: 1
        kpi_data:
          VM_ALIVE-1:
            event_name: 'VM_ALIVE'
            metric_value: 1
            metric_cond: 'GT'
```

```
      metric_type: 'UINT32'
      metric_occurrences_true: 1
      metric_occurrences_false: 30
      metric_collector:
        type: 'ICMPPing'
        nicid: 1
        address_id: 0
        poll_frequency: 10
        polling_unit: 'seconds'
        continuous_alarm: false
admin_rules:
  VM_ALIVE:
    event_name: 'VM_ALIVE'
    action:
      - 'ALWAYS log'
      - 'FALSE recover autohealing'
      - 'TRUE esc_vm_alive_notification'
```

The previous 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.

The recovery of the VNF is to request action against the affected VNFCs determined by the recovery policy defined during the initial deployment or in the recovery request.

There are four 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 by default uses the recovery policy configured at either the VNF-level or at the VNFC-level within the VNFD. The supported policies are:

- 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
- RESET_THEN_REBOOT—reset the state of the VM (Openstack only) and then attempt to reboot the VM
- REDEPLOY_ONLY—only attempt to redeploy the VM

If the recovery policy is configured at a VNF-level, the policy applies to each constituent VNFC. If it is specified at VNFC-level, then that policy prevails. 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).

The HealVnfRequest contains a *cause* parameter that triggers different behaviors within the VNFM while processing the recovery request. If the *cause* is one of the values supported by the VNFM (and listed in the VNFD for the deployment as a supported cause) then certain *additionalParams* keys are activated to support the desired recovery action, as mentioned in the following table. If the NFVO supports the *cause*, the grant receives the *additionalParams* and allows the inputs to be modified before executing the recovery request.

If the *cause* is not one of the overriding causes supported by ESC, then it is assumed that the value provided is simply metadata and ignored; the VNFM would then use the recovery policy configured at the time of deployment. If the cause is supported by ESC, but not listed in the VNFD, then the request is rejected.

Table 1: HealVnfRequest causes

Cause	additionalParams keys	Recovery behavior

`APPLICATION_FAILURE`	*Optional:*
	`vnfcInstanceId`
		The recovery attempts to reboot the entire VNF unless `vnfcInstanceId` is populated with a list of valid identifiers for VNFC instance(s) which constrains the recovery to those VNFCs only. For example:
		`{`
		`...`
		`"vnfcInstanceId": [`
		`"resId1",`
		`"resId2"`
		`]`
		`...`
		`}`

`VIRTUALISATION_FAILURE`	*Optional:*
	`vnfcInstanceId`
	`resourceId`
	`virtualStorageDescId`
		The treatment of the `vnfcInstanceId` is as per `APPLICATION_FAILURE` .
		In addition, if there is a persistent volume to be replaced in the same request, the identifier for the volume in the VNFD and the VIM is supplied to avoid multiple requests. However, the VNFC to which the volume is attached must be in the list of VNFCs to be healed. **This persistent volume update is only applicable to Openstack VIMs.**
		Any ephemeral ports and volumes managed by VNFM that are faulty or deleted will be recreated and attached to ensure the recovery is successful.

`APPLICATION_OR_VIRTUALISATION_FAILURE`	*Optional:* `vnfcInstanceId`	As per `APPLICATION_FAILURE`. Any ephemeral ports and volumes managed by VNFM that are faulty or deleted will be recreated and attached to ensure the recovery is successful if the VMs are redeployed.
`INVALID_VM_STATE`	*Optional:* `vnfcInstanceId`	As per `APPLICATION_FAILURE`.

Cause	additionalParams keys	Recovery behavior
PERSISTENT_VOLUME_FAILURE	*Mandatory:* resourceId virtualStorageDescId *Optional:* vnfcInstanceId	The treatment of the vnfcInstanceId is as per APPLICATION_FAILURE. The mandatory keys allow a new persistent volume to replace the existing volume without redeploying the VM. Once the data model is updated and the volume is replaced, and the VM is rebooted. **This is only applicable to Openstack VIMs.**
CHANGE_PERSISTENT_VOLUME	*Mandatory:* resourceId virtualStorageDescId	The mandatory keys allow a new persistent (including multi-attach) volume to replace the existing volume without redeploying the VM. Once the data model is updated and the volume is replaced, the VM is rebooted. **This is only applicable to Openstack VIMs.**
VIM_FAILURE	*None*	No additionalParams keys are activated, however the Grant from the NFVO **must** include new vimConnectionInfo to redeploy the VNF on a VIM that is available else the recovery request is rejected. **Note** The old deployment is not removed since the VIM is assumed to be unavailable if this cause is used; it needs to be manually removed once the VIM is reachable again.

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 modified against the VNF instance before instantiation.

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, as per the following examples. The parameters are optional by default but subject to the rules in table 9 for the different causes.

Example for *SOL003*:

Method type:

```
POST
```

VNFM Endpoint:

```
/vnf_instances/{vnfInstanceId}/heal
```

HTTP Request Header:

```
Content-Type:application/json
```

Request Payload (ETSI data structure: HealVnfRequest)

```
{
"cause":"VIRTUALISATION_FAILURE",
"additionalParams": {
 "virtualStorageDescId": "cf-cdr1-vol",
 "resourceId": " d8771acb-a32f-66dg-7bc2-8f4ec333ccb8"
},
"vnfcInstanceId": [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"
 }
```

The list of `vnfcInstanceIds` constrains recovery to the required VNFCs. However, the absence of this list means the request applies to the entire VNF.

The cause in the SOL002 HealVnfRequest has the same behavior as in the SOL003 API.

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

Recovering VM During Healing

If the recovery action is `REDEPLOY_ONLY` or `REBOOT_THEN_REDEPLOY` and VM needs to be redeployed during SOL002 and SOL003 heal operation, check whether the:

- ephemeral volume is missing or in error state; and recreate them.
- ephemeral neutron port is missing or in error state; and recreate them.

Note SOL002 heal is constrained to specific VNFCs, if `vnfcInstanceIds` are supplied in the heal payload.

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.