Healing Virtual Network Functions

- Healing Overview, on page 1
- Healing a VM, on page 1
- Recovery and Redeployment Policies, on page 11
- Enabling and Disabling the Host, on page 18
- SNMP Trap Based Recovery, on page 20
- Notifications and Events, on page 25

Healing Overview

As part of life cycle management, ESC heals the VNFs when there is a failure. The healing parameters are configured in the KPI section of the data model. The recovery policy specified during deployment controls the recovery. ESC supports recovery using the policy-driven framework.

ESC uses KPIs to monitor the VMs and events are triggered based on the KPI conditions. The actions to be taken for every event that is triggered are configured in the rules section during the deployment.

Healing a VM

Each VM group is configured to enable the healing. Healing is performed at two stages: Before the service is alive and after the service is alive with a recovery policy defined in the data model (at the VNF-level, and optionally overridden at the VM level).

The VMs are deployed and are being monitored. After ESC receives a VM Alive event, if it receives a VM Down event, the healing workflow attempts to recover the VM with the configured recovery policy.

If ESC doesn't receive a VM Alive after deployment, ESC recovers the VM with the recovery policy when a timeout happens. All the recovery procedures depend on the recovery policy configuration and the policy names described in the following paragraph. Some additional policies can be used to override the behavior when manually recovering the VM.

ESC provides a YANG-based data model with comprehensive details of all the parameters and description that is needed to define the healing. ESC uses two sections in the data model XML file which define the events and rules:

- The <kpi> section defines the type of monitoring, events, polling interval, and other parameters.
- The <rule> section defines the actions when the KPI monitoring events are triggered.

For more information on KPI, rules, and data model, see KPIs, Rules and Metrics.

The configuration involves the following steps:

1. Define kpi.
2. Define rules.

The following example shows how to configure the KPI in the data model:

```
<kpi>
<event_name>VM_ALIVE</event_name>
<metric_value>1</metric_value>
<metric_cond>GT</metric_cond>
<metric_type>UINT32</metric_type>
<metric_collector>
<type>ICMPPing</type>
<nicid>0</nicid>
<poll_frequency>3</poll_frequency>
<polling_unit>seconds</polling_unit>
<continuous_alarm>false</continuous_alarm>
</metric_collector>
</kpi>
```

The following example shows how to configure the rules for every event:

```
<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>
```

In the previous examples, we define a KPI to monitor the ICMP Ping on the nicid 0. It defines the attributes of metric conditions and polling. Based on the KPI, the VM_ALIVE event is triggered with appropriate values. The action in the corresponding rule defines what the next steps are:

- FALSE—Triggers recovery of the VM.
- TRUE—Triggers the defined action.

If recovery is triggered on the VM with the reboot, then redeploy option configured in the recovery policy, ESC reboots the VM as the first step to recover the VM. If it fails, the VM is undeployed and a new VM with the same day-0 configuration is deployed. ESC tries to reuse the same network configuration as MAC and IP Address as the previous VM.

Typically, if the VM is unreachable, ESC starts VM recovery on all unreachable VMs. During a network outage, ESC suspends VM recovery during the network outage, thus delaying the VM recovery. ESC detects the unreachable VM and evaluates the reachability of the gateway first to detect the presence of a network failure.

If ESC cannot ping the gateway, no action is performed to recover the VM. VM recovery resumes when the gateway becomes reachable.

If there is a double fault condition, that is, when the network gateway and the VM failure occur at the same time, the ESC automatically performs VM monitoring after the gateway is reachable again.

For information on healing a VNF using the ETSI API, see the *Cisco Elastic Services Controller NFV MANO Guide*.

Recovery Policy

ESC has the following VM recovery types that you can specify when you deploy a VNF:

1. • **Auto Recovery**
   • **Manual Recovery**

ESC supports recovery using the policy-driven framework, see Recovery Policy (Using the Policy Framework) for details.

There are three types of actions for a VM recovery that can be specified in the deployment data model:

- **REBOOT_THEN_REDEPLOY (default)**—When a VM down event is received or the timer expires, the healing workflow first attempts to reboot the VM, if it fails to reboot, then it attempts to redeploy the VM on the same host.
- **REBOOT_ONLY**—When a VM down event is received or the timer expires, the healing workflow only attempts to reboot the VM.
- **REDEPLOY_ONLY**—When a VM down event is received or the timer expires, the healing workflow only attempts to redeploy the VM.
- **MIGRATE_ONLY**—When a VM down event is received or the timer expires, the healing workflow only attempts to migrate the VM..
- **REBOOT_THEN_MIGRATE**—When a VM down event is received or the timer expires, the healing workflow first attempts to reboot the VM, if it fails to reboot, then it attempts to migrate the VM.

  VM Migrate is supported from 5.7 version onwards. Please refer to the VM Operations section to know more about VM Migrate operation.

Note If the policy involves REBOOT_THEN_REDEPLOY and REDEPLOY_ONLY for redeploying the VMs, and if the placement policy is not enforced, then the VIM decides which host to redeploy the VM on.

Note ESC supports both manual and auto-recovery for vCloud Director. All three types of recovery actions are applicable for the vCloud Director. The REBOOT_THEN_REDEPLOY is the default recovery action. For vCD deployment, see Deploying Virtual Network Functions on VMware vCloud Director (vCD).

Any recovery action that involves redeploying the VM will automatically recreate and attach ephemeral ports and volumes managed by ESC, that are faulty or deleted to ensure the recovery is successful.

Auto Recovery

In Auto recovery, the recovery type parameter is set to Auto. ESC automatically recovers the VM with the specified <action-on-recovery> value in the recovery policy. The recovery type is auto by default if the user does not choose a recovery type.

```
<recovery_policy>
 <recovery_type>AUTO</recovery_type>
 <action_on_recovery>REBOOT_THEN_REDEPLOY</action_on_recovery>
 <max_retries>3</max_retries>
</recovery_policy>
```

Manual Recovery

Manual Recovery of a VM

In manual recovery, ESC sends the VM_MANUAL_RECOVERY_NEEDED notification to northbound (NB) and waits for the instruction from NB for recovery. ESC performs recovery when it receives recovery instruction from NB. For manual recovery of a complete deployment, see Manual Recovery of a Deployment, on page 7

ESC also supports overriding of the actions on a single request basis, using the action-on-recovery parameter in the recovery policy. In addition to the 3 recovery actions listed before, there are 2 more recovery actions available:

1.
   - **RESET_STATE_THEN_REBOOT** – before rebooting the VM, the VM state is reset to allow the VIM to reboot the VM for recovery. This is only applicable to Openstack.
   - **DISASTER_RECOVERY** – when the VIM to which the VNF has been deployed, become unavailable and there is a need to move the VNF to a new VIM for service continuity, this action can be invoked to redeploy the VNF (entire service, not individual VMs) on to a new VIM.

     To use this action, it must be preceded by a model-only service update to update the VIM locator; failure to carry out this step will result in the recovery request failing. See below for details on how to perform this type of service update (via the REST API only).

     The original VNF is not attempted to be removed. Since it is assumed that use of this recovery action implies that the VNF is unreachable from the orchestration stack and when the VIM itself has been recovered, the old deployment **must** be manually cleaned.

The manual recovery policy data model is as follows

```
<vm_group>
        ...
        <recovery_policy>
         <recovery_type>MANUAL</recovery_type>
         <action_on_recovery>REBOOT_THEN_REDEPLOY</action_on_recovery>
         <max_retries>3</max_retries>
        </recovery_policy>
        ...
  </vm_group
```

For more information about recovery policy parameters in the data model, see Elastic Services Controller Deployment Attributes. For more information about configuring the recovery policy in the ESC Portal (VMware only), see the Deploying VNFs on VMware vCenter using ESC Portal.

The VM_MANUAL_RECOVERY_NEEDED notification is as follows:

```
===== SEND NOTIFICATION STARTS =====
 WARN  Type: VM_MANUAL_RECOVERY_NEEDED
 WARN  Status: SUCCESS
 WARN  Status Code: 200
 WARN  Status Msg: Recovery event for VM
[manual-recover_error-g1_0_7d96ad0b-4f27-4a5a-bdf7-ec830e93d07e] triggered.
 WARN  Tenant: manual-recovery-tenant
 WARN  Service ID: NULL
```

```
WARN  Deployment ID: 08491863-846a-4294-b305-c0002b9e8daf
WARN  Deployment name: dep-error
WARN  VM group name: error-g1
WARN  VM Source:
WARN      VM ID: ffea079d-0ea2-4d47-ba31-26a08e6dff22
WARN      Host ID: 3a5351dc4bb7df0ee25e238a8ebbd6c6fcdf225aebcb9dff6ba10249
WARN      Host Name: my-server-27
WARN      [DEBUG-ONLY] VM IP: 192.168.0.3;
WARN  =====  SEND NOTIFICATION ENDS  =====
```

APIs for Manual Recovery of a VM

You can perform the manual recovery using the Confd and Rest APIs. The manual recovery request can be configured to override the predefined recovery action to any desired action.

Netconf API `recovery-vm-action DO generated vm name [xmlfile]`

To perform recovery using the API, login to esc_nc_cli and run the following command:

```
$ esc_nc_cli --user <username> --password <password> recovery-vm-action DO [xmlfile]
```

The recovery is performed and the recovery notification is sent to NB.

Note Recovery (recovery-vm-action DO <VM-NAME>) can be performed after the VM is alive and the service is active. If the deployment is incomplete, it must be completed before performing recovery.

If a failover happens during a configurable manual recovery, the manual recovery resumes with predefined recover action.

The migration of any deployment must always use default recovery policy. You must not provide recovery action for VM/VNF manual recovery in an LCS based recovery.You must not use enable monitor and configurable manual recovery options together.

REST API

```
http://ip:8080/ESCAPI/#!/Recovery_VM_Operations/handleOperation
```

```
POST /v0/{internal_tenant_id}/deployments/recovery-vm/{vm_name}
```

Recovery VM operation payload:

```
{
"operation":"recovery_do",
"properties":{
"property":[
{
"name":"action",
"value":"REDEPLOY_ONLY"
}
]
}
}
```

In order to perform a model-only service update, a new parameter can be supplied to the edit-config API to prevent any action to be taken on the VIM and limit the update to the ESC data model only. This allows the preparation of the data model to complete until such time that the deployment is ready to be updated on the VIM:

```
http://ip:8080/ESCManager/v0/conf/edit-config?modelOnly=true
```

The VIM locator update, for example, prior to invoking the recovery API with `DISASTER_RECOVERY` as the <action-on-recovery>:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <tenants>
      <tenant>
         <name>admin-tenant</name>
         <deployments>
            <deployment>
               <name>test-deploy</name>
               <networks>
                  <network>
                     <name>test-network</name>
                     <locator>
                        <vim_id>my-ucs-59</vim_id>
                        <vim_project>admin</vim_project>
                     </locator>
                  </network>
               </networks>
               <vm_group>
                  <name>g1</name>
                  <locator>
                     <vim_id>my-ucs-59</vim_id>
                     <vim_project>admin</vim_project>
                  </locator>
                  <bootup_time>120</bootup_time>
               </vm_group>
            </deployment>
         </deployments>
      </tenant>
   </tenants>
</esc_datamodel>
```

Note Do not forget that the old deployment has to be deleted in the disaster recovery scenario, once the VIM is available again.

A further use for this API is to update the persistent volume UUID prior to rebooting the VM via the recovery API documented above. This has the benefit of negating the need to remove the VM group and re-add it, as per earlier versions of ESC. Here is an example payload:

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <tenants>
    <tenant>
       <name>my-tenant</name>
       <deployments>
          <deployment>
             <name>my-dep</name>
             <vm_group>
                <name>my-vm</name>
                <bootup_time>1800</bootup_time>
                <volumes>
                   <volume>
                      <name>new-volume</name>
                      <volid>1</volid>
                      <bus>ide</bus>
```

```
                    <type>lvm</type>
                  </volume>
                  <volume nc:operation="delete">
                    <name>old-volume</name>
                    <volid>1</volid>
                  </volume>
                </volumes>
              </vm_group>
            </deployment>
          </deployments>
        </tenant>
      </tenants>
    </esc_datamodel>
```

Supported VM States and Service Combinations for Manual Recovery of a VM

The API, recovery-vm-action, applies to both auto and manual recovery types, but only under certain VM states and services. The following table shows the details. In general, during deployment, service update, undeployment, and recovery, the manual recovery action is rejected by ESC.

VM State	Service State	recovery-vm-action
ALIVE	ACTIVE	supported
ALIVE	ERROR	supported
ERROR	ERROR	supported

Manual Recovery of a Deployment

Recovery Without Monitoring Parameters

ESC supports manual recovery of VMs at the service level, that is, recovery of a complete deployment. After the successful deployment of a service, the service may move into an error state because of failed VMs. ESC can manually recover these failed VMs, or the complete deployment through a deployment recovery request. For manual recovery of a single VM, see Manual Recovery, on page 4.

APIs for Manual Recovery of a Deployment

You can perform the manual recovery using the NETCONF and REST APIs.

The manual recovery request can be configured to override the predefined recovery action to any desired action.

Note There is no service active notification after the deployment recovery. You must run a query, for example, *esc_nc_cli --user <username> --password <password> get esc_datamodel* to see if the service state of the deployment is active or not.

If a failover happens during a configurable manual recovery, the manual recovery resumes with predefined recover action.

The migration of any deployment must always use a default recovery policy. You must not provide recovery action for VM/VNF manual recovery in an LCS-based recovery.You must not use enable monitor and configurable manual recovery options together.

NETCONF API

```
svc-action RECOVER tenant-name deployment-name [xmlfile]
```

To perform recovery using the API, login to `esc_nc_cli`.

REST API

```
POST /v0/{internal_tenant_id}/deployments/service/{internal_deployment_id}
Content-Type: application/xml
Accept: application/json
Callback: http://172.16.0.1:9010/
Callback-ESC-Events: http://172.16.0.1:9010/
<service_operation xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <operation>recover</operation>
</service_operation>
```

where,

internal_tenant_id—is the system admin tenant ID or the tenant name.

internal_deployment_id—is the deployment name.

Supported VM States and Service Combinations for Manual Recovery of a Deployment

The API, svc-action RECOVER, applies to both auto and manual recovery types, but only under certain VM states and services. The following table shows the details. In general, during deployment, service update, undeployment, and recovery, the manual recovery action is rejected by ESC.

Note ESC accepts a VM level recovery request when the service is in an active or error state.

Notifications are not sent to NB if all VMs are in the ALIVE state after a service recovery request.

VM State	Service State	svc-action RECOVER
ERROR	ERROR	supported
ERROR	ERROR	supported

Recovery Enabled with Monitoring Parameters

During manual recovery, you can recover a VM depending on its monitoring parameters. If the VM is in error state, set the monitoring parameters to bring back the VM in error state to live state. If the VM is recovered, then ESC sends a RECOVERY_CANCELLED notification. If the VM does not come back live, then the recovery process is triggered. See Manual Recovery for more details.

NETCONF API

```
svc-action SET_MONITOR_AND_RECOVER <tenant-name> <dep-name>
```

Recovery notification:

```
===== SEND NOTIFICATION STARTS =====
 WARN  Type: VM_RECOVERY_INIT
 WARN  Status: SUCCESS
 WARN  Status Code: 200
 WARN  Status Msg: Recovery with enabling monitor first event for VM Generated ID
[dep-resource_g1_0_74132737-d0a4-4ef0-bd9e-86465c1017bf] triggered.
```

Note Recovery enabled with monitoring parameters is for manual recovery at service level only.

The *monitor_on_error* parameter enables continuous monitoring of the VMs in error state.

```
<recovery_policy>
                        <recovery_type>AUTO</recovery_type>
                        <action_on_recovery>REBOOT_ONLY</action_on_recovery>
                        <max_retries>1</max_retries>
                        <monitor_on_error>true</monitor_on_error>
                    </recovery_policy>
```

The default value is false.

if false, monitoring is **unset** on the vm in error state.

If true, monitoring is **set** on the vm in error state. If any VM Alive event occurs later (after VM_RECOVERY_COMPLETE), the VM is moved back to alive state.

VM Recovery Using Migrate Option

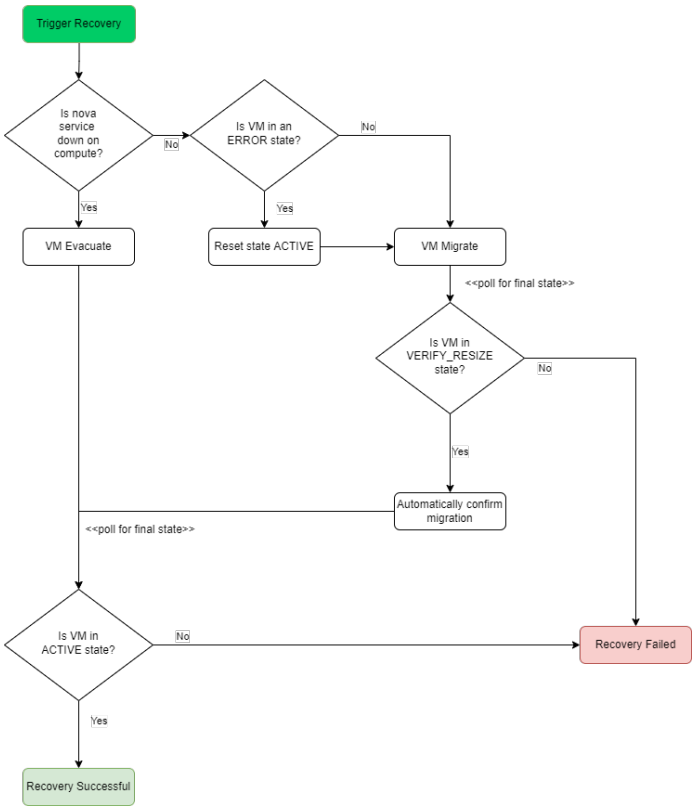
ESC Manager supports two new recovery policies that are REBOOT_THEN_MIGRATE and MIGRATE_ONLY. These recovery policies use OpenStack cold migration without host parameters.

You can perform the manual recovery with these new policies using the existing recovery Netconf or REST API.

Once the VM moves to the VERIFY_RESIZE state in the recovery flow, ESC auto confirms the migration.

ESC performs either of the two actions in the recovery process:

- If the *nova-compute* service is down on the host, then the VM evacutes and OpenStack rebuild the VM in another host.
- If the VM is in an error state, ESC resets the VM to ACTIVE state and migrates.



For the REBOOT_THEN_MIGRATE recovery policy, the VM reboots and if the reboot fails, then ESC triggers the VM migration.

For the MIGRATE_ONLY recovery policy, ESC triggers the VM migration.

Example Payload:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
    <tenants>
        <tenant>
            <name>admin</name>
            <deployments>
                <deployment>
                    <name>Test-Cirros-VNF</name>
                    <vm_group>
                        <name>Test-vm-group</name>
                        <bootup_time>60</bootup_time>
                        <recovery_wait_time>0</recovery_wait_time>
                        <image>Automation-Cirros-Image</image>
                        <flavor>Automation-Cirros-Flavor</flavor>
                        <interfaces>
                            <interface>
                                <nicid>0</nicid>
                                <network>esc-net</network>
                            </interface>
                        </interfaces>
                        <scaling>
                            <min_active>1</min_active>
                            <max_active>1</max_active>
                            <elastic>true</elastic>
                        </scaling>
```

```
                        <kpi_data>
                            <kpi>
                                <event_name>VM_ALIVE</event_name>
                                <metric_value>1</metric_value>
                                <metric_cond>GT</metric_cond>
                                <metric_type>UINT32</metric_type>
                                <metric_collector>
                                    <type>ICMPPing</type>
                                    <nicid>0</nicid>
                                    <poll_frequency>3</poll_frequency>
                                    <polling_unit>seconds</polling_unit>
                                    <continuous_alarm>false</continuous_alarm>
                                </metric_collector>
                            </kpi>
                        </kpi_data>
                        <rules>
                            <admin_rules>
                                <rule>
                                    <event_name>VM_ALIVE</event_name>
                                    <action>"ALWAYS log"</action>
                                    <action>"TRUE servicebooted.sh"</action>
                                    <action>"FALSE recover autohealing"</action>
                                </rule>
                            </admin_rules>
                        </rules>
                        <config_data/>
                        <recovery_policy>
                            <recovery_type>AUTO</recovery_type>
                            <action_on_recovery>REBOOT_THEN_MIGRATE</action_on_recovery>
                            <max_retries>3</max_retries>
                            <monitor_on_error>false</monitor_on_error>
                        </recovery_policy>
                    </vm_group>
                </deployment>
            </deployments>
        </tenant>
    </tenants>
</esc_datamodel>
```

Notifications:

ESC sends the following notifications once the recovery is successful.

```
VM_RECOVERY_INIT
VM_RECOVERY_MIGRATED
VM_RECOVERY_COMPLETE
```

Recovery and Redeployment Policies

ESC uses a policy driven framework to perform actions based on the lifecycle stages in a deployment. A deployment consists of several stages through its lifecycle. Each lifecycle stage (LCS) is associated with a condition. The condition in turn is associated with a predefined action or custom scripts. These conditions and actions are specified within the policies tag in the data model. For more information on Policy driven Framework, see Policy-Driven Data model.

The recovery and redeployment workflows in ESC are policy driven. When VNFs are deployed, the recovery and redeployment policies are specified in the deployment data model. These policies are based on the lifecycle stages of VM or VNF and have actions associated with it.

When a deployment data model is created, you can specify the following policies:

- **Recovery Policy**—The recovery policy is for the VM lifecycle, that is for the recovery of a single VM. Based on the predefined actions, the VM is rebooted, or redeployed. You can perform recovery without using the policy framework. See Recovery Policy, on page 3.
- **Redeployment Policy**—The redeployment policy is for the entire deployment lifecycle, that is for all the VM groups within a deployment. Based on a set of predefined actions, the host is disabled, and VMs are recovered in the deployment.

If the VM recovery fails after the maximum attempts, ESC disables the host and triggers redeployment for all VMs within the deployment. All VMs are undeployed from the old host and redeployed to a new host.

ESC supports redeploying the failed VMs first. During a redeployment, the failed VMs are recovered first, and the VMs that have not failed are queued up for redeployment.

Recovery Policy (Using the Policy Framework)

ESC supports recovery of VMs using the policy-driven framework data model. The recovery is based on the lifecycle stages of VM deployment and predefined actions.

For auto and manual recovery, see Recovery Policy, on page 3.

The table below describes the predefined actions performed at different lifecycle stages.

Predefined Action Name	Scope	Description
SET_RECOVERY::REBOOT_ONLY	Deployment	Sets the recovery action for all VM groups (in a deployment), or for a VM (in a VM group) to REBOOT_ONLY.
SET_RECOVERY::REBOOT_THEN_REDEPLOY	Deployment	Sets the recovery action for all VM groups (in a deployment), or for a VM (in a VM group) to REBOOT_THEN_REDEPLOY.
SET_RECOVERY::REDEPLOY_ONLY	Deployment	Sets the recovery action for all VM groups (in a deployment), or for a VM (in a VM group) to REDEPLOY_ONLY.
SET_RECOVERY::RESET_STATE_THEN_REBOOT	Deployment	Sets the recovery action for all VM groups (in a deployment), or for a VM (in a VM group) to RESET_STATE_THEN_REBOOT (Openstack only).
SET_RECOVERY::MIGRATE_ONLY	Deployment	Sets the recovery action for all VM groups (in a deployment), or for a VM (in a VM group) to MIGRATE_ONLY.
SET_RECOVERY::REBOOT_THEN_MIGRATE	Deployment	Sets the recovery action for all VM groups (in a deployment), or for a VM (in a VM group) to REBOOT_THEN_MIGRATE.

Supported Conditions and Predefined Action Combinations

The following table describes the supported LCS conditions and its actions for recovery and redeployment policies using the policy framework. For more details on the policy driven framework, see Recovery and Redeployment Policies, on page 11.

Condition	Predefined action	Description
LCS::PRE_DEPLOY —Occurs just before deploying VMs in a deployment.	• SET_RECOVERY::REBOOT_ONLY —Sets the recovery action for all VM groups (in a deployment), or for a VM (in a VM group) to REBOOT_ONLY. • SET_RECOVERY:: REBOOT_THEN_REDEPLOY —Sets the recovery action for all VM groups (in a deployment), or for a VM (in a VM group) to REBOOT_THEN_REDEPLOY. • SET_RECOVERY:: REDEPLOY_ONLY —Sets the recovery action for all VM groups (in a deployment), or for a VM (in a VM group) to REDEPLOY_ONLY. • SET_RECOVERY_ REDEPLOY:: SERIALIZED —Queues up the recoveries in the deployment. That is, new recovery does not start until the current ongoing recovery completes.	Choose any one of the predefined action for recovery. Choose SET_RECOVERY_ REDEPLOY::SERIALIZED, if DROP_RECOVERIES action is used. This means the VMs need to be kept on the original host once the redeployment fails. If not chosen, then DROP_RECOVERIES action cannot be used.
LCS:: POST_DEPLOY_ALIVE —Occurs immediately after the deployment is active.		
LCS::DEPLOY_ERR —Occurs immediately after the deployment fails.	DISABLE_HOST —Disables the host(s) the deployment or the VM is using.	-
LCS::POST_ DEPLOY::VM_RECOVERY _ERR —Occurs immediately after the recovery of one VM fails.	DISABLE_HOST —Disables the host(s) the deployment or the VM is using. REDEPLOY_ALL:: DISABLE_HOST —Disables the host the VM is using then trigger redeploy for all VMs (within a deployment), or all VMs on that host.	Choose DISABLE_HOST if needed. REDEPLOY_ALL:: DISABLE_HOST Choose if redeploy is needed after disabling the host. DISABLE_HOST and REDEPLOY_ALL::DISABLE_HOST cannot be together as they overlap.

Condition	Predefined action	Description
LCS::POST_ DEPLOY::VM_RECOVERY_ REDEPLOY_ERR —Occurs immediately after the redeploy of one VM fails.	• DISABLE_HOST —Disables the host(s) the deployment or the VM is using. • DROP_RECOVERIES —Drops all pending recoveries in the deployment.	DISABLE_HOST Choose if DISABLE_HOST is needed. Choose DROP_RECOVERIES if VMs need to be kept on the original host once the redeploy fails. When choosing DROP_RECOVERIES, ensure SET_RECOVERY_ REDEPLOY:: SERIALIZED action is complete.

Redeployment Policy

Redeployment policies are a part of the policy driven framework. Using this framework, you can specify predefined actions for specific lifecycle conditions. For more information on ESC policy driven framework, see Policy-Driven Data model.

Redeployment policies are invoked when a VM recovery fails after the maximum number of attempts. ESC disables the host and triggers redeployment for all VMs within the deployment. All VMs are undeployed from the old host and redeployed to a new host. Based on the combination of lifecycle stages (LCS) and predefined actions, the VMs are redeployed. The redeployment policy is for the entire deployment.

You can use the following lifecycle condition and action combination in the policy datamodel.

Note ESC uses default recovery action, **REBOOT_THEN_REDEPLOY** if nothing is chosen.

A sample redeployment policy data model is as follows:

```
<tenants>
        <tenant>
            <name>xyz-redeploy-ten-0502</name>
            <deployments>
                <deployment>
                    <name>dep</name>
                    <policies>
                        <policy>
                            <name>1</name>
                            <conditions>
                                <condition>
                                    <name>LCS::PRE_DEPLOY</name>
                                </condition>
                            </conditions>
                            <actions>
                                <action>
                                    <name>SET_RECOVERY::REBOOT_THEN_REDEPLOY</name>
                                    <type>pre-defined</type>
                                </action>
```

```
            <action>
                <name>SET_RECOVERY_REDEPLOY::SERIALIZED</name>
                <type>pre-defined</type>
            </action>
        </actions>
    </policy>
    <policy>
        <name>2</name>
        <conditions>
            <condition>
                <name>LCS::POST_DEPLOY_ALIVE</name>
            </condition>
        </conditions>
        <actions>
            <action>
                <name>SET_RECOVERY::REBOOT_ONLY</name>
                <type>pre-defined</type>
            </action>
        </actions>
    </policy>
    <policy>
        <name>3</name>
        <conditions>
            <condition>
                <name>LCS::DEPLOY_ERR</name>
            </condition>
        </conditions>
        <actions>
            <action>
                <name>DISABLE_HOST</name>
                <type>pre-defined</type>
            </action>
        </actions>
    </policy>
    <policy>
        <name>4</name>
        <conditions>
            <condition>
                <name>LCS::POST_DEPLOY::VM_RECOVERY_ERR</name>
            </condition>
        </conditions>
        <actions>
            <action>
                <name>REDEPLOY_ALL::DISABLE_HOST</name>
                <type>pre-defined</type>
            </action>
        </actions>
    </policy>
    <policy>
        <name>5</name>
        <conditions>
            <condition>
               <name>LCS::POST_DEPLOY::VM_RECOVERY_REDEPLOY_ERR</name>

            </condition>
        </conditions>
        <actions>
            <action>
                <name>DISABLE_HOST</name>
                <type>pre-defined</type>
            </action>
            <action>
                <name>DROP_RECOVERIES</name>
                <type>pre-defined</type>
```

```
                        </action>
                    </actions>
                </policy>
            </policies>
            <vm_group>
                <name>Group1</name>
                <image>xyz-redeploy-img-0502</image>
                <flavor>xyz-redeploy-flv-0502</flavor>
                <recovery_policy>
                    <max_retries>1</max_retries>
                </recovery_policy>
        ......
        ......
      </deployment>
     </deployments>
  </tenant>
</tenants>
```

Supported Lifecycle Stages (LCS)

Condition Name	Scope	Description
LCS::PRE_DEPLOY	Deployment	Occurs just before deploying VMs of the deployment.
LCS::POST_DEPLOY_ALIVE	Deployment	Occurs immediately after the deployment is active.
LCS::DEPLOY_ERR	Deployment	Occurs immediately after the deployment fails.
LCS::POST_DEPLOY:: VM_RECOVERY_ERR	Deployment	Occurs immediately after the recovery of one VM fails. (This is specified at deployment level and applies to all VM groups)
LCS::POST_DEPLOY:: VM_RECOVERY_REDEPLOY_ERR	Deployment	Occurs immediately after the redeployment of one VM fails. (This is specified at deployment level and applies to all VM groups)

Supported Predefined actions

Predefined Action Name	Scope	Description
DISABLE_HOST	Deployment	Disables the host(s) the deployment or the VM is using.
REDEPLOY_ALL::DISABLE_HOST	Deployment	Disables the host the VM is using then trigger redeploy for all VMs (within a deployment), or all VMs on that host.
DROP_RECOVERIES	Deployment	Drops all pending recoveries in the deployment.
SET_RECOVERY_REDEPLOY::SERIALIZED	Deployment	Queues up the recoveries in the deployment. That is, new recovery does not start until the current ongoing recovery completes.

Limiting the Number of Redeployments

Cisco Elastic Services Controller (ESC) limits the number of redeployments using the following parameters:

- **max_redep**: limits the maximum number of redeployments. By default, the max_redep value is -1, which indicates that there is no limit on the maximum number of redeployments. You can change this value using the bootvm.py arguments or REST API.
- **redep_count**: consists of the current number of redeployments. The redep_count automatically increases by 1 after a redeployment, irrespective of the success or failure of the redeployment.

Note The redeployment limit is for,

- redeployments triggered by REDEPLOY_ALL::DISABLE_HOST policy.
- deployments with single VIM configuration only.

Cisco Elastic Services Controller (ESC) performs redeployment,

- if the maximum number of redeployments is set to the default value of -1, that is max_redep = -1.
- if the current number of redeployments is less than the maximum number of redeployments (redep_count < max_redep), then ESC performs redeployment, and increases the redeployment count by 1 after the redeployment is complete.

ESC does not perform any redeployment if the redeployment count is more than or equal to the maximum number of redeployments, (redep_count >= max_redep).

You can use the bootvm.py parameters and REST APIs to configure the values.

Using the bootvm.py parameters

Specify the max_redep value in the esc_params.conf file that contains the following line: default.max_redep = 3

Run the command, `bootvm.py ... --esc_params_file <path_to_file>/esc_params.conf ...`

Using the REST APIs

You can retrieve, and reset the redep_count parameter using the following APIs:

- To retrieve the current value of redep_count:

  ```
  GET http://<ESC IP>:8080/ESCManager/v0/systemstate/redep_count
  ```
- To reset redep_count:

  ```
  POST http://<ESC IP>:8080/ESCManager/v0/systemstate/redep_count/reset
  ```

You can also use the REST API to retrieve and change the max_redep value.

- To retrieve the current value of max_redep:

  ```
  GET http://<ESC IP>:8080/ESCManager/v0/config/default/max_redep
  ```
- To change the max_redep value:

  ```
  PUT http://<ESC IP>:8080/ESCManager/v0/config/default/max_redep/<value>
  ```

  where <value> can be,

-1, which is the default value with no limit

0, which does not allow any redeployment

more than zero (> 0), which specifies the maximum number of redeployments allowed.

You can also use the escadm tool to configure these values. For more information on the escadm tool, see the Elastics Services Controller Install and Upgrade Guide.

For more details on the redeployment policy, see Redeployment Policy, on page 14.

The VMs that are not redeployed because of the redeployment limit are moved to error state. ESC manually recovers these VMs in error state by enabling the monitoring operation on each VM.

To enable monitoring operation on a single VM in error state:

```
POST http://<ESC IP>:8080/ESCManager/v0/<internal-tenant-id>/deployments/vm/<vm-name> {
"operation" : "enable_monitoring" }
```

You can also enable monitoring using the esc_nc_cli command:

```
esc_nc_cli --user <username> --password <password> vm-action ENABLE_MONITOR <generated vm
name>
```

As part of the manual recovery process, the enable monitoring operation moves the VMs from error state to alive state. If manual recovery fails for these VMs, then auto recovery is triggered.

To enable the monitoring operation on VMs (in error state) in a deployment:

```
POST http://<ESC
IP>:8080/ESCManager/v0/<internal-tenant-id>/deployments/service/<internal-deployment-id> {
 "operation" : "enable_monitoring" }
```

You can also enable monitoring using the esc_nc_cli command:

```
esc_nc_cli --user <username> --password <password> svc-action ENABLE_MONITOR <tenant> <dep
name>
```

As part of the manual recovery process, the enable monitoring operation moves all the VMs in a deployment from error state to alive state. If manual recovery fails, then auto recovery is triggered on all the VMs in the deployment.

For more information, see Monitoring Operations and Recovery Policy.

Enabling and Disabling the Host

You can enable or disable the host on OpenStack using NETCONF and REST APIs. The host can also be disabled during a VNF recovery or redeployment scenario.

Note Enabling and disabling the host on VMware vCenter is not supported.

You cannot enable or disable a host on a non-default VIM using NETCONF and REST APIs in an ESC with multiple OpenStack VIMs.

Using NETCONF

```
/opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli --user <username> --password <password>
host-action < ENABLE | DISABLE > <host-name>
```

The payload is as follows:

```
<hostAction xmlns="http://www.cisco.com/esc/esc">
  <actionType>ENABLE/DISABLE</actionType>
  <hostName>my-server</hostName>
</hostAction>
```

where,

- actionType is ENABLE or DISABLE
- hostName is the host name or UUID of the target host

Using REST

```
POST /v0/hosts/{hostName}/disable
POST /v0/hosts/{hostName}/enable
GET /v0/hosts/{hostName}/status
```

Enabling the Host

By enabling the host, you bring a disabled host back to OpenStack and deploy new VM instances on it.

Sample NETCONF notification is as follows:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-03-30T15:04:05.95+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Host action successful</status_message>
    <vm_source>
      <hostname>my-server</hostname>
    </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>HOST_ENABLE</type>
    </event>
  </escEvent>
</notification>
```

Sample REST notification is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <host_action_event xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <event_type>HOST_ENABLE</event_type>
    <host_name>my-server</host_name>
    <message>Host action successful</message>
</host_action_event>
```

Disabling a Host

During VNF redeployment, you disable the host, and trigger a host-based redeployment for all the VMs within that deployment. This ensures that the redeployed VMs are on a different host. You can also disable a host when it is not working properly. Once a host is disabled, it is removed from OpenStack, so that no new instances are deployed on it.

Sample NETCONF notification is as follows:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-03-30T15:03:48.121+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
```

```
    <status_code>200</status_code>
    <status_message>Host action successful</status_message>
    <vm_source>
      <hostname>my-server</hostname>
    </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>HOST_DISABLE</type>
    </event>
  </escEvent>
</notification>
```

Sample REST notification is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<host_action_event xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <event_type>HOST_DISABLE</event_type>
  <host_name>my-server</host_name>
  <message>Host action successful</message>
</host_action_event>
```

SNMP Trap Based Recovery

Overview:

ESC monitors the VNFs deployed through itself and triggers the recovery when the VNF is not reachable. This feature enables the ESC to listen to certain SNMP traps related to the host where the VNFs are running and triggers the recovery for all the VNFs deployed on that host in one shot.

This feature applies only when customers use Cisco Virtual Infrastructure Manager (CVIM) as their VIM. To enable this feature, perform certain configurations as a pre-requisite on both CVIM and ESC.

1. Configure CVIM to send traps to ESC:

Login to CVIM Management Node and edit *<your setup data folder>/setup_data.yaml* file as following:

```
CVIM_MON: {enabled: true, ui_access: true}
SNMP:
  enabled: true
  managers:
     - {address: <ESC or VIP IP>, community: <community string>, port: 162, version: v2c}
     - {address: <ESC or VIP IP>, community: <community string>, port: 162, version: v2c}
SERVER_MON:
  enabled: true
  host_info: [ALL]
```

By default, CVIM uses UDP 162 to send the SNMP traps to ESC. To use a custom port, configure the port under SNMP: managers: port

For more information about the CVIM configuration, please refer the CVIM documentation

To apply the configuration changes made in the *setup_data.yaml* file, run the following command from CVIM management node:

```
ciscovim reconfigure --setupfile <your setup data folder>/setup_data.yaml
```

Now, CVIM sends SNMP traps to ESC.

2. Configure ESC to listen to SNMP traps from CVIM:

Perform the following steps in both Active and Standby ESC nodes after a fresh install or In-Service Upgrade or Backup Restore Upgrade. In case of RPM Upgrade, perform the RPM Upgrade Section mentioned and perform the following steps:

- Login to ESC as a root user.
- Edit the file /opt/cisco/esc/esc-config/esc-snmptrapd.conf only if there are changes apart form the default configurations.

```
authCommunity log,execute,net public
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
 Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value
Pair Array:\n%v \n -------------- \n
traphandle SNMPv2-SMI::enterprises.9.9.855.0.1 /usr/bin/python3
/opt/cisco/esc/esc-scripts/esc-process-traps.py
```

- By default, *snmptrapd* listener listens in UDP port 162. To configure a own port, add the following line in */opt/cisco/esc/esc-config/esc-snmptrapd.conf*,

```
snmpTrapdAddr udp:<your port number>
```

  Make sure, the same port has been configured in CVIM in *setup_data.yaml*

- To listen to traps in both IPv4 and IPv6, configure as following, This step is optional

```
snmpTrapdAddr udp:<port number>,udp6:<port>
```

- Configure the following attributes in */opt/cisco/esc/esc-config/esc-process-trap.properties*

  host-recovery-time:

  Set the manual recovery time in minutes for the CVIM hosts. When more than one trap is received by ESC for the same host, ESC triggers the recovery for the VNFs running on that host for the first trap received and ignores further traps received for that host until the host -recovery time expires for that host.

  The default time is 30 minutes.

  handle-trap-severity:

  Set the severity level for the traps to be processed by ESC.

  Ignore the traps received with severity level greater than handle-trap-severity value. The possible values are:

  1. emergency
  2. emergency or critical
  3. emergency or critical or major
  4. emergency or critical or major or alert
  5. emergency or critical or major or alert or informational

  The default is 3, only traps with severity emergency or critical or major are processed

```
cat /opt/cisco/esc/esc_database/esc-process-trap.properties
# This is the configuration file for the esc-process-trap.py
#
# Set the manual recovery time (in minutes) for the host
# During this time period, recovery will not be triggered by esc-process-trap.py
# if the recovery is already triggered for the host. The default is 30 minutes
host-recovery-time=30
```

```
# Set the severity level of traps that needs to be processed by ESC
# Traps received with severity level greater than handle-trap-severity value will be
ignored. Possible values are,
# 1 - emergency
# 2 - emergency / critical
# 3 - emergency/ critical / major
# 4 - emergency / critical / major / alert
# 5 - emergency / critical / major / alert / informational
# The default is 3
handle-trap-severity=3
```

- Enable snmptrapd service.This makes sure that the snmptrapd service starts automatically on every ESC VM reboot.

```
[root@dnd-test-esc-5-10-0-14-ha-1 admin]# systemctl enable snmptrapd
Created symlink from /etc/systemd/system/multi-user.target.wants/snmptrapd.service to
/usr/lib/systemd/system/snmptrapd.service.
```

- Start the snmptrapd service in active node using the following command:

```
systemctl start snmptrapd
```

- Make sure the service is started by using the following command. The command can take upto 3 minutes to check the node's role in active or standby.

```
[root@dnd-test-esc-5-10-0-14-ha-1 admin]# systemctl status snmptrapd
snmptrapd.service - Simple Network Management Protocol (SNMP) Trap Daemon.
Loaded: loaded (/usr/lib/systemd/system/snmptrapd.service; enabled; vendor preset:
disabled)
    Active: active (running) since Fri 2023-02-24 08:11:31 UTC; 6s ago
   Process: 3973 ExecStartPre=/opt/cisco/esc/esc-scripts/esc-snmptrapd-precond.sh
(code=exited, status=0/SUCCESS)
  Main PID: 3986 (snmptrapd)
    CGroup: /system.slice/snmptrapd.service
            └─3986 /usr/sbin/snmptrapd -f -C -c
/opt/cisco/esc/esc_database/esc-snmptrapd.conf -A -Lf /var/log/esc/esc-snmptrapd.log
-f
Feb 24 08:11:30 dnd-test-esc-5-10-0-14-ha-1 systemd[1]: Starting Simple Network Management
 Protocol (SNMP) Trap Daemon....
Feb 24 08:11:30 dnd-test-esc-5-10-0-14-ha-1 esc-snmptrapd-precond.sh[3973]: Waiting to
 check if node becomes ACTIVE in 180 secs
Feb 24 08:11:30 dnd-test-esc-5-10-0-14-ha-1 esc-snmptrapd-precond.sh[3973]: Node becomes
 ACTIVE and /opt/cisco/esc/esc_database is mounted
Feb 24 08:11:31 dnd-test-esc-5-10-0-14-ha-1 systemd[1]: Started Simple Network Management
 Protocol (SNMP) Trap Daemon..
```

To start the service in standby node, use the following command:

```
systemctl --no-block start snmptrapd
```

Note In future, if the configuration files */opt/cisco/esc/esc_database /snmptrapd.conf* needs to be changed, make the changes in both active and standby nodes and restart snmptrapd on both the nodes.

- If ESC VM is deployed on an OpenStack environment,
  - Configure the security group of ESC VM in CVIM to allow incoming traffic in UDP port 162
  - Following is the OpenStack CLI command to add a rule for the ESC's security group,
    1. Get the UUID of the security group, nova list-secgroup

2. Add a rule with the following attributes,

```
For IPv4:
openstack security group rule create --ingress --ethertype IPv4 --protocol UDP
 -dst-port 162 --remote-ip 0.0.0.0/0

For IPv6:
openstack security group rule create --ingress --ethertype IPv6 --protocol UDP
 --dst-port 162 --remote-ip ::/0
```

- If ESC VM is deployed on the KVM environment, then make sure to allow incoming traffic on UDP port 162 in the KVM host's iptables and in KVM network filters if used.

3. Log files

snmptrapd:

All incoming SNMP traps received by snmptrapd are logged here:

```
/var/log/esc/esc-snmptrapd.log
```

esc-process-traps.py:

All traps processed by ESC script that is esc-process-traps.py are logged here

```
/var/log/esc/esc-process-traps.log
```

- Sample SNMP traps generated by CVIM and sent to ESC when nova-compute service on a particular host goes DOWN:

```
Agent Address: 0.0.0.0
 Agent Hostname: <UNKNOWN> (UDP: [172.19.243.21]:45651->[171.10.1.164]:162)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v2c, community public
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.855.0.1
SNMPv2-SMI::enterprises.9.9.855.1.1 = STRING: "soltest-cvim-mpod2"
SNMPv2-SMI::enterprises.9.9.855.1.2 = Hex-STRING: 07 E7 01 05 04 20 1A 07 2B 00 00
SNMPv2-SMI::enterprises.9.9.855.1.3 = STRING: "soltest-compute-5"
SNMPv2-SMI::enterprises.9.9.855.1.4 = STRING: "cp_hypervisor"
SNMPv2-SMI::enterprises.9.9.855.1.5 = INTEGER: 3
SNMPv2-SMI::enterprises.9.9.855.1.6 = INTEGER: 4
SNMPv2-SMI::enterprises.9.9.855.1.7 = STRING: "Hypervisor is down on host
soltest-compute-5"
```

- Sample SNMP traps generated by CVIM and sent to ESC, when nova-compute service on a particular host goes UP:

```
Agent Address: 0.0.0.0
 Agent Hostname: <UNKNOWN> (UDP: [172.19.243.21]:45508->[171.10.1.164]:162)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v2c, community public
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.855.0.2
SNMPv2-SMI::enterprises.9.9.855.1.1 = STRING: "soltest-cvim-mpod2"
SNMPv2-SMI::enterprises.9.9.855.1.2 = Hex-STRING: 07 E7 01 05 04 18 1A 07 2B 00 00
SNMPv2-SMI::enterprises.9.9.855.1.3 = STRING: "soltest-compute-5"
SNMPv2-SMI::enterprises.9.9.855.1.4 = STRING: "cp_hypervisor"
```

```
SNMPv2-SMI::enterprises.9.9.855.1.5 = INTEGER: 3
SNMPv2-SMI::enterprises.9.9.855.1.6 = INTEGER: 4
SNMPv2-SMI::enterprises.9.9.855.1.7 = STRING: "Cleared: Hypervisor is down on host
soltest-compute-5"
```

4. ESC action on receiving SNMP trap:

- Snmptrapd checks whether the received trap's OID matches with configured OIDs in snmptrapd.conf file.
- If the OID matches, then ESC invokes the script configured against that SNMP trap OID

  For example: esc-process-traps.py
- ESC script esc-process-traps.py performs the following actions:
  - Gets the severity of the received trap from SNMPv2-SMI::enterprises.9.9.855.1.5. If the received trap's severity is greater than the configured severity handle-trap-severity, then ignore the trap. If not, proceed to the next step.
  - Gets the CVIM hostname from the trap that is from SNMPv2-SMI::enterprises.9.9.855.1.3
  - Checks the host-recovery-time configured to confirm whether the recovery for that host was already triggered.
  - If recovery for the host is already triggered in the last host-recovery-time minutes, then ignore the trap. If not, proceed to the next step.
  - Gets the list of VNFs that are deployed on that host from the ESC database. ESC ignores other VMs under the host which are not managed by ESC.
  - For each VNF,

    Get the configured recovery policy for the VNF using ConfD CLI

    If recovery policy is configured for the VNF, trigger the recovery as per the configured policy

    If recovery policy is not configured for the VNF, trigger recovery with MIGRATE_ONLY policy

    If a recovery policy is configured with MANUAL recovery type, then do nothing.

Note The esc-process-traps.py script does not log any notifications in ESC's *yangesc.log*. The script only logs the details in *esc-process-trap.log*

5. RPM Upgrade:

After the RPM upgrade, the following are the steps that need to be executed as a "root" user to enable this feature on existing ESC VMs in both Active and Standby nodes.

1. Log in to ESC VM
2. Install the following prerequisites for Python packages.

   If ESC VM cannot connect to the Internet, set the proxy details as follows,

```
export http_proxy="http://<proxy server>:<proxy port>"
export HTTP_PROXY=${http_proxy}
export HTTPS_PROXY=${http_proxy}
export https_proxy=${http_proxy}
```

```
export no_proxy=localhost,127.0.0.1
export NO_PROXY=${no_proxy}

pip3 install requests filelock jproperties
```

Note Make sure “no_proxy” is set for localhost.

3. Append the following line in /etc/sysconfig/snmptrapd

```
OPTIONS="-f -C -c /opt/cisco/esc/esc_database/esc-snmptrapd.conf -A -Lf
/var/log/esc/esc-snmptrapd.log"
```

4. Take a backup of the file /etc/selinux/targeted/src/policy/esc/esc_custom.te
5. Open the file /etc/selinux/targeted/src/policy/esc/esc_custom.te and add the **highlighted selinux policies** from the attached file.
6. Go to /etc/selinux/targeted/src/policy/esc/ folder and execute the following commands to apply the policies,

```
checkmodule -M -m -o esc_custom.mod esc_custom.te
semodule_package -o esc_custom.pp -m esc_custom.mod
semodule -i esc_custom.pp
```

7. Allow UDP traffic in firewall of ESC VM. Edit the /etc/sysconfig/iptables and add the line as mentioned,

```
# Accept SNMP traps
-A INPUT -p udp --dport 162 -j ACCEPT
```

Note This rule must be added before rejecting all traffic that is before the existing entry,

```
# Send UNMATCHED packets to LOG_INPUT_REJECT (log then reject)
-A INPUT -j LOG_INPUT_REJECT
```

Restart the iptables service using the following command,

```
systemctl restart iptables
```

Note For IPv6, add the above rule in */etc/sysconfig/ip6tables* and restart *ip6tables* service as follows,

```
systemctl restart ip6tables
```

8. Follow the steps under Configure ESC to listen to SNMP traps from CVIM in the document.

Notifications and Events

The following notifications are generated by the ESC during healing:

- VM_RECOVERY_INIT
- VM_RECOVERY_DEPLOYED
- VM_RECOVERY_UNDEPLOYED

- VM_RECOVERY_COMPLETE
- VM_RECOVERY_CANCELLED
- VM_RECOVERY_REBOOT
- VM_RECOVERY_MIGRATED

These notifications are generated based on the workflow. Each notification has details about the deployment for which the notification is triggered. All recovery starts with VM_RECOVERY_INIT and ends with VM_RECOVERY_COMPLETE.

During vm recovery, if the vm is back to normal within the recovery wait time, the VM_RECOVERY_CANCELLED notification is sent as there is no recovery action to be performed. If the recovery wait time expires, then the recovery action is triggered. After the recovery is complete, ESC sends the success or failure notification, for example, the VM_RECOVERY_REBOOT notification.

The following table lists the different scenarios and the notifications that are generated for every event:

Scenario	Notifications

ESC-NORTHBOUND Recovery Call Flow After VM Alive - Reboot	When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alive received. The following NETCONF notification is triggered:
	
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	
	After ESC receives VM down event, the following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	
	ESC performs hard reboot on the VM, and the VM alive event is received within the boot time.
	
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>SUCCESS</status>`
	
	ESC receives an error while attempting to recover through Reboot. The following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_COMPLETE</type>`
	
	`<status>FAILURE</status>`

Scenario	Notifications

ESC-NORTHBOUND Recovery Call Flow After VM Alive - Undeploy/Redeploy	When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alive received. The following NETCONF notification is triggered:
	
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	
	After ESC receives VM down event, the following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	
	ESC fails to recover the VM by *Reboot* and proceeds with recovery by *Undeploy* and then *Redeploy*.
	
	It unsets monitoring and un-deploys the VM.
	
	The following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_UNDEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	ESC deploys VM and sets KPI to monitor VM Alive event and triggers the following NETCONF notifications:
	
	`<type>VM_RECOVERY_DEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	ESC receives a VM Alive event and triggers the following NETCONF notifications:
	
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>SUCCESS</status>`

Scenario	Notifications

ESC-NORTHBOUND Recovery Call Flow Multiple Recovery Attempts	When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alive received. The following NETCONF notification is triggered:
	
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	
	After ESC receives VM down event, the following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	
	ESC fails to recover the VM by *Undeploy* and then *ReDeploy* until it receives a VM Alive event. It keeps attempting the recovery for a specified boot time until the maximum attempts of recovery is reached.
	
	It un-sets monitoring and un-deploys the VM.
	
	The following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_UNDEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	ESC deploys VM and sets KPI to monitor VM Alive event.
	
	The following NETCONF notifications is triggered:
	
	`<type>VM_RECOVERY_DEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	ESC receives a VM Alive event and triggers the following NETCONF notifications:
	
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>SUCCESS</status>`

Scenario	Notifications

ESC-NORTHBOUND Recovery Call Flow Before VM Alive - Undeploy/Redeploy	When Northbound places a deploy request to ESC, ESC deploys VMs and sets KPI to monitor on all VM Alive received.
	ESC does not receive a VM Alive event after the deployment. Recovery is performed by *Undeploying* and *Redeploying* the VM.
	The following NETCONF notification is triggered:
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	ESC un-sets the monitoring and un-deploys the VM.
	The following NETCONF notification is triggered:
	`<type>VM_RECOVERY_UNDEPLOYED</type>`
	`<status>SUCCESS</status>`
	ESC deploys VM and sets KPI to monitor VM Alive event and triggers the following NETCONF notifications:
	`<type>VM_RECOVERY_DEPLOYED</type>`
	`<status>SUCCESS</status>`
	ESC receives a VM Alive event and triggers the following NETCONF notifications:
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>SUCCESS</status>`

Scenario	Notifications

Error Path For ESC-NORTHBOUND Recovery Call Flow After VM Alive - Undeploy/ReDeploy	When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alives received. The following NETCONF notification is triggered:
	
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	
	After ESC receives VM down event, the following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	
	ESC fails to recover the VM by *Reboot* and proceeds with recovery by *Undeploy* and then *Redeploy*.
	
	It un-sets monitoring and un-deploys the VM.
	
	The following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_UNDEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	If ESC receives an error or if the maximum attempts for recovery is reached.
	
	The following NETCONF notifications is triggered:
	
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>FAILURE</status>`

Scenario	Notifications

Error Path For ESC-NORTHBOUND Recovery Call Flow Before VM Alive - Undeploy/Redeploy	When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alives received. The following NETCONF notification is triggered:
	
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	
	After ESC receives VM down event, the following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	
	ESC un-sets monitoring and un-deploys the VM. Recovery is performed by *Undeploy* and then *Redeploy*.
	
	The following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_UNDEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	If ESC receives an error or if the maximum attempts for recovery is reached.
	
	The following NETCONF notifications is triggered:
	
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>FAILURE</status>`
	
	`<type>SERVICE_ALIVE</type>`
	`<status>FAILURE</status>`

ESC-NORTHBOUND Recovery Call Flow After VM Alive -VM_RECOVERY_CANCELLED	When Northbound places a deploy request to ESC, ESC deploys VMs and sets KPI to monitor all VM Alive notifications received. The following NETCONF notification is triggered:
	
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	
	After ESC receives VM down event, the following NETCONF notification is triggered:
	
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	
	During the recovery wait time, if VM is back to normal, then the VM_RECOVERY_CANCELLED notification is sent. Recovery action is not performed.
	
	`<type>VM_RECOVERY_CANCELLED</type>`
	`<status>SUCCESS</status>`

Scenario	Notifications

ESC-NORTHBOUND Recovery Call Flow After VM Alive - Reboot	When Northbound places a deploy request to ESC, ESC deploys VMs and sets KPI to monitor all VM Alive notifications received. The following NETCONF notification is triggered:
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	After ESC receives VM down event, the following NETCONF notification is triggered:
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	ESC performs hard reboot on the VM and sends reboot notification.
	`<type>VM_RECOVERY_REBOOT</type>`
	`<status>SUCCESS</status>`
	And the VM alive event is received within the boot time.
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>SUCCESS</status>`

Error Path For ESC-NORTHBOUND Recovery Call Flow After VM Alive - Reboot	When Northbound places a deploy request to ESC, ESC deploys VMs and sets KPI to monitor on all VM Alive notifications received. The following NETCONF notification is triggered:
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	After ESC receives VM down event, the following NETCONF notification is triggered:
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	Then ESC sends reboot notification.
	`<type>VM_RECOVERY_REBOOT</type>`
	`<status>FAILURE</status>`
	ESC receives an error while attempting to recover through *Reboot*.
	The following NETCONF notification is triggered:
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>FAILURE</status>`