Virtual Network Function Operations

VNF Operations

You can start, stop, and reboot VNFs. You can perform start, stop, and reboot operations using the RESTful interface.

You require a payload for VNF operations:

```
POST ESCManager/v0/{internal_tenant_id}/deployments/service/{internal_deployment_id}
```

Example,

```
<?xml version='1.0' encoding='UTF-8'?>
<service_operation xmlns='urn:ietf:params:xml:ns:netconf:base:1.0'>
   <operation>stop</operation>
</service_operation>
```

You must mention start, stop, or reboot in the operation field.

- Start VNF: Starts all VMs, enables monitoring, and reassigns thresholds according to the KPI details. The VMs start running and move to VM_ALIVE_STATE. The service is in service_active_state. Only undeploy can interrupt the start VNF workflow.
- Stop VNF: After the service is stopped, monitoring is disabled and all the VM services are stopped. The VMs are no longer available. The service is in service_stopped_state. VM is in shutoff_state. You can't perform any recovery, scale out, scale in. You can only undeploy the VNFs.
- Reboot VNF: Disables monitoring, and reboots all VMs. It stops and then starts in OpenStack, enables monitoring, and reassigns thresholds according to the KPI details. The VM is in VM_ALIVE_STATE and the service is in service_alive_state. Only undeploy can interrupt the reboot operation.

You can't start monitoring a VNF which is already running. After a reboot, logging back into the VM must indicate the reboot, update, and monitoring details. It must also indicate recovery.

VM Operations

Similar to VNF operations, you can start, stop, and reboot, resize and migrate individual VMs.

You require a payload for VM operations:

```
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>stop</operation>
   <force>true/false</force>
</vm_operation>
```

You must mention start, stop, or reboot, resize, or migrate in the operation field.

VM Resize

Note The VM Resize feature is available from the 5.7 release.

The VM resize feature in ESC provides flexibility in resizing the VMs once deployed by changing the flavor per VM group.

Changing the VM Flavor for Existing Deployment

To change the flavor of a VM, perform a service update request through the Netconf or REST API.

When a resize request is made from the ESC, the ESC stops monitoring the VM and makes the appropriate API request to OpenStack to resize the VM.

When OpenStack accepts the resize operation, the VM moves to the “VERIFY_RESIZE” state in OpenStack.

You can send a CONFIRM_RESIZE or REVERT_RESIZE request from ESC. Once the VM moves to the ACTIVE state, the ESC monitors the VM.

The REST API and RPC command in *esc_nc_cli supports* the confirm or revert requests.

Pre-requisite:

- The new flavor, for example, SolTest-Cirros-Flavor-v2 to which the VM resize needs to be done must be present in OpenStack.

Service Update Request for Change Flavor:

- For Netconf:

  esc_nc_cli edit-config sample_deployment_update.xml

- For REST API:

  PUT /v0/deployments/{internal_deployment_id}

ESC NETCONF:

Changing [*VM - ACTIVE* to *VERIFY_RESIZE* state]

The following example shows the *sample_deployment_update.xml* for VM resize service update request for NETCONF and REST API. In this xml file, you needs to modify the existing flavor name to a new flavor name for example `SolTest-Cirros-Flavor-v2` that is present in OpenStack.

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
    <tenants>
        <tenant>
            <name>admin</name>
            <deployments>
                <deployment>
                    <name>Cirros-VNF</name>
                    <vm_group>
                        <name>SolTest-Cirros-VM-Group</name>
                        <bootup_time>60</bootup_time>
                        <recovery_wait_time>0</recovery_wait_time>
                        <image>SolTest-Cirros-Image</image>
                        <flavor>SolTest-Cirros-Flavor-v2</flavor>
                        <interfaces>
                            <interface>
                                <nicid>0</nicid>
                                <network>SolTest-Network</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/>
                    </vm_group>
                </deployment>
            </deployments>
        </tenant>
    </tenants>
</esc_datamodel>
```

Upon successful deployment, an XML payload returns with *<ok/>*.

Upon unsuccessful deployment, a validation error or OpenStack API error with an appropriate error message returns.

ESC REST API:

An HTTP PUT operation specifies the *ESCManager* API with a deployment XML payload passed with an updated flavor name inside the tag.

For Example: *<flavor>UpdateflavorName</flavor>.*

PUT:/ESCManager/v0/deployments/{internal_deployment_id}

Upon successful deployment, an HTTP 200 code returns.

Upon an unsuccessful deployment, a validation error or OpenStack API error, along with an appropriate HTTP error code, and error messages return.

Notifications:

- ESC sends the *SERVICE_UPDATED* notification, once the request in OpenStack VIM for resizing the operation is accepted.

  Sample Notification sent by ESC for VM RESIZE request:

```
2022-04-11 05:45:47.232 INFO
2022-04-11 05:45:47.232 INFO  ===== SEND NOTIFICATION STARTS =====
2022-04-11 05:45:47.232 INFO  Type: SERVICE_UPDATED
2022-04-11 05:45:47.232 INFO  Status: SUCCESS
2022-04-11 05:45:47.232 INFO  Status Code: 200
2022-04-11 05:45:47.232 INFO  Status Msg: Service group update completed successfully
2022-04-11 05:45:47.233 INFO  Tenant: admin
2022-04-11 05:45:47.233 INFO  Deployment ID: 84779a3b-01e0-4eeb-9b1c-fe30317b6bb5
2022-04-11 05:45:47.233 INFO  Deployment name: Cirros-VNF
2022-04-11 05:45:47.233 INFO  =====  SEND NOTIFICATION ENDS  =====
```

- ESC sends the *VM_VERIFY_RESIZE* notification, once the VM goes to VERIFY_RESIZE state.

  Sample Notification sent by ESC for VM RESIZE request:

```
2022-04-11 05:46:09.425 INFO
2022-04-11 05:46:09.425 INFO  ===== SEND NOTIFICATION STARTS =====
2022-04-11 05:46:09.425 INFO  Type: VM_VERIFY_RESIZE
2022-04-11 05:46:09.425 INFO  Status: SUCCESS
2022-04-11 05:46:09.426 INFO  Status Code: 200
2022-04-11 05:46:09.426 INFO  Status Msg: VM Resize request processed
2022-04-11 05:46:09.426 INFO  VM State: VERIFY_RESIZE
2022-04-11 05:46:09.426 INFO  Tenant ID: 31fb89f6647b4c109efda76479c07332
2022-04-11 05:46:09.426 INFO  Deployment ID: 84779a3b-01e0-4eeb-9b1c-fe30317b6bb5
2022-04-11 05:46:09.426 INFO  Deployment name: Cirros-VNF
2022-04-11 05:46:09.426 INFO  VM group name:SolTest-Cirros-VM-Group
2022-04-11 05:46:09.426 INFO  VM Source:
2022-04-11 05:46:09.426 INFO      VM ID: 3b24cef1-5cf4-4e9c-b98e-682bfbdf8240
2022-04-11 05:46:09.426 INFO     VM Name:cirros-vm_0_357b379e-bac5-4c6a-a214-0c0b1343ad4d
2022-04-11 05:46:09.427 INFO      VM Name (Generated):
cirros-vm_0_357b379e-bac5-4c6a-a214-0c0b1343ad4d
2022-04-11 05:46:09.427 INFO      VIM ID: default_openstack_vim
2022-04-11 05:46:09.427 INFO      VIM Project: admin
2022-04-11 05:46:09.427 INFO      VIM Project ID: 31fb89f6647b4c109efda76479c07332
2022-04-11 05:46:09.427 INFO      Host ID:
549a38c4fd21432060486a69bf9f82839f126adbe7f02e9cd2f4f3b6
2022-04-11 05:46:09.427 INFO      Host Name: esc-soltest-116
2022-04-11 05:46:09.427 INFO  =====  SEND NOTIFICATION ENDS  =====
2022-04-11 05:46:09.427 INFO
```

Confirm or Revert Request:

ESC NETCONF:

To confirm or revert the request, that is to change [VM - *VERIFY_RESIZE* to *ACTIVE* state], use the following RPC command.

esc_nc_cli Impaction <action: CONFIRM_RESIZE or *REVERT_RESIZE><Generated VM Name>*

For Example:

esc_nc_cli vm-action CONFIRM_RESIZE cirros-vm_0_357b379e-bac5-4c6a-a214-0c0b1343ad4d

Upon successful action, an XML payload returns with *<ok/>*

Upon unsuccessful action that is validation error or OpenStack API error, an appropriate error message returns.

ESC REST API:

An HTTP POST operation specifies the *ESCManager* API with an XML payload passed with the operation name inside the tag.

For Example,*<operation>confirm_resize</operation>* or <operation>*revert_resize</operation>*

POST:/ESCManager/v0/SystemAdminTenantId/deployments/vm/{Generated_vm_name}

The following shows the POST Sample payload:

```
[admin@dnd-admin-5-7-0-52-vtp-scan ~]$ cat operation.xml
<vm_operation xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <operation>confirm_resize</operation>
</vm_operation>
```

To revert VM resize, use the following payload:

```
<vm_operation xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <operation>revert_resize</operation>
</vm_operation>
```

Upon successful deployment, an HTTP 200 code returns.

Upon an unsuccessful deployment that is for a validation error or OpenStack API error, an appropriate HTTP error code, and error messages returns.

Notifications:

To confirm resize:

- ESC sends the VM_CONFIRM_RESIZE_INIT notification once the confirm request in OpenStack VIM is accepted.
- ESC sends the VM_CONFIRM_RESIZE_COMPLETE notification once it receives confirmation that the operation completed successfully in OpenStack VIM.
- ESC sends the VM_ALIVE notification once the monitor is set and the VM is in the ACTIVE state.

To revert resize:

- ESC sends the VM_REVERT_RESIZE_INIT notification once the confirm request in OpenStack VIM is accepted..
- ESC sends the VM_REVERT_RESIZE_COMPLETE notification once it confirms that the operation completes successfully in OpenStack VIM.
- ESC sends the VM_ALIVE notification once you set the monitor, and the VM is in the ACTIVE state.

Note In ESC, the flavor change request is at the VM Group level. But a VM Group can have more than one VM. This resize feature works only when the VM Group has a single VM in it.

VM Migrate

Note The VM Migrate feature is available from the 5.7 release.

ESC provides flexibility in migrating the VMs. Migration is of two types:

- Live Migration
- Cold Migration

Live migration transfers a live VM from one host to another while cold migration transfers a powered-off VM from one host to another.

VM Live Migration:

Use the following commands to perform VM live migration from one host to another:

Note The host parameter is optional. If the host parameter is not included, the NOVA scheduler selects the host.

Example for REST API:

```
POST /v0/{internal_tenant_id}/deployments/migrate-vm/{vm_name}
curl -X POST
'http://localhost:8080/ESCManager/v0/test-tenant/deployments/migrate-vm/test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008'
 \
-H 'Content-Type: application/xml' \
-H 'Callback: http://localhost:9009/callback' \
-H 'Callback-ESC-Events: http://localhost:9009/event' \
--data-raw '<vm_migrate_operation xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <operation>migrate</operation>
   <migrationType>live</migrationType>
   <host>Host-24</host>
</vm_migrate_operation>'
```

Example for RPC Payload:

```
<vmMigrate xmlns=http://www.cisco.com/esc/esc>
   <vmName>test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008</vmName>
   <migrationType>live</migrationType>
   <host>Host-24</host>
</vmMigrate>
```

Example for NETCONF:

```
esc_nc_cli supports VM migration as follows:
Migrate VM Action : migrate-vm-action <vm-name> <migration-type> [block-migration] [host]
                               migration-type : = live|cold
                               block-migration : = true|false (live migration only)
```

Example for esc_nc_cli live migrate:

```
esc_nc_cli migrate-vm-action test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008 live
```

Example for esc_nc_cli live migrate with host:

```
esc_nc_cli migrate-vm-action test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008 live Host-22
```

Example for esc_nc_cli live migrate with block-migration:

```
esc_nc_cli migrate-vm-action test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008 live true
```

Example for esc_nc_cli live migrate with host and block-migration:

```
esc_nc_cli migrate-vm-action test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008 live true
Host-22
```

Notifications:

ESC sends the following notifications once the live migration is successful:

```
VM_MIGRATE_INIT
VM_MIGRATED
VM_MIGRATE_COMPLETE
```

VM Cold Migration:

Use the following commands to perform VM cold migration from one host to another:

Note The host parameter is optional. If the host parameter is not included, the NOVA scheduler selects the host.

Example for REST API:

```
POST /v0/{internal_tenant_id}/deployments/migrate-vm/{vm_name}
curl -X POST
'http://localhost:8080/ESCManager/v0/test-tenant/deployments/migrate-vm/test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008'
 \
-H 'Content-Type: application/xml' \
-H 'Callback: http://localhost:9009/callback' \
-H 'Callback-ESC-Events: http://localhost:9009/event' \
--data-raw '<vm_migrate_operation xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <operation>migrate</operation>
   <migrationType>cold</migrationType>
   <host>Host-24</host>
</vm_migrate_operation>'
```

Example for RPC Payload:

```
<vmMigrate xmlns="http://www.cisco.com/esc/esc">
 <vmName>test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008</vmName>
 <migrationType>cold</migrationType>
 <host>Host-24</host>
</vmMigrate>
```

Example for NETCONF:

Example for esc_nc_cli cold migrate:

```
esc_nc_cli migrate-vm-action test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008 cold
```

Example for esc_nc_cli cold migrate with host:

```
esc_nc_cli migrate-vm-action test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008 cold Host-22
```

Confirm or revert the migration once ESC sends the VM_VERIFY_RESIZE notification.

To confirm or revert, use either of the following commands: esc_nc_cli (Netconf) command or REST API command

```
VM Action : vm-action <action> <generated vm name>
```

```
action:=STOP|START|REBOOT|DISABLE_MONITOR|ENABLE_MONITOR|CONFIRM_RESIZE|REVERT_RESIZE
```

Example for esc_nc_cli to confirm the migration or resize:

```
esc_nc_cli vm-action CONFIRM_RESIZE test-dep_g1_0_6be93ee9-f7fb-473e-a3f6-eca281802008
```

Once the action is triggered, the following notifications are received:

- VM_CONFIRM_RESIZE_INIT
- VM_CONFIRM_RESIZE_COMPLETE
- VM_ALIVE

REST API to confirm the migration or resize:

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

Sample payload:

```
<vm_operation xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <operation>confirm_resize</operation>

</vm_operation>
```

Notifications:

ESC sends the following notifications once the cold migration is successful:

```
VM_MIGRATE_INIT
VM_MIGRATED
VM_VERIFY_RESIZE

VM_CONFIRM_RESIZE_INIT
VM_CONFIRM_RESIZE_COMPLETE
VM_ALIVE
```

or

```
VM_REVERT_RESIZE_INIT
VM_REVERT_RESIZE_COMPLETE
VM_ALIVE
```

VNF Backup and Restore Operations

This section describes VNF backup and restore operations using VM snapshots.

VNF Backup Operations

Manage VM Snapshots

ESC creates a snapshot, which is an image (and a volume in certain circumstances) on OpenStack VIM. ESC APIs manage the snapshots of VNFs managed by ESC. ESC supports the below main snapshot operations:

- Create VM snapshots
- List VM snapshots
- Delete VM snapshots

ESC executes the VM snapshot operations using ESC REST API with HTTP and HTTPS protocols. The create and delete VM snapshot operations are supported through `esc_nc_cli` script. Both Netconf and REST API notifications are generated during the various stages of the snapshot operations for create and delete operations.

Note The VM snapshot operations are supported on the OpenStack VIM only.

Create VM Snapshot

You can create a snapshot (using the REST API or the esc_nc_cli script) from any VNF that is managed by the ESC VM. A snapshot can only be created for active or stopped VNFs, which translate to the ESC VM status of VM_ALIVE or VM_STOPPED respectively. You can specify the snapshot name in the payload of the API invocations. A unique ID is generated along with the snapshot name, which is used as a reference when specifying the ESC snapshot operation payloads, if the snapshot name is not unique. A snapshot (an image) is created on OpenStack. In case of a VNF which uses a bootable volume, a *volume snapshot* is also created on OpenStack.

Create Snapshot Using REST API

To create a snapshot, specify an HTTP POST operation to the ESCManager API:

```
POST: /ESCManager/v0/<tenant-id>/deployments/snapshot-vm/<generated-vm-name>
```

The payload must contain *operation* and *name* values, and the operation value must be **snapshot**.

```
operation: snapshot
name: <snapshot-name>
```

If successful, an HTTP 200 code is returned, and there is no payload.

If unsuccessful (validation error or OpenStack API error), an appropriate HTTP error code and error message are returned.

The following shows the API invocation to create a snapshot:

```
[admin@localhost]$ cat snapshot.json
{
    "operation": "snapshot",
    "name": "my-snapshot-name"
}

[admin@localhost]$ curl -X POST -d @snapshot.json -H 'Content-Type: application/json' -H
'callback: http://localhost:9009' -H 'Callback-ESC-Events: http://localhost:9009'
"http://localhost:8080/ESCManager/v0/snapshot-tenant/deployments/snapshot-vm/new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba"
```

Create Snapshot Using `esc_nc_cli script`

To create a snapshot using the `esc_nc_cli` script, fixed parameters can be passed specifying the generated VM name and operation:

```
VM Backup Action : vm-backup-action vm-name backup-name [<action-type>] [<xmlfile>]
    action-type := SNAPSHOT|EXPORT
```

The optional action-type parameter defaults to SNAPSHOT if not specified. The following shows the script invocation to create a snapshot:

```
[admin@localhost]$ esc_nc_cli vm-backup-action
new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba my-snapshot-name SNAPSHOT
VM Backup Action
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=esc-nc-admin
--privKeyFile=/home/admin/.ssh/confd_id_rsa --privKeyType=rsa
--rpc=/tmp/tmp_esc_nc_cli.c8d9kAjcGf
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <ok/>
</rpc-reply>
```

If successful, an XML payload with a single <ok/> element is returned. If unsuccessful (validation error or OpenStack API error), an appropriate error message is returned.

Notes

Note the following for both ESC REST API and the `esc_nc_cli`:

- Snapshot names must be less than or equal to 255 characters in length.
- The generated VM name must be valid.
- action-type must be SNAPSHOT or EXPORT (`esc_nc_cli` only).
- xmlfile - if specified - must contain a valid XML document (`esc_nc_cli` only).

Notifications

Both Netconf notifications and ESC REST callback messages are sent during the create snapshot operation.

Table 1:

Notification (NETCONF or ESC callback)	When the notification is sent
VM_BACKUP_INIT	When the API is invoked and validation passed.
VM_BACKUP_CREATED	When OpenStack has successfully received and validated the snapshot create request.
VM_BACKUP_COMPLETE	When OpenStack has finished the snapshot create request operation, and it was either a success, or an error occurred.

The following shows an example VM_BACKUP_CREATED successful Netconf notification (other notifications are similar):

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2021-09-14T12:18:39.836+00:00</eventTime>
    <escEvent xmlns="http://www.cisco.com/esc/esc">
        <status>SUCCESS</status>
        <status_code>202</status_code>
        <status_message>Snapshot is now active.</status_message>
        <depname>snapshot-deployment-name</depname>
        <tenant>snapshot-tenant</tenant>
        <tenant_id>7d61b5de73874f88a458d486759a9b83</tenant_id>
        <depid>ae0bea05-9630-4d17-a9e7-926f1f625dc7</depid>
        <vm_group>snapshot-group</vm_group>
        <vm_source>
            <vmid>1773914c-20cd-4f50-b337-1e46be2cf295</vmid>
           <vmname>new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba</vmname>

<generated_vmname>new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba</generated_vmname>

            <vim_id>default_openstack_vim</vim_id>
            <vim_project>snapshot-tenant</vim_project>
            <vim_project_id>7d61b5de73874f88a458d486759a9b83</vim_project_id>
            <hostid>95503baadeccce2d33e5d924322390aee9d30c6ed24043284bf46984</hostid>
            <hostname>pf-ucs-27</hostname>
        </vm_source>
        <event>
            <type>VM_BACKUP_CREATED</type>
        </event>
    </escEvent>
</notification>
```

For the failure cases, Netconf notifications and ESC REST callback messages are still generated, but:

- the <status> value will be *FAILURE*,
- the <status_code> will be *500*, and
- the <status_message> will be an appropriate message, either internally generated or sent back from OpenStack.

List Snapshot

You can list the snapshots using the ESC REST API. Only the snapshots managed by ESC can be listed. A subset of the snapshot data can be specified as query parameters to reduce the number of snapshots returned. The returned snapshot data can be in XML or JSON format, controlled by the HTTP *Accept* header. The Accept header value defaults to XML if not specified.

Only the ESC REST API supports listing snapshots. The `esc_nc_cli` does not supported listing ESC managed entities.

List Snapshot Usig ESC REST API

To list snapshots, an HTTP GET operation can be specified to the ESCManager API:

```
GET: /ESCManager/v0/snapshots
```

Optional query parameters can also be specified: *internalTenantId*, *generatedVMName*

An HTTP 200 code is always returned regardless of how many snapshots are returned.

The following shows the API invocation to list a snapshot for a specific internal tenant id and generated VM name:

```
[admin@localhost]$ curl -X GET --header "Accept: application/xml"
"http://localhost:8080/ESCManager/v0/snapshots?internalTenantId=snapshot-tenant&generatedVMName=new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba"
 | xmllint --format -

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<snapshots>
    <snapshot xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <id>7813c20b-94b6-492b-ae74-0bd36c1168dc</id>
        <name>my-snapshot-name</name>
        <creation_start_date>2021-07-20T11:26:47.532Z</creation_start_date>
        <creation_end_date>2021-07-20T11:27:53.139Z</creation_end_date>
        <status>available</status>
        <status_message>Snapshot image for VM [gen_vm_name] is active.</status_message>

<gen_vm_name>new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba</created_from_generated_vm_name>

        <vim_id>default_openstack_vim</vim_id>
        <tenant>snapshot-tenant</tenant>
        <bootable_volume_snapshot_id:c3cd5d13-63bf-49f0-b864-df3bc024d5e4/>
    </snapshot>
</snapshot>
```

Note There are no notifications generated when this API is invoked.

Delete Snapshots

Snapshot created by ESC can be deleted using REST API or the `esc_nc_cli` script. Only snapshots managed by the current ESC VM can be deleted, and only one snapshot can be deleted at a time. If successful, the snapshot is deleted from ESC, and also the related image and volume snapshot (if applicable) within OpenStack.

Delete Snapshots Using REST API

To delete a snapshot previously created via ESC, an HTTP DELETE operation can be specified to the ESCManager API:

```
DELETE: /ESCManager/v0/snapshots/<snapshot-id|snapshot-name>
```

Either a snapshot id can be passed, or a snapshot name. If successful, an HTTP 200 code is returned, and there is no payload. If unsuccessful (validation error or OpenStack API error), an appropriate HTTP error code and error message are returned. The following shows the API invocation to delete a snapshot:

```
[admin@localhost]$ curl -X DELETE -H 'callback: http://localhost:9009' -H
'Callback-ESC-Events: http://localhost:9009'
"http://localhost:8080/ESCManager/v0/snapshots/7813c20b-94b6-492b-ae74-0bd36c1168dc"
```

Delete Snapshot Using `esc_nc_cli`

To delete a snapshot using the `esc_nc_cli` script, only a snapshot id or snapshot name needs to be passed as the single parameter:

```
Snapshot Action : snapshot-action <snapshot-id|snapshot-name>
```

The following shows the script invocation to create a snapshot:

```
[admin@localhost]$ esc_nc_cli snapshot-action delete my-snapshot-name
```

or

```
[admin@localhost]$ esc_nc_cli snapshot-action delete my-snapshot-name-1
<?xml version="1.0" encoding="UTF-8"?>
<error xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <error_code>404</error_code>
    <error_message>Snapshot image [my-snapshot-name-1] not found.</error_message>
</error>
```

If successful, an XML payload with a single <ok/> element is returned. If unsuccessful (validation error or OpenStack API error), an appropriate error message is returned.

Note the following for both ESC REST API and `esc_nc_cli`.

- The snapshot id or snapshot name must be valid.

- If a snapshot name is specified, it must be unique.

Notifications

Both Netconf notifications and ESC REST callback messages are sent during the delete snapshot operation.

The notifications are:

Table 2:

Notification (Netconf and/or ESC Callback)	When the notification is sent
VM_SNAPSHOT_DELETING	Sent upon successful submission and validation.
VM_SNAPSHOT_DELETED	When OpenStack has finished the snapshot delete operation, and it was either a success, or an error occurred. If the snapshot has a volume snapshot along with the image snapshot, then the notification will not be sent until both are deleted.

The following shows an example VM_SNAPSHOT_DELETED successful Netconf notification (other notifications are similar):

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2021-09-14T12:18:39.836+00:00</eventTime>
    <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Snapshot image [2ffadd36-3b41-4c13-a9d6-a48c07764d1a] has been
deleted.</status_message>
    <depname>snapshot-deployment-name</depname>
    <tenant>snapshot-tenant</tenant>
    <tenant_id>7d61b5de73874f88a458d486759a9b83</tenant_id>
    <depid>ae0bea05-9630-4d17-a9e7-926f1f625dc7</depid>
    <vm_group>snapshot-group</vm_group>
    <vm_source>
        <vmid>1773914c-20cd-4f50-b337-1e46be2cf295</vmid>
        <vmname>new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba</vmname>

<gen_vm_name>new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba</generated_vmname>

        <vim_id>default_openstack_vim</vim_id>
        <vim_project>snapshot-tenant</vim_project>
        <vim_project_id>7d61b5de73874f88a458d486759a9b83</vim_project_id>
        <hostid>95503baadeccce2d33e5d924322390aee9d30c6ed24043284bf46984</hostid>
        <hostname>pf-ucs-27</hostname>
    </vm_source>
    <event>
        <type>VM_SNAPSHOT_DELETED</type>
    </event>
    </escEvent>
</notification>
```

For the failure cases, Netconf notifications and ESC REST callback messages are still generated, but:

- the <status> value will be *FAILURE*,
- the <status_code> will be *500*, and

- the <status_message> will be an appropriate message, either internally generated or sent back from OpenStack.

VM Snapshot Polling Parameters

Vim Manager configuration properties can be used to control the interval between calls to OpenStack to check the status of the create and delete operations, along with the maximum number of minutes an operation is allowed before timing out with an error.

These properties are:

- vim.asyncpoller.snapshot.create.poll.secs # default 15, the number of seconds between polls
- vim.asyncpoller.snapshot.create.timeout.mins # default 20, maximum number of minutes for the create snapshot operation
- vim.asyncpoller.snapshot.delete.poll.secs # default 15, the number of seconds between polls
- vim.asyncpoller.snapshot.delete.timeout.mins # default 20, maximum number of minutes for the delete snapshot operation

The default values can be overridden by setting them in the application.properties file under */opt/cisco/esc/vimmanager/application.properties*, and restarting the vim manager service, as shown below:

```
[admin@localhost]$ sudo cat /opt/cisco/esc/vimmanager/application.properties
vim.asyncpoller.snapshot.create.poll.secs=5
vim.asyncpoller.snapshot.create.timeout.mins=10
vim.asyncpoller.snapshot.delete.poll.secs=10
vim.asyncpoller.snapshot.delete.timeout.mins=60

[admin@localhost]$ sudo escadm vimmanager restart
Stopping vimmanager service: [OK]
Starting vimmanager service: [OK]

[admin@localhost]$ sudo escadm vimmanager show
VimManager System Configurations.
{
    "ccp.pollRetries": "200",
    "ccp.pollRetryDelaySecs": "15",

    . . .

    "vim.asyncpoller.snapshot.create.poll.secs": "5",
    "vim.asyncpoller.snapshot.create.mins": "10",
    "vim.asyncpoller.snapshot.delete.poll.secs": "10",
    "vim.asyncpoller.snapshot.delete.timeout.mins": "60",

    . . .

    "vmware.ovftool.params": "--acceptAllEulas --disableVerification --noSSLVerify
--allowExtraConfig",
    "vmware.powerOnRetry": "8"
}
```

In an HA setup, the application properties file will need to be copied to both nodes.

Alternatively, the values can be set dynamically (although their values will not be persisted after a restart), using the escadm script:

```
[admin@loclhost] sudo escadm vimmanager set --config
vim.asyncpoller.snapshot.create.poll.secs=200
vim.asyncpoller.snapshot.create.timeout.mins=1
```

```
VimManager configuration [vim.asyncpoller.snapshot.create.poll.secs] has updated to [200].
VimManager configuration [vim.asyncpoller.snapshot.create.timeout.mins] has updated to [1].
```

Snapshots of VNFs with Bootable Volumes

If a snapshot is taken of an ESC managed VNF which has a boot volume, then both an image snapshot and a volume snapshot are created within OpenStack.

Note The image snapshot name will be the snapshot name specified in the snapshot payload. The *volume snapshot* name (if applicable) will be be prepended with *snapshot for* .

For example, if a snapshot is taken on an ESC VM of a VNF with a bootable volume, and the snapshot was named *my-snapshot-name*, then the following would hold true:

```
[admin@localhost]$ openstack volume snapshot list | grep my-snapshot-name
| 52a96891-f22d-4863-bb47-bd9442ca0cb1 | snapshot for my-snapshot-name | None | available
| 2 |
```

```
[admin@localhost]$ openstack image list | grep my-snapshot-name
| c8846c14-48e4-45db-88a0-f838fc3ac29d | my-snapshot-name | active |
```

Volume snapshots cannot be used directly either within ESC or natively on OpenStack in a restore operation: a bootable volume must be created from the snapshot first. ESC supports creating bootable volumes from volume snapshots. For more information, see VNF Restore Operations.

VNF Restore Operations

Snapshot of VNFs Without Bootable Volumes

If ESC takes a snapshot of a VNF with a non-bootable volume, then that snapshot is stored in OpenStack as a snapshot image, and that snapshot image can be used to restore from via ESC's service update functionality. A service update XML can be created identical to the original deployment XML but with the snapshot image name. Once this service update XML is deployed to ESC using the REST or `esc_nc_cli` interfaces, ESC will internally update its image name for the VNF, and upon the next redeployment (typically triggered manually using the REST or `esc_nc_cli` interfaces), a new deployment will be created with the new image.

Snapshots of VNFs With Bootable Volumes

If ESC takes a snapshot of a VNF with a bootable volume, then that snapshot is stored in OpenStack as both a snapshot image and a volume snapshot. The volume snapshot cannot be used directly within the restore process of a VNF, either by ESC or by OpenStack. A bootable volume must first be created from the volume snapshot which is considered out-of-band from ESC's perspective (i.e. not managed directly by ESC). Once the bootable volume is created and available, then it can be used to restore from via ESC's service update functionality. A service update XML can be created identical to the original deployment XML, but with a delete operation against the named original volume, and a create operation specifying the new, bootable volume. Once this service update XML is deployed to ESC using the REST or `esc_nc_cli` interfaces, ESC will automatically swap out the original volume for the new volume without the need for a VNF redeployment, thus preserving all OpenStack UUIDs and resources.

The original volume (which was detached from the VNF) will still remain in OpenStack and must be cleaned up manually.

Create Bootable Volume from Volume Snapshot

If ESC takes a snapshot of a VNF with a bootable volume, then a volume snapshot is created in OpenStack, and ESC can then be used to create a bootable volume from the volume snapshot. Creating a bootable volume from the volume snapshot is required for any restoration operations, either within ESC or the OpenStack APIs directly. The volume name can be specified in the payload of the ESC REST API invocation. The `esc_nc_cli` script does not support creating a bootable volume from a volume snapshot. Volume names do not have to be unique, as a unique ID is generated alongside the name which can be used as a reference when specifying ESC restore operation payloads. The end result of a successfully created volume from volume snapshot will be a new, bootable volume in OpenStack

Note The new, bootable volume on OpenStack is not managed by ESC. It is out-of-band and must be managed by an Orchestrator directly.

Create Bootable Volume from Volume Snapshot Using REST API

To create a volume from a volume snapshot, an HTTP POST operation can be specified to the snapshot endpoint of the ESCManager API:

```
POST: /ESCManager/v0/snapshots/<snapshot-id>/volumes
```

The payload must contain a *name* value which will be the new volume's name, but can *volume_type*, *multiattach* and *bootable* can be optionally specified.

```
operation: snapshot
name: <snapshot-name>
volume_type: <valid-volume-type>   # defaults to the OpenStack default volume type
multiattach: <true|false>          # defaults to false
bootable: <true|false>             # defaults to true
```

If successful, an HTTP 202 code is returned (indicating that the operation has bee successfully submitted to OpenStack), and there is no payload. If unsuccessful (validation error or OpenStack API error), an appropriate HTTP error code and error message are returned.

The following shows the API invocation to create a snapshot after listing it within ESC first to determine the snapshot id:

```
[admin@localhost]$ curl -s
"http://localhost:8080/ESCManager/v0/snapshots?internalTenantId=dave-2000" | xmllint --format
 -
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<snapshots>
    <snapshot xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <id>171ffa7d-8318-47d1-acab-b01db4501a39</id>
        <name>my-snapshot-name</name>
        <creation_start_date>2021-09-20T08:29:22.074+01:00</creation_start_date>
        <creation_end_date>2021-09-20T08:33:41.193+01:00</creation_end_date>
        <status>active</status>
        <status_message>Snapshot image for
[new-dep-4_new-gr_0_34b9da8a-af64-4452-a8a3-8972e23e4e98] is active.</status_message>

<created_from_generated_vm_name>new-dep-4_new-gr_0_34b9da8a-af64-4452-a8a3-8972e23e4e98</created_from_generated_vm_name>

        <vim_id>my-snapshot-vim</vim_id>
        <tenant>dave-2000</tenant>
        <volume_snapshot_id>c4548ba4-0480-4b42-8229-ad98de44b3ea</volume_snapshot_id>
    </snapshot>
</snapshots>
```

```
[admin@localhost]$ cat volume_from_volume_snapshot.json
{
    "name": "my-bootable-volume-from-snapshot-volume",
    "multiattach": true
}

[admin@localhost]$ curl -X POST -d @volume_from_volume_snapshot.json -H 'Content-Type:
application/json' -H 'callback: http://localhost:9009' -H 'Callback-ESC-Events:
http://localhost:9009'
"http://localhost:8080/ESCManager/v0/snapshots/171ffa7d-8318-47d1-acab-b01db4501a39/volumes"

[admin@localhost]$ openstack volume list | grep my-bootable-volume
| c4548ba4-0480-4b42-8229-ad98de44b3ea | my-bootable-volume-from-snapshot-volume | available
 | 2 | |
```

The ESC REST API applies the following validation:

- Volume names must be less than or equal to 255 characters in length.
- The snapshot id must be for a snapshot that ESC managed (i.e. it must be one of the snapshots that a *list snapshot* operation would return).
- *name* must be specified in the payload - all other attributes can optionally be set.
- Non-supported attribute names in the payload are ignored.
- The snapshot must have been taken for a VNF which had a bootable volume.

Notifications

Only ESC REST callback messages are generated for this operation.

Two callback messages are generated: VM_VOLUME_ACCEPTED_EVENT and VM_VOLUME_CREATED_EVENT.

Following is an example of the VM_VOLUME_CREATED_EVENT ESC REST callback message:

```
{
    "escTransactionId": "5acac790-9213-45c0-8fde-9dd7d3111fdb",
    "eventType": "VM_VOLUME_CREATED_EVENT",
    "eventSourceContext": null,
    "eventTargetContext": null,
    "message": "Create volume snaphot request completed",
    "stateMachineEventNBInfo": {
        "id": "de9440c0-1342-441d-a16e-5c8267231ae5",
        "message": {},
        "logNames": [],
        "keywords": {},
        "actionInfo": {},
        "stackTrace": ""
    },
    "escParameter": {
        "external_volume_id": "c3cd5d13-63bf-49f0-b864-df3bc024d5e4",
        "size": "2",
        "sizeunit": null,
        "bus": "virtio",
        "type": "LVM",
        "outOfBand": "false",
        "bootIndex": null,
        "name": "daves-ooband-bootable-volume-for-restore",
        "format": null,
        "deviceType": null,
        "storageLocation": null,
        "external_tenant_id": null,
        "internal_tenant_id": null,
        "internal_volume_id": null,
        "volid": null,
        "event_type": null,
        "image": null
    },
    "vmUpdateType": null,
    "requestDetails": null,
    "statusCode": "201",
    "notificationOnlyEvent": false
}
```

Polling Configuration Parameters

Vim Manager configuration properties can be used to control the interval between calls to OpenStack to check the status of the create volume operation, along with the maximum number of minutes an operation is allowed before timing out with an error.

These properties are:

- vim.asyncpoller.volume.create.poll.secs # default 15, the number of seconds between polls
- vim.asyncpoller.volume.create.timeout.mins # default 20, maximum number of minutes for the create volume operation

```
[admin@localhost]$ sudo cat /opt/cisco/esc/vimmanager/application.properties
vim.asyncpoller.volume.create.poll.secs=5
vim.asyncpoller.volume.create.timeout.mins=10

[admin@localhost]$ sudo escadm vimmanager restart
Stopping vimmanager service: [OK]
Starting vimmanager service: [OK]

[admin@localhost]$ sudo escadm vimmanager show
VimManager System Configurations.
{
    "ccp.pollRetries": "200",
    "ccp.pollRetryDelaySecs": "15",

    . . .

    "vim.asyncpoller.volume.create.poll.secs": "5",
    "vim.asyncpoller.volume.create.mins": "10",

    . . .

    "vmware.ovftool.params": "--acceptAllEulas --disableVerification --noSSLVerify
--allowExtraConfig",
    "vmware.powerOnRetry": "8"
}
```

Note In an HA setup, the application properties file will need to be copied to both nodes.

Managing Individual and Composite VNFs

An individual service consists of a single VNF. A coupled service or a composite VNF consists of several VMs of different types. The ESC interface receives VM interdependency information from the northbound system, and uses this information during VM and VNF creation, and life cycle management. Interdependency could include VM specific workflow in the group of VMs in a single VNF, VNF monitoring and scalability and so on.

Create, read, update and delete operations are allowed on the VMs. To add more VM instances to a deployed VNF using static IP, you must provide additional IP addresses into the static IP pool. If you are using an existing static IP deployment, the minimum number of VMs is altered.

If the new minimum value, which is the number of VMs is greater than the active VMs, a new VM is added to the service. If the value is greater than the max value, the update is rejected.