



Virtual Network Function Operations

- [VNF Operations, on page 1](#)
- [VNF Backup and Restore Operations, on page 2](#)
- [Managing Individual and Composite VNFs, on page 12](#)

VNF Operations

You can start, stop and reboot VNFs. Start, stop and reboot operations are performed using the RESTful interface.

A payload is required 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 the KPI details. The VMs start running and move to VM_ALIVE_STATE. The service will be in service_active_state. Only undeploy can interrupt the start VNF workflow.
- **Stop VNF:** Once the service is stopped, monitoring is disabled and all the VM services are stopped. The VMs are no longer available. The service will be in service_stopped_state. VM will be in shutoff_state. You cannot perform any recovery, scale out, scale in. You can only undeploy the VNFs.
- **Reboot VNF:** Disables monitoring, reboots all VMs, that is stop and then start in OpenStack, enables monitoring, and reassigns thresholds according to 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 cannot 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 individual VMs.

A payload is required 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 in the operation field.

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 three 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>
    </vm_source>
    <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>95503baadecce2d33e5d924322390aee9d30c6ed24043284bf46984</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 support listing ESC managed entities.

List Snapshot Using 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>
</snapshots>
```



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>
  </vm_source>
  <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>95503baadecce2d33e5d924322390aee9d30c6ed24043284bf46984</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@localhost] 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 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 been 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/volume"

[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 snapshot 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,
    "valid": 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.