# Configuring Interfaces



# Interface Configurations

The Interface configuration allows to choose various configuration for the interface including network, subnet, ip address, mac address, vim interface name, model, and so on.

This section describes these basic and advance interface configurations for Elastic Services Controller (ESC) and procedures to configure these.

## Basic Interface Configurations

In ESC Datamodel, Interface refers to the VNIC attached to the VM. We can add one or more Interface under a VM Group. The interface section will have details to configure the VNIC.

This section describes basic interface configurations for Elastic Services Controller (ESC).

### Configuring Basic Interface Settings

This section describes basic interface configurations, such as:

- Network
- Subnet
- IP address
- MAC address
- VIM interface name, and so on for Elastic Services Controller (ESC).

### Configuring an Interface Name

To configure VIM interface name, specify attribute `<vim_interface_name>` for an interface in the Deployment XML file. Use `<vim_interface_name>` to use a specific name when generating an interface name. If these attribute is not specified, ESC will auto-generate an interface name, which is a combination of the

deployment_name, group_name, and a random UUID string. For example: my-deployment-na_my-gro_0_8053d7gf-hyt33-4676-h9d4-9j4a5599472t.

**Note** This feature is currently supported only on OpenStack.

If the VM group is elastic and a `vim_interface_name` has been specified, a numeric index is added after the interface name for the second interface name onwards (the first one remains unchanged). For example, if the specified interface name is set as `<vim_interface_name>interface_1</vim_interface_name>` and scaling is set to 3, three VMs are created with three different interface name, interface_1, interface_1_1, and interface_1_2. If a VM group only has a single VM, then there is no "_<index>" appended to the custom interface name. A single deployment can contain multiple VM groups, and each individual VM group can specify a different `vim_interface_name` value, if required. For example, a deployment could have two VM groups: the first group specifies a `vim_interface_name` and all VMs have their names generated as described above. The second VM group does not specify a `vim_interface_name`, therefore all VM names created from this group are auto generated. The same interface name can be used in separate interface sections within the same VM group, or in separate VM groups within a deployment, or in different deployments if required.

If attributes `<vim_interface_name>` or `<port>` are used for the same interface, the `vim_interface_name` value will be ignored and the value in the `port` attribute will be used.

```
<esc_datamodel xmlns="https://www.cisco.com/esc/esc"> <tenants><tenant>
<name>Admin</name>
<deployments>
<deployment>
<deployment_name>NwDepModel_nosvc</deployment_name>
<interface>
<nicid>0</nicid>
<vim_interface_name>interface_1</vim_interface_name>
<network>my-network</network>
</interface>
```

**Note** You can use a maximum of 61 characters for an interface name should not contain special characters and can only contain alphanumeric characters and "_" and "-". The following are some output samples with the custom port name. If the `vim_interface_name` was set during the deployment, the same value will be shown in the output. If this value was not set during the deployment, ESC will auto-generate the port name.

- Below is an example of the output operational data fetched using the esc_nc_cli script after adding a custom interface name. A new element called `vim_interface_name` will be shown under the interface element.

```
[admin@esc-3-1-xxx]$ esc_nc_cli --user <username> --password <password> get
esc_datamodel/opdata
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
. . .
    <interface>
        <nicid>0</nicid>
        <type>virtual</type>
        <port_id>e4111069-5d00-493b-8ea9-1a2ca134b5c8</port_id>
        <vim_interface_name>interface_1</vim_interface_name>      <!-- NEW IN OUTPUT
-->
        <network>c7fafeca-aa53-4349-9b60-1f4b92605420</network>
```

```
        <subnet>255.255.255.0</subnet>
        <ip_address>192.168.2.1</ip_address>
        <mac_address>fa:16:3e:d7:5e:da</mac_address>
        <netmask>255.255.240.0</netmask>
        <gateway>192.168.2.255</gateway>
    </interface>
```

- Below is an example output operational data fetched using a REST API.

```
GET http://localhost:8080/ESCManager/v0/deployments/example-deployment-123
| xmllint --format -
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<deployments>
. . .
    <interface>
        <network_uuid>c7fafeca-aa53-4349-9b60-1f4b92605420</network_uuid>
        <gateway>172.16.0.1</gateway>
        <ip_address>172.16.12.251</ip_address>
        <mac_address>fa:16:3e:30:0c:99</mac_address>
        <netmask>255.255.240.0</netmask>
        <nic_id>0</nic_id>
        <port_forwarding/>
        <port_uuid>1773cdbf-fe5f-4af1-adff-3a9c1dd1c47d</port_uuid>
       <vim_interface_name>interface_1</vim_interface_name>         <!-- NEW IN OUTPUT
 -->
        <security_groups/>
        <subnet_uuid>7b2ce63b-eb20-4ff8-8d49-e46ee8dde0f5</subnet_uuid>
        <type>virtual</type>
    </interface>
```

In all the above scenarios, if `vim_interface_name` is not specified in the deployment.xml, the output will still contain this element, however with an internally generated interface name. For example:

```
<vim_interface_name>vm-name-deployme_Grp1_1_0f24cd7e-cae7-402e-819a-5c84087103ba</vim_interface_name>
```

## Assigning the MAC Address

ESC deployment on VMware vCenter supports assigning MAC address using the MAC address range, or MAC address list from the MAC address pool to deploy VMs to the network.

You can assign MAC address in the following ways:

Using the Interface

```
<interfaces>
    <interface>
 <nicid>1</nicid>
 <network>MANAGEMENT_NETWORK</network>
 <ip_address>172.16.0.11</ip_address>
 <mac_address>fa:16:3e:73:19:a0</mac_address>
    </interface>
</interfaces>
```

During scaling, you can assign the MAC address list or MAC address range from the MAC address pool.

```
<scaling>
 <min_active>2</min_active>
 <max_active>2</max_active>
 <elastic>true</elastic>
 <static_ip_address_pool>
  <network>MANAGEMENT_NETWORK</network>
  <ip_address>172.16.0.11</ip_address>
  <ip_address>172.16.0.12</ip_address>
```

```
  <ip_address>172.16.0.13</ip_address>
 </static_ip_address_pool>
 <static_mac_address_pool>
  <network>MANAGEMENT_NETWORK</network>
  <mac_address>fa:16:3e:73:19:a0</mac_address>
  <mac_address>fa:16:3e:73:19:a1</mac_address>
  <mac_address>fa:16:3e:73:19:a2</mac_address>
 </static_mac_address_pool>
</scaling>
```

Assign MAC address using MAC address range.

```
<scaling>
 <min_active>2</min_active>
 <max_active>2</max_active>
 <elastic>true</elastic>
 <static_ip_address_pool>
  <network>MANAGEMENT_NETWORK</network>
  <ip_address_range>
   <start>172.16.0.25</start>
   <end>172.16.0.27</end>
  </ip_address_range>
 </static_ip_address_pool>
 <static_mac_address_pool>
  <network>MANAGEMENT_NETWORK</network>
  <mac_address_range>
   <start>fa:16:3e:73:19:b0</start>
   <end>fa:16:3e:73:19:b2</end>
  </mac_address_range>
 </static_mac_address_pool>
</scaling>
```

**Note** You cannot change the MAC or IP pool in an existing deployment, or during scaling (when min and max value are greater than 1) of VM instances in a service update.

In VMware vCenter, while assigning the MAC address, the server might override the specified value for "Generated" or "Assigned" if it does not fall in the right ranges or is determined to be a duplicate. Because of this, if ESC is unable to assign the MAC address the deployment fails.

## Configuring Subnet for an Interface

Subnets can be passed through the datamodel. Subnet within interfaces can be specified in the Interface section of the Deployment XML file. If there is no subnet specified in the datamodel, ESC will let OpenStack select the subnet for interface creation and will use the subnet from the port created by OpenStack.

```
<interface>
   <nicid>0</nicid>
   <network>my-network</network>
  <subnet>my-subnet</subnet>
    </interface>
```

The `no_gateway` attribute allows ESC to create a subnet with the gateway disabled. In the example below, the `no_gateway` attribute is set to true to create a subnet without gateway.

```
<networks>
<network>
<name>mgmt-net</name>
<subnet>
```

```
<name>mgmt-net-subnet</name>
<ipversion>ipv4</ipversion>
<dhcp>false</dhcp>
<address>172.16.0.0</address>
<no_gateway>true</no_gateway><!-- DISABLE GATEWAY -->
<gateway>172.16.0.1</gateway>
<netmask>255.255.255.0</netmask>
</subnet>
</network>
</networks>
```

## Configuring an Out-of-Band Port

ESC also allows you to attach an out-of-band port to a VNF. To do this, pass the UUID or the name of the port in the deployment request file while initiating a service request. For more information, see, Out-of-band Volumes section in the Cisco Elastic Services Controller User Guide.

**Note** While undeploying or restoring a VNF, the ports attached to that VNF will only be detached and not deleted. ESC does not allow scaling while using out-of-band port for a VM group. You can configure only one instance of VM for the VM group.Updating the scaling value for a VM group, while using the out-of-band port is not allowd during a deployment update.

```
<esc_datamodel xmlns="https://www.cisco.com/esc/esc">
    <name>tenant</name>
    <deployments>
        <deployment>
            <name>depz</name>
            <vm_group>
                <name>g1</name>
                <image>Automation-Cirros-Image</image>
                <flavor>Automation-Cirros-Flavor</flavor>
                <bootup_time>100</bootup_time>
                <reboot_time>30</reboot_time>
                <recovery_wait_time>10</recovery_wait_time>
                <interfaces>
                    <interface>
                        <nicid>0</nicid>
                        <port>057a1c22-722e-44da-845b-a193e02807f7</port>
                        <network>my-network</network>
                    </interface>
                </interfaces>
            </vm_group>
        </deployment>
    </deployments>
</esc_datamodel>
```

## Dual Stack Support

A dual stack network allows you to assign multiple IP addresses. These multiple IP addresses can be assigned on different subnets to a given interface within a VNF deployment using ESC.

ESC supports the following for dual stack:

- Configuring the network and list of subnet
- Configuring the network and list of subnet and ip address
- Configuring the network and list of ip address ( no subnet)

- Specifying the network and list of subnet/ip ( same subnet but different ip)

**Note** Currently, ESC supports dual stack only on OpenStack. ESC supports end-to-end IPv6 for OpenStack deployments.

A new container element named addresses is added to the Interface. This container holds a list of address elements. An address element must have an address_id (key). The subnet and fixed-ip address fields are optional, but you must specify either one.

The container address is as follows:

```
container addresses {
  list address {
    key "address_id";
    leaf address_id {
      description "Id for the address in address list.";
      type uint16;
      mandatory true;
    }
    leaf subnet {
      description "Subnet name or uuid for allocating IP to this port";
      type types:escnetname;
    }
leaf ip_address {
description "Static IP address for this specific subnet";
type types:escipaddr;
must "../../../../../scaling/max_active = 1 or
count(../../../../../scaling/static_ip_address_pool) > 0"
{
error-message "Static ip address pools must be configured when static ip addresses are
configured.";
}
}
```

Dual stack now supports KPI monitoring. A new child element address_id has been added to the metric_collector element. This accepts a value which points to an address within the specified nicid to be used for KPI monitoring. That is, it allows one of the addresses defined beneath an interface to be used for KPI monitoring.

```
…
    <interface>
        <nicid>1</nicid>
        <network>demo-net</network>
        <addresses>
            <address>
                <address_id>0</address_id>
                <subnet>demo-subnet</subnet>
            </address>
        </addresses>
    </interface>

           <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_occurrences_true>5</metric_occurrences_true>
               <metric_occurrences_false>5</metric_occurrences_false>
```

```
            <metric_collector>
                <type>ICMPPing</type>
                <nicid>1</nicid>
                <address_id>0</address_id>
                <poll_frequency>10</poll_frequency>
                <polling_unit>seconds</polling_unit>
               <continuous_alarm>false</continuous_alarm>
              </metric_collector>
            </kpi>
          </kpi_data>
...
```

**Note** The address_id under the metric_collector element must be the same as one of the address_id beneath the interface.

Dual stack interfaces can now be used in day-0 variable substitution. This means the ability to substitute the values from the multiple addresses defined under a single interface. Day 0 configuration is defined in the datamodel under the config_data tag.

In case of dual stack with multiple IP addresses, the variables are in the form NICID_<n>_<a>_<PROPERTY> where:

- <n> is the nicid for the interface.
- <a> is the address_id of an address within that interface.

The list of possible day-0 substitution variables from dual stack is:

| | | |
|---|---|---|
| NICID_n_a_IP_ALLOCATION_TYPE | string containing FIXED \| DHCP | ipv4 or ipv6 |
| NICID_n_a_IP_ADDRESS | IP address | ipv4 or ipv6 |
| NICID_n_a_GATEWAY | Gateway address | ipv4 or ipv6 |
| NICID_n_a_CIDR_ADDRESS | CIDR prefix address | ipv4 or ipv6 |
| NICID_n_a_CIDR_PREFIX | Integer with CIDR prefix-length | ipv4 or ipv6 |
| NICID_n_a_NETMASK | If an ipv4 CIDR address and prefix are present, ESC will automatically calculate and populate the netmask variable. This is not substituted in the case of an IPv6 address and should not be used. | ipv4 only |

For information on day-0 confirguration for single IP address, see Day Zero Configuration chapter in the Cisco Elastic Services Controller User Guide.

The template file defined in the config_data with day-0 configurations is as follows:

```
NICID_0_NETWORK_ID=${NICID_0_NETWORK_ID}
NICID_0_MAC_ADDRESS=${NICID_0_MAC_ADDRESS}

NICID_0_0_IP_ALLOCATION_TYPE=${NICID_0_0_IP_ALLOCATION_TYPE}
NICID_0_0_IP_ADDRESS=${NICID_0_0_IP_ADDRESS}
NICID_0_0_GATEWAY=${NICID_0_0_GATEWAY}
NICID_0_0_CIDR_ADDRESS=${NICID_0_0_CIDR_ADDRESS}
```

```
NICID_0_0_CIDR_PREFIX=${NICID_0_0_CIDR_PREFIX}
NICID_0_0_NETMASK=${NICID_0_0_NETMASK}

NICID_0_1_IP_ALLOCATION_TYPE=${NICID_0_1_IP_ALLOCATION_TYPE}
NICID_0_1_IP_ADDRESS=${NICID_0_1_IP_ADDRESS}
NICID_0_1_GATEWAY=${NICID_0_1_GATEWAY}
NICID_0_1_CIDR_ADDRESS=${NICID_0_1_CIDR_ADDRESS}
NICID_0_1_CIDR_PREFIX=${NICID_0_1_CIDR_PREFIX}
```

The datamodel is as follows:

```
<?xml version="1.0" encoding="ASCII"?>
<esc_datamodel xmlns="https://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>dep-tenant</name>
      <deployments>
        <deployment>
          <name>cirros-dep</name>
          <vm_group>
            <name>Grp1</name>
            <bootup_time>600</bootup_time>
            <recovery_wait_time>30</recovery_wait_time>
            <flavor>Automation-Cirros-Flavor</flavor>
            <image>Automation-Cirros-Image</image>
           <interfaces>
              <interface>
                <!-- No dual stack support on mgmt interface in ESC 4.1 -->
                <nicid>0</nicid>
                <network>my-network</network>
              </interface>
              <interface>
                <nicid>1</nicid>
                <network>ent-network1</network>
                <addresses>
                  <address>
                    <!-- IPv4 Dynamic -->
                    <address_id>0</address_id>
                    <subnet>v4-subnet_A</subnet>
                  </address>
                  <address>
                    <!-- IPv6 Dynamic -->
                    <address_id>1</address_id>
                    <subnet>v6-subnet_B</subnet>
                  </address>
                </addresses>
              </interface>
              <interface>
                <nicid>2</nicid>
                <network>ent-network2</network>
                <addresses>
                  <address>
                    <!-- IPv4 Static -->
                    <address_id>0</address_id>
                    <subnet>v4-subnet_C</subnet>
                    <ip_address>172.16.87.8</ip_address>
                  </address>
                  <address>
                    <!-- IPv6 Static -->
                    <address_id>1</address_id>
                    <subnet>v6-subnet_D</subnet>
                    <ip_address>fd07::110</ip_address>
                  </address>
                </addresses>
```

```
        </interface>
        <interface>
          <nicid>3</nicid>
          <network>ent-network3</network>
          <addresses>
            <address>
              <!--  Only ip config -  ipv6 but no subnet  -->
              <address_id>0</address_id>
              <ip_address>fd07::110</ip_address>
            </address>
            <address>
              <!--  Only ip config - ipv4 but no subnet  -->
              <address_id>1</address_id>
              <ip_address>172.16.88.9</ip_address>
            </address>
          </addresses>
        </interface>
        <interface>
          <nicid>4</nicid>
          <network>ent-network4</network>
          <addresses>
            <address>
              <!-- ipv4 same subnet as address_id 6 -->
              <address_id>0</address_id>      //
              <subnet>v4-subnet_F</subnet>
              <ip_address>172.16.86.10</ip_address>
            </address>
            <address>
              <!-- ipv4 same subnet as id 5 -->
              <address_id>1</address_id>
              <subnet>v4-subnet_F</subnet>
              <ip_address>172.16.86.11</ip_address>
            </address>
          </addresses>
        </interface>
      </interfaces>
      <kpi_data>
...
```

After successful deployment using multiple IPs, ESC provides a list of addresses as notification, or opdata.

A list of multiple <address> elements under the parent <interface> element containing the following:

- **address_id**—the address id specified in the input XML
- **subnet element**—subnet name or uuid
- **ip_address element**—the port's assigned IP on that subnet
- **prefix**—the subnet CIDR prefix
- **gateway**—the subnet gateway address
- ESC Static IP support

Notification:

```
        <vm_id>1834124d-b70b-41b9-9e53-fb55d7c901f0</vm_id>
        <name>jenkins-gr_g1_0_e8bc9a81-4b9a-437a-807a-f1a9bbc2ea3e</name>
        <generated_name>custom_vim_name
      <host_id>dc380f1721255e2a7ea15932c1a7abc681816642f75276c166b4fe50</host_id>

        <hostname>my-server</hostname>
```

```
<interfaces>
  <interface>
    <nicid>0</nicid>
    <type>virtual</type>
    <vim_interface_name>custom_vim_name
    <port_id>4d57d4a5-3150-455a-ad39-c32fffbb10b1</port_id>
    <mac_address>fa:16:3e:d2:50:a5</mac_address>
    <network>45638651-2e92-45fb-96ce-9efdd9ea343e</network>
    <address>
      <address_id>0<address_id>
      <subnet>6ac36430-4f58-454b-9dc1-82f7a796e2ff</subnet>
      <ip_address>172.16.0.22</ip_address>
      <prefix>24</prefix>
      <gateway>172.16.0.1</gateway>
    </address>
    <address>
      <address_id>1<address_id>
      <subnet>8dd9f501-19d4-4782-8335-9aa9fbd4dab9</subnet>
      <ip_address>2002:dc7::4</ip_address>
      <prefix>48</prefix>
      <gateway>2002:dc7::1</gateway>
    </address>
    <address>
      <address_id>2<address_id>
      <subnet>a234501-19d4-4782-8335-9aa9fbd4caf6</subnet>
      <ip_address>172.16.87.8</ip_address>
      <prefix>20</prefix>
      <gateway>172.16.87.1</gateway>
    </address>
  </interface>
```

Sample opdata:

```
<interfaces>
 <interface>
    <nicid>0</nicid>
    <type>virtual</type>
    <vim_interface_name>custom_vim_name
    <port_id>4d57d4a5-3150-455a-ad39-c32fffbb10b1</port_id>
    <mac_address>fa:16:3e:d2:50:a5</mac_address>
    <network>45638651-2e92-45fb-96ce-9efdd9ea343e</network>
    <address>
       <address_id>0</address_id>
       <subnet>6ac36430-4f58-454b-9dc1-82f7a796e2ff</subnet>
       <ip_address>172.16.0.22</ip_address>
       <prefix>24</prefix>
       <gateway>172.16.0.1</gateway>
    </address>
    <address>
       <address_id>1</address_id>
       <subnet>8dd9f501-19d4-4782-8335-9aa9fbd4dab9</subnet>
       <ip_address>2002:dc7::4</ip_address>
       <prefix>48</prefix>
       <gateway>2002:dc7::1</gateway>
    </address>
 </interface>
</interfaces>
```

You can also see that the day-0 substitution values are replaced in the output data. Sample output data with the values populated in the day-0 configuration is as follows:

```
NICID_0_NETWORK_ID=45638651-2e92-45fb-96ce-9efdd9ea343e
NICID_0_MAC_ADDRESS=fa:16:3e:d2:50:a5
```

```
NICID_0_0_IP_ALLOCATION_TYPE=DHCP
NICID_0_0_IP_ADDRESS=172.16.0.22
NICID_0_0_GATEWAY=172.16.0.1
NICID_0_0_CIDR_ADDRESS=172.16.0.0
NICID_0_0_CIDR_PREFIX=24
NICID_0_0_NETMASK=255.255.255.0

NICID_0_1_IP_ALLOCATION_TYPE=DHCP
NICID_0_1_IP_ADDRESS=2002:dc7::4
NICID_0_1_GATEWAY=2002:dc7::1
NICID_0_1_CIDR_ADDRESS=2002:dc7::/48
NICID_0_1_CIDR_PREFIX=48
```

### Dual Stack with Static IP Support

ESC supports dual stack with static IP support. As part of the initial configuration the user can provide the subnet and IP to be configured.

**Note** ESC supports static IP only when the scaling is false or minimum /maximum =1.

When you create a VM with out-of-band network, and specify a list of subnets with static IP (the network has multiple subnets), then ESC applies both subnet and the corresponding static IP.

In the example below, two subnets (ipv4 and ipv6 ) are added to a single interface.

```
<?xml version="1.0" encoding="ASCII"?>
<esc_datamodel xmlns="https://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>dep-tenant</name>
      <deployments>
        <deployment>
          <name>cirros-dep</name>
          <vm_group>
            <name>Grp1</name>
            <bootup_time>600</bootup_time>
            <recovery_wait_time>30</recovery_wait_time>
            <flavor>Automation-Cirros-Flavor</flavor>
            <image>Automation-Cirros-Image</image>
            <interfaces>
              <interface>
                <nicid>0</nicid>
                <network>ent-network2</network>
                <addresses>
                  <address>
                    <!-- IPv4 Static -->
                    <address_id>0</address_id>
                    <subnet>v4-subnet_C</subnet>
                    <ip_address>172.16.87.8</ip_address>
                  </address>
                  <address>
                    <!-- IPv6 Static -->
                    <address_id>1</address_id>
                    <subnet>v6-subnet_D</subnet>
                    <ip_address>fd07::110</ip_address>
                  </address>
                </addresses>
              </interface>
```

```
    </interfaces>
  <kpi_data>
```

For information on deploying VNFs, see Deploying Virtual Network Functions on OpenStack.

# Advanced Interface Configurations

This section describes several interface configurations for Elastic Services Controller (ESC) and the procedure to configure the hardware interfaces.

For information on basic interface settings, see Basic Interface Configurations.

## Configuring Advance Interface Settings

### Configuring SR-IOV in ESC

Single Root I/O Virtualization (SR-IOV) allows multiple VMs running a variety of guest operating systems to share a single PCIe network adapter within a host server. It also allows a VM to move data directly to and from the network adapter, bypassing the hypervisor for increased network throughput and lower server CPU burden.

#### Configuring SR-IOV in ESC for OpenStack

Before you configure SR-IOV in ESC for OpenStack, configure the hardware and OpenStack with the correct parameters.

To enable SR-IOV in ESC for OpenStack, specify the interface `type` as `direct`. The following snippet shows a sample datamodel:

```
<interfaces>
 <interface>
  <nicid>0</nicid>
  <network>my-network</network>
  <type>direct</type>
  </interface>
</interfaces>
...
```

#### Configuring SR-IOV in ESC for VMware

Before you configure SR-IOV in ESC for VMware, consider the following:

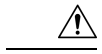
- Enable SR-IOV Physical Functions on desired ESXi hosts. For more information, see VMware documentation.
- Consider the following important points before enabling SR-IOV:
  - Review the list of physical network adaptors that VMware supports for SR-IOV. See VMware documentation.
  - Review the list of VM features that are not supported on a VM with SR-IOV configured. See VMware documentation.
  - In a cluster deployment (defined by "zone" in the datamodel) with SR-IOV, make sure that each ESXi host has identical Physical Functions enabled for SR-IOV selection. For example, if a VM is going to use vmnic7 as the Physical Function, make sure that each host has vmnic7 and SR-IOV status for each vmnic7 is enabled.

To enable SR-IOV in ESC for VMware, specify interface`<type>` as `direct` and also extension `<name>` as `sriov_pf_selection` in the deployment datamodel. Interface Type `direct`indicates an SR-IOV device and extension name `sriov_pf_selection` indicates the physical function. The following snippet shows a sample datamodel:

```
<vm_group>
...
<interface>
  <nicid>2</nicid>
  <network>MgtNetwork</network>
  <type>direct</type>
</interface>
<interface>
  <nicid>3</nicid>
  <network>MgtNetwork</network>
  <type>direct</type>
</interface>
...
<extensions>
 <extension>
 <name>sriov_pf_selection</name>
 <properties>
 <property>
 <name>nicid-2</name>
 <value>vmnic1,vmnic2</value>
 </property>
 <property>
 <name>nicid-3</name>
 <value>vmnic3,vmnic4</value>
 </property>
 </properties>
 </extension>
 </extensions>
</vm_group>
```

### Limitations for SR-IOV Interfaces

- When you boot the VNFs, the SR-IOV interfaces might show up in reverse order when compared to the order presented in ESXi. It causes interface configuration errors with a lack of network connectivity for a particular VNF. This is a known problem with VMware 6.5.
- Deploying a service with multiple SR-IOV interfaces on ESXi 7.0 is supported from ESC 5.7 Version onwards.



**Caution** Verify the interface mapping before you begin configuring the SR-IOV network interfaces on the VNF. This ensures that the network interface configuration applies to the correct physical MAC address interface on the VM host.

After the VNF boots, you can confirm which MAC address maps to which interface. Use the `show interface` command to see detailed interface information, including the MAC address for an interface. Compare the MAC address to the results of the `show kernel ifconfig` command to confirm the correct interface assignment.

# Configuring Allowed Address Pair

Cisco Elastic Services Controller allows you to specify the address pairs in the deployment datamodel to pass through a specified port regardless of the subnet associated with the network.

The address pair is configured in the following ways:

- List of Network—When a list of network is provided on a particular interface, ESC will get the subnet details from the OpenStack for these networks and add them to the corresponding port or interface. The following example explains how to configure address pairs as a list of network:

```
<interface>
             <nicid>1</nicid>
             <network>network1</network>
             <allowed_address_pairs>
               <network>
                   <name>bb8c5cfb-921c-46ea-a95d-59feda61cac1</name>
               </network>
               <network>
                   <name>6ae017d0-50c3-4225-be10-30e4e5c5e8e3</name>
               </network>
             </allowed_address_pairs>
           </interface>
         </interfaces>
```

- List of Address— When a list of address is provided, ESC will add these addresses to the corresponding interface. The following example explains how to configure address pairs as a list of address:

```
<interface>
               <nicid>0</nicid>
               <network>esc-net</network>
               <allowed_address_pairs>
                <address>
                 <ip_address>10.10.10.10</ip_address>
                 <netmask>255.255.255.0</netmask>
                </address>
                <address>
                 <ip_address>10.10.20.10</ip_address>
                 <netmask>255.255.255.0</netmask>
                </address>
               </allowed_address_pairs>
             </interface>
```

# Configuring Security Group Rules

Cisco Elastic Services Controller (ESC) allows you to associate security group rules to the deployed instances on OpenStack. These security group rules are configured by specifying the necessary parameters in the deployment datamodel. In addition to configuring security group rules, if any VNF instance fails, ESC recovers the instance and applies the security group rules for the redeployed VNF.

To configure security group rules, do the following:

### Before you begin

- Make sure you have created a tenant through ESC.
- Make sure you have security groups created.
- Make sure you have the security group name or UUID.

**Step 1** Log in to the ESC VM as a root user.

**Step 2** Run the following command to check the UUIDs of a given security group:

```
nova --os-tenant-name <NameOfTheTenant> secgroup-list
```

**Step 3** Pass the following arguments in the deployment data model:

```
<interfaces>
 <interface>
  <nicid>0</nicid>
  <network>my-network</network><!-- depends on network name -->
  <security_groups>
  <security_group>0c703474-2692-4e84-94b9-c29e439848b8</security_group>
  <security_group>bbcdbc62-a0de-4475-b258-740bfd33861b</security_group>
  </security_groups>
</interface>
<interface>
 <nicid>1</nicid>
 <network>sample_VmGrpNet</network><!--depends on network name  -->
 <security_groups>
 <security_group>sample_test_SQL</security_group>
 </security_groups>
</interface>
```

**Step 4** Run the following command to verify whether the security groups are associated with the VM instance:

```
nova --os-tenant-name <NameOfTenant> show <NameOfVMinstance>
```

# Configuring SNAT Router and Floating IP

This section describes how to create, update, and delete a router.

**Overview**

Source Network Address Translation (SNAT) router is an OSP feature. SNAT router allows traffic from a private network to go out to the public network. Virtual machines launched on a private network can get into the internet through a gateway to perform SNAT. The gateway replaces the source IP of the originating packet with its public side IP.

## Creating Router

Create a router as out-of-band with various specifications like admin state, an external gateway with SNAT enable property,internal interface, static routes, distribution and so on. By default, the SNAT property is enabled.

Create a router in two ways:

1. Create a simple out-of-band router with one create request.
2. Create an out-of-band router with interface added into it - First creates a request for router creation and successive update request for attaching interface to the router and adding static routes to router.

### Creating router using esc_nc_cli script:

To create a router using esc_nc_cli script, an XML payload is passed with the router name and required configuration properties:

```
<?xml version='1.0' encoding='ASCII'?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
    <routers>
        <router>
           <name>testRouter</name>
           <admin_state>true</admin_state>
           <external_network>internet-net</external_network>
           <snat_enable>true</snat_enable>
           <distribution>false</distribution>
           <description>check for desc</description>
           <interfaces>
               <interface>
                  <subnet>automation_subnet</subnet>
                  <port_id>18b6e6df-fc48-49dc-842e-a1cee546173e</port_id>
               </interface>
    </interfaces>
    <static_routes>
                <route>
                   <route_name>RouteA</route_name>
                   <destination>172.26.0.0/24</destination>
                   <next_hop>10.85.103.93</next_hop>
                 </route>
            </static_routes>
        </router>
     </routers>
</esc_datamodel>
```

If the router is successfully created, you will receive an XML payload with a single `<ok/>` element. If the router does not get created and the action is unsuccessful, you will get an error message with a validation error or OpenStack API error.

### Creating router using ESC REST API:

To create a router, an HTTP POST operation is specified to the ESCManager API:

```
POST: /ESCManager/v0/<tenant-id>/routers/<internal-router-id>
```

If the router gets successfully created, you will receive an HTTP 200 code. If the router does not get created and the action is unsuccessful, you will receive a validation error or OpenStack API error with an appropriate HTTP error code and error message.

For example:

```
{
        "name": "rout0020",
        "admin_state": true,
        "route": [{
                        "route_name": "RouteA",
                        "destination": "172.26.0.0/24",
                        "next_hop": "10.85.103.93"
                }
        ],
        "interface":[{
                        "subnet":"automation_subnet"
                }
        ],
        "external_network": "internet-net"
}
```

```
[admin@localhost]$ curl --user admin:P@55w0rd! -k -X POST -d @rest.json -H 'Content-Type:
application/json' -H 'callback: https://localhost:9009' -H 'Callback-ESC- Events:
https://localhost:9009'
https://localhost:8443/ESCManager/v0/SystemAdminTenantId/routers/testRouterId
```

**Note** For adding internal interface,

1. You can specify <subnet> (or) <subnet> & <port_id>.
2. You can add *n* number of valid interfaces.

For example:

```
<interface>
   <subnet>testsubnet</subnet>
   <port_id>portuuidid</port_id>
</interface>
```

To add static routes, specify the following:

1. `route_name` - to maintain uniqueness among the routes.(should be unique and this name is maintained at ESC level)
2. Destination - based on need
3. next_hop - based on need

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

1. Router names should be less than or equal to 255 characters in length.
2. Usage of Non-Existing external network, subnet, a port in the payload throws an error.
3. Adding static routes is allowed only when an internal/external interface is added for corresponding `next_hop`.
4. XML file - if specified, must contain a valid XML document (esc_nc_cli only).

# Updating the Router

Once the router is created, update the router with some valid configurations.

During the router update the following operations are supported:

- Add/Clear External gateway
- Attach/Detach Interface
- Add/Delete Static routes
- Admin state UP/DOWN.

### Updating Router Using esc_nc_cli Script

You can add an external gateway, interface, static routes to update the router configurations. To perform these actions, add valid tags to the payload. Configure only one external gateway to the router. Whereas you can add any number of interfaces and static routes to the router.

To clear or remove external gateway, interface, static routes, add operation = *delete* tag with the start tags. For Example:

```
<external_network operation="delete">internet-net</external_network>
```

You can detach/remove all interfaces/routes at once by giving the operation delete to parent of `<interface> & <route>`. For example:

```
<interfaces operation="delete">
    <interface>
        <subnet>automation_subnet</subnet>
        <port_id>18b6e6df-fc48-49dc-842e-a1cee546173e</port_id>
   </interface>
</interfaces>
```

For updating admin state as UP or DOWN, change the boolean value of `<admin_state>`.

If updating the router is successful, you will receive an XML payload with a single `<ok/>` element. However, if the action is unsuccessful, you will receive a validation error or OpenStack API error with an appropriate error message.

**Updating Router Using ESC REST API:**

You can add an external gateway, interface, static routes to update the router, to perform these actions you need to add a valid JSON value to the payload. Configure only one external gateway to the router. Whereas you can add any number of interfaces and static routes to the router.

To clear or remove the external gateway, interface, static routes then remove the respective JSON data from the payload.

For updating Admin state UP or DOWN, Change the boolean value of `admin_state`. To update a router, HTTP PUT operation can be specified to the ESCManager API:

```
PUT: /ESCManager/v0/<tenant-id>/routers/<internal-router-id>
```

The payload must contain the existing router name and update properties as mentioned previously.

If successful, you will receive an HTTP 200 code. However, if the action remains unsuccessful, you will receive a validation error or OpenStack API error with an appropriate HTTP error code and error message.

Following is an example of API invocation to update a router :

```
[admin@localhost]$ curl --user admin:P@55w0rd! -k -X PUT -d @rest.json -H 'Content-Type:
application/json' -H 'callback: https://localhost:9009' -H 'Callback-ESC-Events:
https://localhost:9009'
https://localhost:8443/ESCManager/v0/SystemAdminTenantId/routers/testRouterId
```

# Deleting a router

Delete a router using both esc_nc_cli & REST API interfaces. Only the routers managed by the current ESC VM can be deleted. You can only delete one router at a time.

**Deleting router using esc_nc_cli script:**

To delete router in esc_nc_cli, pass the router name with `esc_nc_cli` command. Use the following command to delete the router:

```
esc_nc_cli delete-router <router-name>
```

If the router gets deleted and the action is successful, you will receive an XML payload with a single `<ok/>` element. However, if the action is unsuccessful, you will receive a validation error or OpenStack API error with an appropriate error message.

**Deleting router using ESC REST API:**

To delete a router, use an HTTP DELETE operation in the ESCManager API.

For example:

```
DELETE: /ESCManager/v0/<tenant-id>/routers/<internal-router-id>
```

The payload should contain the existing router name and update properties.

If the delete router action is successful, you will receive an HTTP 200 code. However, if the action remains unsuccessful, you will receive a validation error or OpenStack API error with an appropriate HTTP error code, and error message.

Following is an example of API invocation to delete a router :

```
[admin@localhost]$  curl --user admin:P@55w0rd! -k -X DELETE -H 'callback:
https://localhost:9009' -H 'Callback-ESC-Events: https://localhost:9009'
https://localhost:8443/ESCManager/v0/SystemAdminTenantId/routers/testRouterId
```

**Note** You cannot delete Router until all of the static routes are deleted.

**Notifications:**

You will receive both NETCONF notifications and ESC REST callback messages during the router operations.

***Table 1:***

| Notification (NETCONF or ESC callback) | When the notification is sent |
|---|---|
| CREATE_ROUTER | When the OpenStack has finished the router creates a request operation, and it was either a success, or an error occurred. |
| UPDATE_ROUTER | When the OpenStack has finished the router update request operation, and it was either a success, or an error occurred. |
| ATTACH_INTERFACE | When the OpenStack has attached an interface to the router, and it was either a success, or an error occurred. |
| DETACH_INTERFACE | When the OpenStack has detached an interface from the router and it was either a success, or an error occurred. |
| ADD_STATIC_ROUTE | When the OpenStack has added a static route to the router and it was either a success, or an error occurred. |
| DELETE_STATIC_ROUTE | When the OpenStack has deleted a static route from the router and it was either a success, or an error occurred. |

| Notification (NETCONF or ESC callback) | When the notification is sent |
|---|---|
| DELETE_ROUTER | When the OpenStack has finished the router delete request operation, and it was either a success, or an error occurred. |

The following example shows a CREATE_ROUTER successful NETCONF notification:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2021-09-25T10:31:26.76+00:00</eventTime>
    <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Router successfully created.</status_message>
    <router>testRouter-1</router>
    <tenant>admin</tenant>
    <event>
        <type>CREATE_ROUTER</type>
    </event>
    </escEvent>
</notification>
```

The following example shows an ATTACH_INTERFACE 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>2021-09-25T10:31:28.891+00:00</eventTime>
    <escEvent xmlns="http://www.cisco.com/esc/esc">
         <status>SUCCESS</status>
         <status_code>200</status_code>
         <status_message>Interface successfully attached.</status_message>
         <router>testRouter-1</router>
         <router_interface>mgmt-net-subnet</router_interface>
              <event>
                   <type>ATTACH_INTERFACE</type>
               </event>
    </escEvent>
</notification>
```

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

the `<status>` value is *FAILURE*,

the `<status_code>` is *500*, and

the `<status_message>` is an appropriate message, either internally generated or sent back from OpenStack.

# Associating Floating IP to VM

Assign floating IP as Boolean true or false. When the value is set to true, a free-floating IP from the OpenStack is associated with the interface. You can assign a particular floating IP as input for an interface. To disassociate a floating IP from the interface, specify the floating IP as false.

ESC performs the following actions:

- ESC collects the list of floating IPs available from the OpenStack and associates a free-floating IP to the VM during deployment.
- Disassociate the same floating IP while deleting the deployment.

- During recovery, the same floating IP has to be associated with the deployment.
- When the floating IP is specified as false, the floating IP is dissociated from the interface.

### Floating IP at VM Group Level

If the floating IP is mentioned at the VM group level, a free-floating IP from OpenStack is associated with the interface with nic ID 0. As the floating IP at VMGroup corresponds to nic ID 0, the floating IP value cannot be specified at both VM group and interface level at the same time. If specified, you will receive an error message. The value of floating IP is updated in the interface table under the column `floating_ip`.

Following is an example for assigning floating IP at the VM group level.

```
<vm_group>
 <name>cirros1</name>
 <bootup_time>60</bootup_time>
 <recovery_wait_time>0</recovery_wait_time>
 <image>Automation-Cirros-Image</image>
 <flavor>medium2</flavor>
 <floating_ip>true</floating_ip>
 ....
 ....
</vm_group>
```

### Floating Ip at Interface Level:

When the floating IP is mentioned at the interface level, the floating IP is associated with the corresponding interface. The value of floating IP is updated in the interface table under the column `floating_ip`.

Following is an example for assigning floating IP at the interface level.

```
<interface>
<nicid>1</nicid>
<floating_ip>10.85.103.99</floating_ip>
<network>esc-net</network>
</interface>
```

### Floating Ip at Dual interface :

When the port has dual interfaces, you can specify the floating IP to a particular interface. The value of floating IP is updated in the interface_addresses table under the column `floating_ip`.

Following is an example for assigning floating IP at the dual interface level.

```
</interface>
<interface>
<nicid>1</nicid>
<network>udhanasenet</network>
<addresses>
<address>
<address_id>0</address_id>
<floating_ip>true</floating_ip>
<subnet>udh-sub</subnet>
</address>
<address>
<address_id>1</address_id>
<floating_ip>10.85.103.95</floating_ip
<subnet>udh-sub</subnet>
</address>
</addresses>
</interface>
```

### Disassociation of floating IP from an interface:

Assign the floating IP values as *false* to dissociate the floating IP from the interface.

```
<floating_ip>false</floating_ip>
```

**Error Scenarios :**

You will receive an error message while performing the following tasks:

1. If you assign floating IP both at VM group level and interface level of nic id 0.
2. If you try to assign floating IP to an IPV6 address in OpenStack.
3. If you try to assign floating IP when there is no free-floating IP available at OpenStack under a particular tenant.
4. During scaling, only dynamically allocated floating IP is supported. You cannot specify a fixed floating IP address) if you try to give specific floating IP, you will receive an error message.

# Hardware Acceleration Support

You can configure hardware acceleration features on VIM using the *flavor data model*. The following hardware acceleration features can be configured:

- **vCPU Pinning**—enables binding and unbinding of a process to a vCPU (Virtual Central Processing Unit) or a range of CPUs, so that the process executes only on the designated CPU or CPUs rather than any CPU.
- **VMware vSphere performance optimization for large pages and non-uniform memory access (NUMA)**—enables improvement of system performance for large pages and NUMA i.e., system's ability to accept higher load and modify the system to handle a higher load.
- **VMware vCenter support for PCIe Passthrough interface**—enables assigning a PCI device to an instance on OpenStack.

The following example explains how to configure hardware acceleration features using *flavor data model*:

```
$ cat example.xml
<?xml version='1.0' encoding='ASCII'?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <flavors>
    <flavor>
      <name>testfl6</name>
      <vcpus>1</vcpus>
      <memory_mb>2048</memory_mb>
      <root_disk_mb>10240</root_disk_mb>
      <ephemeral_disk_mb>0</ephemeral_disk_mb>
      <swap_disk_mb>0</swap_disk_mb>
      <properties>
        <property>
          <name>pci_passthrough:alias</name>
          <value>nic1g:1</value>
        </property>
      </properties>
    </flavor>
  </flavors>
</esc_datamodel>
$ /opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli --user <username> --password <password>
edit-config ./example.xml
```

# Creating Additional Parameters for VMware vSphere NUMA Attributes

ESC enhances NUMA for VMware vSphere by adding additional configuration parameters.

This enhancement adds the additional or advanced configuration for VMware vSphere as a prefix to pass configuration parameters instead of passing these values through the day-0 configuration files.

Prefix: `extConfigParam`

Example:

```
<configuration>
    <dst>extConfigParam:mgmt-ipv4-addr</dst>
    <data>$NICID_1_IP_ADDRESS/16</data>
</configuration>
```

The additional configuration helps to minimize the data model changes, and restrict configuration changes to the VIM layer.

# Configuring PCI or PCIe Device Passthrough on VMware vCenter

ESC supports VMware vCenter PCI or PCIe device passthrough (VMDirectPath I/O). This enables VM access to physical PCI functions on platforms with an I/O memory management unit.

**Before You Begin**

For the PCI / PCIe devices of a host VM to enable passthrough, the vSphere administrator must mark these devices in the vCenter.

**Note** You must reboot the host after PCI settings. Put the host to maintenance mode, power off or migrate all VMs to other hosts.

To specify PCI device passthrough request in ESC deployments, include the <type> attribute with value set to ***passthru***. To specify the PCI device to be selected for a particular vm_group or network, include the *pci_id*. The data model is as follows:

```
<tenants>
  <tenant>
   <name>admin</name>
   <deployments>
    <deployment>
     <name>test</name>

     <vm_group>
      <name>test-g1</name>
      <image>uLinux</image>
      <bootup_time>300</bootup_time>
      <recovery_wait_time>10</recovery_wait_time>
      <interfaces>
       <interface>
        <nicid>1</nicid>
        <network>MgtNetwork</network>
        <ip_address>192.168.0.102</ip_address>
    </interface>
      <interface>
      <nicid>2</nicid>
      <network>VM Network</network>
```

```
  <type>passthru</type>
  <ip_address>172.16.0.0</ip_address>
  </interface>
   <interface>
    <nicid>3</nicid>
    <network>VM Network</network>
    <type>passthru</type>
    <ip_address>192.168.46.117</ip_address>
</interface>
  <interface>
             <nicid>3</nicid>
        <type>passthru</type>
        <network>MgtNetwork</network>
        <pci_id>0000:07:10.3</pci_id>
        </interface>
   </interfaces>
```

After successful deployment, the ***passthru*** value is set in the interface section of the notification as well as in the operational data.

## Auto Selecting PCI or PCIe PassThrough Device

ESC needs one or more PCI or PCIe passthrough devices to be attached to each deployment without a particular PCI ID. ESC first selects a host. ESC selects the next available PCI or PCIe passthrough enabled device and attaches it during the deployment. If there is no PCI or PCIe passthrough enabled device available, ESC fails the deployment. The vSphere administrator has to ensure all computing-host within the target computing-cluster have enough number of PCI or PCIe passthrough enabled devices.

**Note**

- PCI or PCIe passthrough is not considered by ESC placement algorithm. For example, ESC does not select a host because it has available resources to complete the PCI or PCIe passthrough requests.
- ESC selects the PCI or PCIe passthrough device randomly. ESC does not consider the type or specification of the device. It selects the next available PCI or PCIe device from the list.
- Recovery fails if the VNF is recovered to a computing-host that ESC has selected based on the ESC placement algorithm, and if that computing-host does not have any PCI or PCIe passthrough enabled devices available.
- DRS must be turned off for the passthrough to work.