Cisco Elastic Services Controller 5.8 Administration Guide

First Published: 2022-07-22

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

PREFACE

CHAPTER 1

CHAPTER 2

About This Guide

This guide helps you to perform ESC administration related tasks such as basic configurations, monitoring the health of ESC, and viewing system logs.

Audience

This guide is designed for network administrators responsible for provisioning, configuring, and monitoring VNFs. Cisco Elastic Services Controller (ESC) and the VNFs whose lifecycle it manages are deployed in a Virtual Infrastructure Manager (VIM). Currently OpenStack, VMware vCenter, VMware vCloud Director, CSP 2100 / 5000, and Amazon Web Services (AWS) are the supported VIMs. The administrator must be familiar with the VIM layer, vCenter, OpenStack and AWS resources, and the commands used.

Cisco ESC is targeted for Service Providers (SPs) and Large Enterprises. ESC helps SPs reduce cost of operating the networks by providing effective and optimal resource usage. For Large Enterprises, ESC automates provisioning, configuring and monitoring of network functions.

Terms and Definitions

The below table defines the terms used in this guide.

Table 1: Terms and Definitions

Terms	Definitions
AWS	Amazon Web Services (AWS) is a secure cloud services platform, offering compute, database storage, content delivery and other functionalities.
ESC	Elastic Services Controller (ESC) is a Virtual Network Function Manager (VNFM), performing lifecycle management of Virtual Network Functions.
ETSI	European Telecommunications Standards Institute (ETSI) is an independent standardization organization that has been instrumental in developing standards for information and communications technologies (ICT) within Europe.

Terms	Definitions
ETSI Deployment Flavour	A deployment flavour definition contains information about affinity relationships, scaling, min/max VDU instances, and other policies and constraints to be applied to the VNF instance. The deployment flavour defined in the VNF Descriptor (VNFD) must be selected by passing the *flavour_id* attribute in the InstantiateVNFRequest payload during the instantiate VNF LCM operation.
HA	ESC High Availability (HA) is a solution for preventing single points of ESC failure and achieving minimum ESC downtime.
KPI	Key Performance Indicator (KPI) measures performance management. KPIs specify what, how and when parameters are measured. KPI incorporates information about source, definitions, measures, calculations for specific parameters.
MSX	Cisco Managed Services Accelerator (MSX) is a service creation and delivery platform that enables fast deployment of cloud-based networking services for both Enterprises and Service Providers customers.
NFV	Network Function Virtualization (NFV) is the principle of separating network functions from the hardware they run on by using virtual hardware abstraction.
NFVO	NFV Orchestrator (NFVO) is a functional block that manages the Network Service (NS) lifecycle and coordinates the management of NS lifecycle, VNF lifecycle (supported by the VNFM) and NFVI resources (supported by the VIM) to ensure an optimized allocation of the necessary resources and connectivity.
NSO	Cisco Network Services Orchestrator (NSO) is an orchestrator for service activation which supports pure physical networks, hybrid networks (physical and virtual) and NFV use cases.
OpenStack Compute Flavor	Flavors define the compute, memory, and storage capacity of nova computing instances. A flavor is an available hardware configuration for a server. It defines the *size* of a virtual server that can be launched.
Service	A service consists of a single or multiple VNFs.
VDU	The Virtualisation Deployment Unit (VDU) is a construct that can be used in an information model, supporting the description of the deployment and operational behaviour of a subset of a VNF, or the entire VNF if it was not componentized in subsets.
VIM	The Virtualized Infrastructure Manager (VIM) adds a management layer for the data center hardware. Its northbound APIs are consumed by other layers to manage the physical and virtual resources for instantiation, termination, scale in and out procedures, and fault & performance alarms.
VM	A Virtual Machine (VM) is an operating system OS or an application installed on a software, which imitates a dedicated hardware. The end user has the same experience on a virtual machine as they would have on dedicated hardware.
VNF	A Virtual Network Function (VNF) consists of a single or a group of VMs with different software and processes that can be deployed on a Network Function Virtualization (NFV) Infrastructure.

Terms	Definitions
VNFC	A Virtual Network Function Component is (VNFC) a composite part of the VNF, synonymous with a VDU, which could be implemented as a VM or a container.
VNFM	Virtual Network Function Manager (VNFM) manages the life cycle of a VNF.

Related Documentation

The Cisco ESC doc set comprises of the following guides to help you perform installation, configuration; the lifecycle management operations, healing, scaling, monitoring and maintenance of the VNFs using different APIs.

Guide	Information Provided in This Guide
Cisco Elastic Services Controller Release Notes	Includes new features and bugs, known issues.
Cisco Elastic Services Controller Install and Upgrade Guide	Includes procedure for new installation and upgrade scenarios, pre and post installation tasks, and procedure for ESC High Availability (HA) deployment.
Cisco Elastic Services Controller User Guide	Includes lifecycle management operations, monitoring, healing and scaling of the VNFs.
Cisco Elastic Services Controller ETSI NFV MANO User Guide	Includes lifecycle management operations, monitoring, healing and scaling of the VNFs using the ETSI APIs.
Cisco Elastic Services Controller Administration Guide	Includes maintenance, monitoring the health of ESC, and information on system logs generated by ESC.
Cisco Elastic Services Controller NETCONF API Guide	Information on the Cisco Elastic Services Controller NETCONF northbound API, and how to use them.
Cisco Elastic Services Controller REST API Guide	Information on the Cisco Elastic Services Controller RESTful northbound API, and how to use them.
Cisco Elastic Services Controller ETSI REST API Guide	Includes information on the Cisco Elastic Services Controller ETSI APIs, and how to use them.
Cisco Elastic Services Controller Deployment Attributes	Includes information about deployment attributes used in a deployment datamodel.
Cisco Elastic Services Controller Open Source	Includes information on licenses and notices for open source software used in Cisco Elastic Services Controller.

Obtaining Documentation Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see *What's New in Cisco Product Documentation*, at: http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html.

Subscribe to *What's New in Cisco Product Documentation*, which lists all new and revised Cisco technical documentation, as an RSS feed and deliver content directly to your desktop using a reader application. The RSS feeds are a free service.

CHAPTER 1

Elastic Services Controller Overview

Elastic Services Controller Overview

Cisco Elastic Services Controller (ESC) is a Virtual Network Functions Manager (VNFM) managing the lifecycle of Virtual Network Functions (VNFs). ESC provides agentless and multi vendor VNF management by provisioning the virtual services. ESC monitors the health of VNFs , promotes agility, flexibility, and programmability in Network Function Virtualization (NFV) environments. It provides the flexibility to define rules for monitoring and associate actions that are triggered based on the outcome of these rules. Based on the monitoring results, ESC performs scale in or scale out operations on the VNFs. In the event of a VM failure ESC also supports automatic VM recovery.

ESC fully integrates with Cisco and other third party applications. As a standalone product, the ESC can be deployed as a VNF Manager. ESC integrates with Cisco Network Services Orchestrator (NSO) to provide VNF management along with orchestration. As a Specialized Virtual Network Function Manager (SVNFM), ESC tightly integrates with the Cisco Mobility VNFs. ESC can also be utilized as a Generic Virtual Network Function Manager (GVNFM) to provide lifecycle management for both Cisco and third-party VNFs.

ESC as a VNF Manager targets the virtual managed services and all service provider NFV deployments such as virtual packet core, virtual load balancers, virtual security services and so on. Complex services include multiple VMs that are orchestrated as a single service with dependencies between them.

CHAPTER 2

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 2:

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>
```

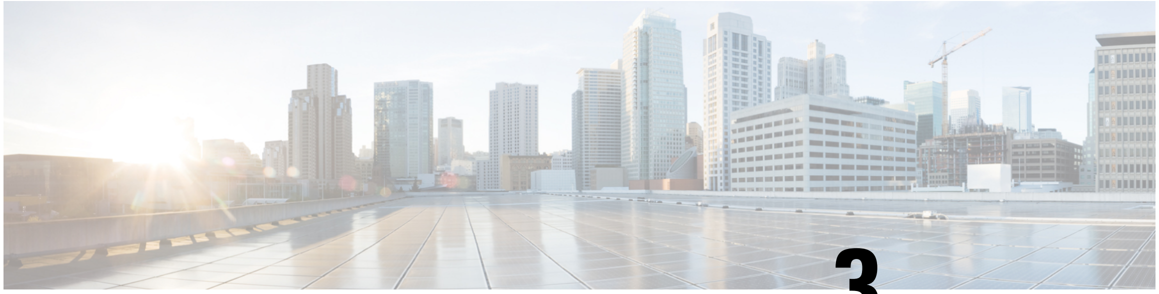
After successful deployment, the ***passthru*** value is set in the interface section of the notification as well as in the operational data.

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.



CHAPTER 3

Monitoring ESC Health

You can monitor the health of ESC and its services, using one of the following:

Monitoring the Health of ESC Using REST API

ESC provides REST API for any third party software to monitor the health of ESC and its services. Using the API, the third party software can query the health condition of ESC periodically to check whether ESC is in service. In response to the query, API provides status code and messages, see Table 3: ESC Health API Status Code and Messages in Standalone and Active-Standby High Availability, on page 29 for details. In an HA setup the virtual IP (VIP) must be used as the monitoring IP. The return value provides the overall condition of the ESC HA pairs. See the Table 5: Health API Status Messages for Standalone ESC and HA, on page 31 for details.

The REST API to monitor the health of ESC is as follows:

```
GET to https://<esc_vm_ip>:8060/esc/health
```

Note

- The monitoring health API is secured using the existing REST basic HTTP authentication. The user can retrieve the report by using the ESC REST API credentials.
- The ESC Health API port number is changed from 60000 to 8060.

The monitoring health API response with error conditions is as follows:

Example of the JSON response:

```
<?xml version="1.0" encoding="UTF-8" ?>
<esc_health_report>
<status_code>{error status code}</status_code>
<message>{error message}</message>
</esc_health_report>
```

The monitoring health API response for local Active/Active is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<esc_health_report>
    <status_code>2010</status_code>
    <message>ESC service is being provided. ESC AA cluster one or more node(s) not
healthy</message>
    <nodes>
        <node>
            <name>aa-esc-1.novalocal</name>
            <status>HEALTHY</status>
            <datacenter>dc1</datacenter>
            <services>
                <service>
                    <name>escmanager</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>elector</name>
                    <status>leader</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>drbd</name>
                    <status>active</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>pgsql</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                ...
            </services>
        </node>
        <node>
            <name>aa-esc-2.novalocal</name>
            <status>HEALTHY</status>
            <datacenter>dc1</datacenter>
            <services>
                <service>
                    <name>escmanager</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>elector</name>
                    <status>follower</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>drbd</name>
                    <status>standby</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>pgsql</name>
                    <status>stopped</status>
                    <is_expected>True</is_expected>
                </service>

                ...
            </services>
        </node>
        <node>
```

```
            <name>aa-esc-3.novalocal</name>
            <status>NOT_HEALTHY</status>
            <datacenter>dc1</datacenter>
            <services>
                <service>
                    <name>escmanager</name>
                    <status>stopped</status>
                    <is_expected>False</is_expected>
                </service>
                <service>
                    <name>elector</name>
                    <status>follower</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>vimmanager</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                ...
            </services>
        </node>
    </nodes>
</esc_health_report>
```

XML and JSON responses are also supported for the monitoring health API.

If the API response is successful, an additional field called *stage* is introduced.

```
<?xml version="1.0" encoding="UTF-8" ?>
<esc_health_report>
<status_code>{success status code}</status_code>
<stage>{Either INIT or READY}</stage>
<message>{success message}</message>
</esc_health_report>
```

The stage field has INIT or READY parameters.

INIT: The INIT parameter is the initial stage, where ESC accepts *pre-provisioning* requests such as configuring the config parameters or registering a vim connector.

READY: ESC is ready for any kind of *provisioning* requests such as deploying, undeploying and so on with this parameter.

The status code and messages below provide the health condition of ESC. The status codes with 2000 series imply that the ESC is operational. The status codes with 5000 series imply that at least one ESC component is not in service.

Table 3: ESC Health API Status Code and Messages in Standalone and Active-Standby High Availability

Status Code	Message
2000	ESC services are running.
2010	ESC services are being provided. ESC AA cluster one or more node(s) not healthy.
2040	ESC services running. VIM is configured, ESC initializing connection to VIM.
5010	ESC service, ESC_MANAGER is not running.

Status Code	Message
5020	ESC service, CONFD is not running.
5030	ESC service, MONA is not running.
5040	ESC service, VIM_MANAGER is not running.
5060	ESC service, ETSI is not running.
5070	Vim Connector IDs [vimId_1,vimId_2,...,vimId_N] are down. or 6 of 25 VIM Connectors are down. **Note** If more than 5 VIM connector IDs are down, then a summary message is printed instead of a list of VIM IDs.
5080	The NFVO service is not available.
5090	More than one ESC service (for example, confd and mona) are not running.
5091	One or more ESC services is not running and the NFVO service is not available.
5092	VIM Connector ID [vim-1] is down. The NFVO service is NOT available.

Table 4: ESC Health API Status Code and Messages in Active-Active High Availability

Status Code	Message
2000	ESC services are running (Active-Active setup).
2010	ESC services are provided. In ESC Active/Active cluster one or more node(s) are not healthy.
5000	ESC services not being provided, ESC AA cluster not healthy

Note ESC HA mode refers to ESC HA in DRBD setup only. For more information on the ESC HA setup, see the Cisco Elastic Services Controller Install Guide.

The table below describes the status message for standalone ESC and HA with success and failure scenarios. For more information on ESC standalone and HA setup, see the Cisco Elastic Services Controller Install Guide.

Table 5: Health API Status Messages for Standalone ESC and HA

	Success	Partial Success	Failure
Standalone ESC	The response is collected from the monitoring health API and the status code is 2000.	NA	• Monitor cannot get the response from the monitoring health API. • The response is collected from the monitoring health API and the status code returned is in the 5000 series.
ESC in HA (Active-Standby)	The response is collected from the monitoring health API and the status code is 2000.	The response is collected from the monitoring health API and the status code is 2010. This indicates that the ESC standby node cannot connect to ESC active node in ESC HA. However, this does not impact the ESC service to northbound.	• The monitor cannot get the response from the monitoring health API for more than two minutes. **Note** ESC monitoring health API may not be available for a certain period during the HA switchover period. The monitoring software must set a proper threshold to report service failure in this scenario. • The response is collected from the monitoring health API and the status code returned is in the 5000 series.

	Success	**Partial Success**	**Failure**
ESC in HA (Active-Active)	The response is collected from the monitoring health API and the status code is 2000.	The response is collected from the monitoring health API and the status code is 2010. This indicates that the ESC services are being provided but one or more nodes are not healthy in the ESC AA cluster. This does not impact the ESC service to northbound.	• For Local Active-Active, if the monitor cannot get the response from the monitoring health API for more than one minute. For Geo Active-Active, if the monitor cannot get the response from the monitoring health API for more than seven minutes (this depends on the configuration in heat template) **Note** ESC monitoring health API may not be available for a certain period during the local and geo switchover period. The monitoring software must set a proper threshold to report service failure in this scenario. • The geo switchover period depends upon the configuration in the heat template. By default, the switchover starts five minutes after the primary datacenter failure. The response is collected from the monitoring health API and the status code returned is 5000. **Note** During switchover, the status code returned will temporarily be 5000 until the new leader becomes healthy.

ESC Health Monitor Enhancements

The ESC Health Monitor API is enhanced to:

- Determine the status of the ESC components.
- Provide a single point of contact for the SNMP agent to simplify the connectivity and authentication details.

The ESC monitor component hosts the Health Monitor API, which can be used to provide a listing of the downed ESC components. The Health Monitor uses both public and internal health URLs for each ESC component to determine its individual status. For example, the VNFM status is determined by the health monitor by executing the URL:

```
https://localhost:8252/etsi/health
```

The URL determines the status of the ESC components, and returns a relevant status code and status message as part of the SNMP trap notifications.

ESC Health Monitor API for VIM Connector Status

The ESC Health Monitor API is extended to query the VIM connector details using the new ESC Health Monitor API (URL):

```
http://<escmanager-host>:8088/escmanager/vims
```

The URL is executed against the active node in the ESC standalone and HA setup, and against every node in the ESC Active/Active setup.

The health monitor payload returns additional information to determine the binary status of all the configured VIM connectors. The status of the VIM connectors is either *healthy* or *down*.

To determine if a single VIM connector is healthy, the ESC Health Monitor API performs a query on the VIM to which a VIM connector is defined. If the result is has a **CONNECTION_SUCCESSFUL** internal status, then the VIM connector is healthy.

If the query fails, then the VIM connector is down.

Furthermore, the returned status message contains a comma separated list of the specific VIM IDs which are down. The example shows the payload the ESC Health Monitor returns for two VIM connectors that are down:

```
{
"message": "VIM Connector IDs [vim-connector-site-1A, vim-connector-site-1C] are down.",
"status_code": "5070"
}
```

For details on the SNMP trap notifications for the VIM connectors, see Monitoring the Health of ESC Using SNMP Trap Notifications, on page 34.

The ESC Health Monitor does not monitor the VIM connector status by default. To enable the ESC Health Monitor, see Enabling SNMP Traps for VIM and NFVO Monitoring in SNMP Trap Notifications, on page 46.

ESC Health Monitor API for the NFVO Connectivity Status

The ESC Health Monitor API can determine the connectivity to the NFVO. ESC provides an API to query the connectivity of the NFVO to ESC. The NFVO responds to the standard SOL003 defined API query. The URL is as follows:

```
https://<vnfm-host>:8252/etsi/nfvo/health
```

If the NFVO authenticates successfully and responds to the SOL003 defined API, then the NFVO is reachable and healthy.

The example shows the payload the ESC Health Monitor returns when the NFVO is configured but not reachable:

```
{
"message": "The NFVO service is NOT available.",
"status_code": "5080"
}
```

The ESC Health Monitor does not monitor the NFVO connection status by default. To enable the ESC Health Monitor, see Enabling SNMP Traps for VIM and NFVO Monitoring in SNMP Trap Notifications, on page 46.

For information on the ETSI deployment, see the *Cisco Elastic Services Controller ETSI NFV MANO User Guide*

Monitoring the Health of ESC Using SNMP Trap Notifications

You can also configure notifications on the health of various ESC components via SNMP traps using an SNMP Agent. This Agent is installed as part of the standard ESC installation and supports the SNMP version 2c and 3 protocols. The SNMP traps currently support only the state of the ESC product and not of the VNFs managed by ESC. This section describes the steps required to configure the ESC SNMP agent and also cover the events that will be triggered as part of the notifications.

Before you begin

- Ensure the **CISCO-ESC-MIB** and **CISCO-SMI MIB** files are available on your system. These are located in the /opt/cisco/esc/snmp/mibs directory. Download these to your SNMP Manager machine and place them in the $HOME/.snmp/mibs directory.
- Configure SNMP Agent. There are three methods to configure SNMP agent. These methods are discussed in detail in the section below.

Configuring SNMP Agent

In order to receive the SNMP traps, configure the SNMP Agent parameters. The agent can be configured using three different methods described in this section. The best or most applicable method to use depends on your use case.

1. **Enabling and configuring SNMP Agent during ESC installation**:

   - **Standalone or Active/Standby HA setup via BootVM**

     While installing ESC, use the following additional parameters to configure SNMP agent:

     ```
     % bootvm.py <esc_vm_name> --image <image-name> --net <net-name> --enable-snmp-agent
      --ignore-ssl-errors
     --managers "udp:ipv4/port,udp:[ipv6]/port"
     ```

Note The value for managers is a comma separated list of locations where SNMP traps are delivered in the format "udp:ipv4/port" or "udp:[ipv6]/port". The IP and port must be replaced with the actual values.

   - **Active/Active HA setup**

     You can enable the SNMP agent during the Active/Active installation. You can pass the config parameters `ignore_ssl_errors` and list of `managers` to configure the agent on install. It can be defined in the aa-params.yaml or passed on the following command line.

     ```
     openstack stack create name-aa --template aa.yaml -e aa-params.yaml \
     --parameter nameprefix=ESC_AA \
     ```

```
--parameter image_name=ESC-5_2_0_43 \
--parameter flavor_name=m1.large \
...
--parameter snmp_agent_startup: auto \
--parameter snmp_agent_ignore_ssl_errors: true \
--parameter snmp_agent_managers: [ "udp:ipv4/port,udp:[ipv6]/port" ]
```

2. **Enabling and Configuring via ESCADM**

   - **Standalone or Active/Standby HA setup**

     Using the escadm tool, you can modify the SNMP agent configuration parameters such as managers and ignoreSslErrors properties.

     ```
     sudo escadm snmp set --ignore_ssl_errors=true
     --managers="udp:ipv4/port,udp:[ipv6]/port"
     ```

   - **Active/Active HA setup**

     Run the following command on all the Leader ready nodes which is the ESC node 1, node 2, node 4, and node 5:

     ```
     sudo escadm snmp set --startup=auto
     ```

Note If a node is deleted and recreated by a stack update, you must rerun the previous command.

Restart ESC services on the SNMP enabled nodes only on the primary datacenter which is node 1 and 2. One node at a time.

```
sudo escadm stop
sudo escadm restart
```

Once the leader node is healthy, and SNMP agent is running, you can add the SNMP agent configurations on the leader node as follows:

```
sudo escadm snmp set --ignore_ssl_errors=true
--managers="udp:ipv4/port,udp:[ipv6]/port"
```

Note The `ignore-ssl-errors` parameter is mainly for a developer environment to prevent SSL errors, where self signed certificates are used on the ESC VM.

The value for managers is a comma separated list of locations where SNMP traps are delivered `"udp:ipv4/port" or "udp:[ipv6]/port"` format. The IP and port must be replaced with the actual values.

3. **Updating the configuration file**

   The SNMP agent must already be enabled for this configuration update to take effect.

   The configuration is in the file /opt/cisco/esc/esc_database/snmp.conf. This file is in JSON format. Following is an example:

   ```
   {
   "publicCommunities": "public",
   "users": [],
   ```

```
   "sysDescr": "admin@localhost",
   "ignoreSslErrors": "yes",
   "logLevel": "INFO",
   "sysName": "system name",
   "managers": [{
      "privPassword": "enc:95w3hE+uZ1A3vyykaPpKEw==",
      "targetEndpoint": "udp:localhost/12000",
      "privProtocol": "AES128",
      "targetCommunity": "public",
      "label": "some manager",
      "targetProtocol": "v2",
      "authProtocol": "SHA",
      "authPassword": "enc:IYt1UIW8wug3vyykaPpKEw==",
      "authentication": "authpriv",
      "username": "admin",
      "engineId": "80:00:00:00:01:02:03:04"
   }]
   }
```

The configuration is in the file */opt/cisco/esc/esc_database/snmp.conf.* for a user defined community string. This file is in JSON format.

Note This configuration is applicable to SNMP version 2c protocol.

```
{
    "publicCommunities": "test",
    "users": [],
    "sysDescr": "TestSNMPAgentConfiguration SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "dnd-admin-1208",
    "managers": []
}
```

Use the following to configure the snmptrapd.conf config file:

```
AuthCommunity log,execute,net test
disableAuthorization yes
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
Array:\n%v \n -------------- \n
```

Output:

```
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c test -M +/opt/cisco/esc/snmp/mibs localhost:2001
 CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
[admin@dnd-admin-1208 ~]$ snmpwalk -v2c -c test -M +/opt/cisco/esc/snmp/mibs 172.24.0.33:2001
 CISCO-ESC-MIB::vnfm
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
CISCO-ESC-MIB::escStatusCode.0 = STRING: "2000"
```

The configuration is in the file */opt/cisco/esc/esc_database/snmp.conf.* with multiple communities delimited with commas.

```
{
    "publicCommunities": "public, foo ,bar",
    "users": [],
    "sysDescr": "TestSNMPAgentConfiguration SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "dnd-admin-1208",
    "managers": []
}
```

Use the following to configure the snmptrapd.conf config file:

```
AuthCommunity log,execute,net public, foo ,bar
disableAuthorization yes
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
Array:\n%v \n -------------- \n
```

Example:

```
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c foo -M +/opt/cisco/esc/snmp/mibs localhost:2001
CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c public -M +/opt/cisco/esc/snmp/mibs localhost:2001
 CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c bar -M +/opt/cisco/esc/snmp/mibs localhost:2001
CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
```

Defining ESC SNMP MIBs

The following table describes the content of ESC MIB. These values are configurable in the snmp.conf file.

Variable	Simple IOD	Description
sysName	SNMPv2-MIB::sysName.0	Specify the name of the ESC machine. The host name is taken by default.
sysDescr	SNMPv2-MIB::sysDescr.0	Specify the name of the SNMP Agent.
sysLocation	SNMPv2-MIB::sysLocation.0	Specify where the ESC machine is located.
sysContact	SNMPv2-MIB::sysContact.0	Specify the Admin contact.

The following table contains the trap entries of the SNMP MIB. The enterprise OID is 1.3.6.1.4.1.

Table 6: SNMP MIB Trap Entries

Node	Index	Parent
cisco	9	enterprises
ciscoMgmt	9	cisco
ciscoEscMIB	844	ciscoMgmt
escNotifs	0	ciscoEscMIB
escMIBObjects	1	ciscoEscMIB
vnfm	1	escMIBObjects

Node	Index	Parent
escStatusMessage	1	vnfm
escStatusCode	2	vnfm
escPreviousStatusCode	3	vnfm
escPreviousStatusMessage	4	vnfm

Enabling SNMP Trap Notifications

```
sudo escadm snmp start
```

Use the escadm tool to start the SNMP services.

You can also use esadm tool to stop, get the status, and modify the configurations of the SNMP agent.

```
sudo escadm snmp stop
sudo escadm snmp status
sudo escadm snmp restart
```

Managing SNMP Traps in ESC

This section covers:

- Understanding the SNMP Notification Types in ESC
- Managing SNMP Traps in ESC (SNMP Manager)
- SNMP GET/WALK Examples
- Managing Trap Endpoints (SNMP Managers)
- Managing ESC SNMP in an HA Environment
- Managing ESC SNMP Agent in an Active/Active Environment
- Managing Self-Signed Certificates in ESC

Understanding the SNMP Notification Types in ESC

The following table lists all the events supported by this version of the SNMP agent. These status codes and messages will be returned via a SNMP trap to a registered manager only when there is a change of state of ESC. The status codes with 2000 series imply that the ESC is operational. The status codes with 5000 series imply that at least one ESC component is not in service. For more details on status codes with 2000 series and 5000 series, see section, *Monitoring ESC Health Using REST API*.

Status Code	SNMP Agent-specific Message
5100	An HTTP error was received when using the ESC Monitor API

Status Code	SNMP Agent-specific Message
5101	The ESC Monitor replied, but the data could not be understood.
5102	The Agent could not create a network connection to the ESC Monitor API.
5199	An unhandled error occurred (details will be included in the message).
5210	"**AA LEADER node change**". In an AA environment where a node has become the LEADER, the agent on the node will send this notification. Only for local leader change.
5200	"**HA ACTIVE node change**". In an A/S HA environment where a node has become the ACTIVE node the agent sends this notification.
5220	"**Geo AA Primary datacenter change**" In a GEO A/A environment, after GEO switchover, when a node becomes the LEADER, the agent on the node will send this notification. Only for GEO leader change.

Managing SNMP Traps in ESC (SNMP Managers)

An SNMP manager is deployed in another system and is registered in the ESC SNMP agent. For example, an assurance system is a typical consumer of SNMP traps from ESC.

The examples below use basic UNIX SNMP tools such as *snmptrapd*, *snmpget* and *snmpwalk*.

SNMPv2c example

Configure the SNMP Trap daemon config file with the following:

```
authCommunity log,execute,net public
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
Array:\n%v \n -------------- \n
```

This lets the *snmptrapd* process notifications received using the "public" community string. Start the daemon in a terminal session, run the following command:

```
snmptrapd -f -C -c ./snmptrapd.conf -Le 12000
```

Open a second session to check if traps are being received:

```
snmptrap -v 2c -c public -n "" localhost:12000 0 linkUp.0
```

That should produce the following in session 1.

```
Agent Address: somehost.somedomain
Agent Hostname: localhost (UDP: [127.0.0.1]:51331->[0.0.0.0]:0)
Enterprise OID: .
Trap Sub-Type: 0
Community/Infosec Context: TRAP2, SNMP v2c, community public
Uptime: 0
PDU Attribute/Value Pair Array:
```

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::snmpTrapOID.0 = OID: IF-MIB::linkUp.0
--------------
```

Test the ESC SNMP agent, use the following manager entry in `snmp.config`. Traps produced by the SNMP agent will also be logged by the daemon. Make sure the Cisco and ESC MIB's are present in ~/.snmp/mibs.

SNMPv2 Managers Entry

```
"managers": [{
    "targetEndpoint": "udp:localhost/12000",
    "targetCommunity": "public",
    "label": "Trap test v2c",
    "targetProtocol": "v2c"
}]
```

SNMPv3 Example

Update the snmptrapd.conf file as follows:

```
disableAuthorization no
authCommunity log,execute,net public

createUser -e 0x8000000001020304 admin SHA authpassword AES privpassword
authUser log admin

format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
Array:\n%v \n -------------- \n
```

This adds the *admin* user. The "-e" specifies an engine ID: a hexadecimal string between 5 and 32 characters. Every SNMP v3 agent has an engine ID, which serves as a unique identifier for the agent. The engine ID is used with a hashing function to generate keys for authentication and encryption of the messages.

For systems to communicate, both sides must use the same authProtocol (MD5 or SHA) and privProtocol (AES or DES). Some devices do not support all of these combinations. You must check what can be used to ensure the trap receiver is configured in the same way. Start the daemon again in one terminal session:

```
snmptrapd -f -C -c ./snmptrapd.conf -Le 12000
```

Test the configuration in the second session, matching the username, passwords, engine ID and so on. Note that the *authPriv* security level selects both authentication and encryption.

```
snmptrap -v 3 -n "" -a SHA -A authpassword -x AES -X privpassword -l authPriv -u admin -e
0x8000000001020304 localhost:12000 0 linkUp.0
```

This should log a trap in window 1.

Example output:

```
Agent Address: casper.cisco.com
 Agent Hostname: localhost (UDP: [127.0.0.1]:53434->[0.0.0.0]:0)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v3, user admin, context
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::snmpTrapOID.0 = OID: IF-MIB::linkUp.0
```

To use the above configuration in ESC, use the following example. Note that the digits of the engine ID are separated by colons, not the "0x" format used by the trap daemon.

SNMPv3 Managers Entry

~~~
"managers": [{
   "privPassword": "privpassword",
   "targetEndpoint": "udp:localhost/12000",
   "privProtocol": "AES128",
   "targetCommunity": "public",
   "label": "V3 trap test",
   "targetProtocol": "v3",
   "authProtocol": "SHA",
   "authPassword": "authpassword",
   "authentication": "authpriv",
   "username": "admin",
   "engineId": "80:00:00:00:01:02:03:04"
}],
```
~~~

Example ESC Output for a v3 Message

```
Agent Address: casper.cisco.com
 Agent Hostname: localhost (UDP: [127.0.0.1]:52103->[0.0.0.0]:0)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v3, user admin, context 80:00:00:00:01:02:03:04
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (27252277) 3 days, 3:42:02.77
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.844.0.1
SNMPv2-MIB::sysDescr.0 = STRING: SNMP Agent
SNMPv2-SMI::enterprises.9.9.844.1.1.2.0 = STRING: "2000"
SNMPv2-SMI::enterprises.9.9.844.1.1.1.0 = STRING: "ESC services are running."
 --------------
```

Trap output

Typically, the trap contains four entries: statusCode, statusMessage, previousStatusCode and previousStatusMessage.

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (3971) 0:00:39.71
SNMPv2-MIB::snmpTrapOID.0 = OID: CISCO-ESC-MIB::statusNotif
SNMPv2-MIB::sysDescr.0 = STRING: ESC SNMP Server
CISCO-ESC-MIB::escStatusCode.0 = STRING: "2000"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
CISCO-ESC-MIB::escPreviousStatusCode.0 = STRING: "5102"
CISCO-ESC-MIB::escPreviousStatusMessage.0 = STRING: "Warning: Could not connect to ESC
Monitor. See log for details."
```

The ESC SNMP agent sends SNMP traps with the previous status and status code messages. This allows the client to determine what the latest SNMP trap is in response to.

If there is no previous status code and message, then those strings are empty. For example, The SNMP agent returns the value of the previous status code and status message as a MIB string:

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "2000"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
CISCO-ESC-MIB::escPreviousStatusCode.0 = STRING: "5090"
CISCO-ESC-MIB::escPreviousStatusMessage.0 = STRING: "More than one ESC service (confd, etsi)
 not running."
```

This allows the SNMP client to know that all services are running, and that this SNMP trap is in response to the Confd and ETSI services, which were not running previously, and are coming back.

SNMP Manager Options

Table 7: SNMP Manager Options

Key	Protocol	Description
targetCommunity	v2c	Which community to send the trap to. Defaults to *public*.
label	v2c/v3	A name for this manager.
targetEndpoint	v2c/v3	Address and port of where the trap is sent. Example: *udp:localhost/12000*.
targetProtocol	v2c/v3	SNMP protocol to use for this manager. Either v2c or v3. Defaults to v2c.
authPassword	v3	Password for the user. Enter the plain password and the agent will detect and encrypt it.
authProtocol	v3	The authentication protocol to use. Either SHA or MD5.
authentication	v3	Type of authentication can be one of the following: AuthPriv, AuthNoPriv or NoAuthNoPriv.
engineId	v3	The engine ID to use for this trap in hexadecimal. The engine ID must match the ID used by the manager. For example, 80:00:00:00:01:02:03:04
privPassword	v3	The encryption (privacy) password. Enter the plain password. The agent detects and encrypts it.
privProtocol	v3	The encryption protocol can be one of the following: DES, AES, AES128, AES192 or AES256
username	v3	Name of the user (or security name) for authentication

SNMP GET/WALK Examples

This section provides an example of how SNMP *gets* can be performed using the SNMP tools, *snmpwalk* and *snmpget*.

Note The examples assume that the ESC MIBS have been added to the SNMP MIB path.

SNMP GET - command line examples

Table 8:

SNMP Example	Command	Example Output
SNMPv2c Example	`snmpget -v2c -c public localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`	**Example Output** `CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."`

SNMPv3 NoAuthNoPriv Example	The following user is added to the ESC SNMP agent configuration.
	V3 SNMP users entry
	`"users": [{`
	`    "username": "admin"`
	`}],`
	Command
	`snmpget -v3 -l authpriv -u admin localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`
		Example Output
		`CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."`

SNMPv3 AuthNoPriv Example	The following user is added to the ESC SNMP agent configuration.
	V3 SNMP users entry
	`"users": [{`
	`"username": "admin",`
	`"authProtocol": "SHA",`
	`"authPassword": "authpassword"`
	`}],`
	Command
	`snmpget -v3 -l authpriv -u admin -a "SHA" -A "authpassword" localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`
		Example Output
		`CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."`

SNMP Example	Command	Example Output

SNMPv3 AuthPriv example	The following user is added to the ESC SNMP agent configuration.
	`"users": [{`
	`"username": "admin",`
	`"authProtocol": "SHA",`
	`"authPassword":`
	`"authpassword",`
	`"privProtocol": "AES128",`
	`"privPassword": "privpassword"`
	`}],`
	Command
	`snmpget -v3 -l authpriv -u`
	`admin -a "SHA" -A`
	`"authpassword" -x "AES128" -X`
	`"privpassword" localhost:2001`
	`CISCO-ESC-MIB::escStatusMessage.0`
		Example Output
		`CISCO-ESC-MIB::escStatusMessage.0`
		` = STRING: "ESC services are`
		`running."`

Managing Trap Endpoints (SNMP Managers)

The SNMP agent monitors its configuration file for changes and reloads when a change is made. Add or remove manager endpoints to the configuration file and the new configuration will be used in future traps.

Managing ESC SNMP Agent in an HA Environment

Two or more ESC nodes can be deployed in a HA configuration and the SNMP agent does support this configuration. However, consider the following points in an HA deployment:

- Both active and standby nodes must be configured to enable SNMP
- Only one ESC node (the active node) can send SNMP traps
- The SNMP agent on the standby node automatically receives the active configuration when switchover occurs.
- If a standby node becomes the active node due to failover, it generates a trap.

Managing ESC SNMP Agent in an AA Environment

The SNMP agent service is also supported in local or GEO ESC Active/Active setup. Following are the considerations in an Active/Active deployment:

- SNMP agent runs and sends traps on the leader node only.
- Traps are sent in the following scenarios:
  - On ESC health API status code change. The SNMP agent polls the Health Monitor API for AA, if there is a change in the status code returned, it is sent as a trap to its subscribers.
  - After local switchover by the node which becomes the new Leader to signify local switchover.
  - After GEO switchover by the node which becomes Leader in new GEO Primary datacenter.
- Changes made to the configuration in leader node is carried forward by new leader after switchover.

Managing Multiple Managers with Dedicated Target Community

An SNMP manager is deployed in another system and is registered in the ESC SNMP agent.

The ESC SNMP agent supports an array of multiple manager configurations that receive SNMP traps and each configuration has a dedicated target community.

The following example displays multiple managers support with dedicated target community:

Open window 1

Open the SNMP Agent configuration file */opt/cisco/esc/esc_database/snmp.conf* and add the following information:

```
{
    "publicCommunities": "public",
    "users": [],
    "sysDescr": "TestSNMPAgentConfiguration SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "dnd-admin-1208",
    "managers": [
        {
            "targetEndpoint": "udp:localhost/12006",
            "targetCommunity": "test1",
            "label": "Trap test v2c",
            "targetProtocol": "v2c"
        },
        {
            "targetEndpoint": "udp:localhost/12004",
            "targetCommunity": "test2",
            "label": "Trap test v2c",
            "targetProtocol": "v2c"
        }
    ]
}
```

Open Window 2

Use the following to Configure the SNMP Trap daemon config file:

```
AuthCommunity log,execute,net test1

disableAuthorization yes

format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
Array:\n%v \n -------------- \n
```

Open Window 3

Use the following to Configure the SNMP Trap daemon config file:

```
AuthCommunity log,execute,net test2
disableAuthorization yes
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%b)\n Enterprise OID: %N \n Trap
Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value Pair
Array:\n%v \n -------------- \n
```

Use escadm tool, to stop the confd services in the ESC VM:

sudo escadm confd stop

The escadm allows the snmptrapd process to receive notifications using the "test1" community string. Start the daemon in window 2 to run the following command:

```
snmptrapd -f -C -c ./snmptrapdm.conf -Le 12006
```

Once you run the previous command, you will see the following in window 2:

```
Agent Address: 0.0.0.0
 Agent Hostname: dnd-admin-1208.novalocal (UDP: [127.0.0.1]:57472->[127.0.0.1]:12006)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v2c, community test1
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1043610566) 120 days, 18:55:05.66
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.844.0.1
SNMPv2-MIB::sysDescr.0 = STRING: TestSNMPAgentConfiguration SNMP Agent
SNMPv2-SMI::enterprises.9.9.844.1.1.2.0 = STRING: "5020"
SNMPv2-SMI::enterprises.9.9.844.1.1.1.0 = STRING: "ESC service ESC_CONFD not running."
SNMPv2-SMI::enterprises.9.9.844.1.1.3.0 = STRING: "2000"
SNMPv2-SMI::enterprises.9.9.844.1.1.4.0 = STRING: "ESC services are running."
```

The escadm allows the snmptrapd process to receive notifications using the "test2" community string. Start the daemon in window 3 to run the following command:

```
snmptrapd -f -C -c ./snmptrapdm.conf -Le 12004
```

Once you run the previous command, you will see the following in window 3:

```
Agent Address: 0.0.0.0
 Agent Hostname: dnd-a-1208.novalocal (UDP: [127.0.0.1]:45804->[127.0.0.1]:12004)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v2c, community test2
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1043610566) 120 days, 18:55:05.66
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.844.0.1
SNMPv2-MIB::sysDescr.0 = STRING: TestSNMPAgentConfiguration SNMP Agent
SNMPv2-SMI::enterprises.9.9.844.1.1.2.0 = STRING: "5020"
SNMPv2-SMI::enterprises.9.9.844.1.1.1.0 = STRING: "ESC service ESC_CONFD not running."
SNMPv2-SMI::enterprises.9.9.844.1.1.3.0 = STRING: "2000"
SNMPv2-SMI::enterprises.9.9.844.1.1.4.0 = STRING: "ESC services are running."
```

SNMP Trap Notifications

Enabling SNMP Traps for VIM and NFVO Monitoring

The SNMP Agent uses the ESC Health Monitor API to query the status of ESC components, VIM connectors and NFVO connectivity statuses. By default, the ESC health monitor does not monitor the VIM or NFVO connectivity. The SNMP Traps are not generated for the same.

To enable VIM and NFVO connectivity status change traps, ensure that the ESC Health Monitor configuration file, `/opt/cisco/esc/esc-config/esc-config.yaml` has the following parameters:

```
monitor:
(2)report:
(4)nfvo:
(6)enabled: true
(4)vim_connectors:
(6)enabled: true
(6)name_threshold: 5
```

If the above parameters are not specified in the configuration file, then the monitoring of both vim and nfvo connectivity components defaults to false. The vim_conectors and name_threshold refers to how many vim connector IDs are output in the status before a generic message. The message states the number of vim connectors which are down, but not detailing their names, such as: "6 of 25 VIM Connectors are down."

See "SNMP Trap Notifications for VIM connectors" for status messages.

SNMP Trap Notifications for NFVO Connectivity

SNMP Traps are sent when the NFVO details are configured within the ETSI VNFM service, NFVO monitoring is enabled within the ESC Health Monitor configuration, and the NFVO cannot be reached.

The ETSI VNFM service tests the NFVO connectivity by using a standard SOL003 API to which the NFVO responds.

If the NFVO cannot be reached, the following SNMP trap is generated:

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "5080"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "The NFVO service is NOT available."
```

Note

- If the NFVO is reachable, but the credentials are incorrect, then the status is *not available*.
- The status of the NFVO connection is reported only when the ESC Monitor Health API is executed. The NFVO availability is not monitored periodically.

SNMP Trap Notifications for VIM Connectors

SNMP Traps are sent when the VIM connectors are configured within ESC, vim monitoring is enabled within the ESC Health Monitor configuration, and any of the configured vim connectors are not reachable. An unreachable VIM connector is one which has an internal ESC status which is not equal to **CONNECTION_SUCCESSFUL**.

- If a single VIM connector is not available, then the following trap is generated:

  ```
  CISCO-ESC-MIB::escStatusCode.0 = STRING: "5070"
  CISCO-ESC-MIB::escStatusMessage.0 = STRING: "VIM Connector ID [vim-id1] is down."
  ```

- If a two or more VIM connector are not available, then the following trap is generated:

  ```
  CISCO-ESC-MIB::escStatusCode.0 = STRING: "5070"
  CISCO-ESC-MIB::escStatusMessage.0 = STRING: "VIM Connector IDs [vim-id1, vim-id2,
  vim-id3] are down."
  ```

  Note The default number of vim connectors is 5. This can be configured in the esc-config.yaml file. See "Enabling SNMP Traps for VIM and NFVO Monitoring".

- If the number of VIM connectors which are not available exceeds the name threshold, then the following trap is generated:

```
    CISCO-ESC-MIB::escStatusCode.0 = STRING: "5070"
    CISCO-ESC-MIB::escStatusMessage.0 = STRING: "6 of 25 VIM Connectors are down."
```

For information on the ESC health monitor API, see Monitoring the Health of ESC Using REST API, on page 27.

Combined and Split SNMP Trap Modes

The SNMP agent is configured to return *combined* or *split* traps.

- **Combined Traps**: Currently, the SNMP agent generates combined traps. It considers the output from the ESC Health Monitor and sends it as a single, complete trap, even if that output indicates multiple ESC components or events. The output is from the last SNMP agent polling period, which sends multiple downed ESC services as a single trap.

- **Split Traps**: ESC Release 5.4 and later supports a single trap per **UP** or **DOWN** event for each ESC service or component. Each **UP** or **DOWN** event has its own unique status message and status code.

> **Note** A monitored *ESC service* is the health status of any existing ESC component: MONA, confd, ETSI, ESCMANAGER and VIMMANAGER. The VIM connector validity and NFVO connectivity are part of the VIM manager component (monitored as part of the VIMMANAGER).
>
> The monitoring of both VIM connector validity and NFVO connectivity is disabled by default. When enabled, the ESC Health Monitor automatically reports the connectivity statuses respectively. The SNMP agent uses the results when sending out traps, along with the existing ESC services.

The output of individual traps per **UP** or **DOWN** event (split traps) removes status codes and traps which indicate an event has occurred to multiple ESC services, therefore the following ESC Health Monitor does not appear as SNMP trap codes when operating in *split* mode, effectively removing any trap which combines ESC component information.

Configuration

The combined or a split trap mode is controlled by a new property called the *trapMode*, which can be set in the `/opt/cisco/esc/esc_database/snmp.conf` file as shown below:

```
{
    "publicCommunities": "public",
    "users": [],
    "sysDescr": "TestSNMPAgentTraps SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "test-5-4-0-51-keep",
    "trapMode": "combined",
    "managers": []
}
```

The default value when this file is auto generated is *combined*, which is also the default value if the *trapMode* is not present in the configuration file - this maintains backward compatibility during an upgrade.

SNMP ESC Component Status Codes

The status codes for **UP** event traps (when MONA was down but is now back up) are new, as a trap has not been generated before to indicate a single ESC service being restored. A list of codes the SNMP agent sends out for all ESC services are listed below:

Table 9: SNMP ESC Component Status Codes

ESC Component	UP Code	DOWN Code	UP Code Message	DOWN Code Message
ALL SERVICES UP	2000		ESC services are running	
ESC_MANAGER	2010	5010	ESC service ESC_MANAGER running.	ESC service ESC_MANAGER not running.
ESC_CONFD	2020	5020	ESC service ESC_CONFD running.	ESC service ESC_CONFD not running.
MONA	2030	5030	ESC service MONA running.	ESC service MONA not running.
VIM_MANAGER	2040	5040	ESC service VIM_MANAGER running.	ESC service VIM_MANAGER not running.
ETSI	2060	5060	ESC service ETSI running.	ESC service ETSI not running.
Connectivity Service				
VIM CONNECTORS	2070	5070	Vim Connector ID [vimid_1] is up	Vim Connector ID [vimid_1] is down.
NFVO	2080	5080	The NFVO service is available.	The NFVO service is NOT available.

High Availability

When ESC is operating in a High Availability pair, the above status codes and messages still apply, but there is one additional status code which can apply:

Table 10:

ESC Component	Code	Message
ALL SERVICES UP - ESC HA NODE	2010	ESC services are running. ESC High-Availability node not reachable.

A 2010 SNMP trap is sent out with the details above when this situation occurs. There is no 5010 equivalent for High Availability. When the situation is resolved, the *2000 - ESC Services running* message is sent. The **UP** traps are not sent for the 2010 status code.

Active/Active

The split mode traps are identical to combined mode traps in an Active/Active environment (including GEO A/A). The SNMP agent does not break down A/A high level status codes into ESC components.

SNMP Agent Internal Traps

The SNMP agent traps are also sent out for erroneous conditions. SNMP agent traps generally refer to internal connectivity errors. The following SNMP agent traps are sent when they are received and when the situation resolves itself:

Table 11: SNMP Agent Internal Traps

Condition	DOWN/UP Code	Message
ESC Health Monitor - HTTP Error	2100/5100	A HTTP error was received when using the ESC Monitor API (*HTTP error included in the message*)
ESC Health Monitor - Unknown response	2101/5101	The ESC Monitor replied, but the data could not be understood (*data included in the message*)
ESC Health Monitor - Health Monitor is down.	2102/5102	Could not connect to ESC Monitor.
ESC Health Monitor - Un-identified error.	2199/5199	An unhandled error occurred *(details will be included in the message)*
HA Node change.	5200	HA ACTIVE node change (5200 is only valid in an HA environment, and there is no equivalent "up" trap. To the end SNMP client, when the SNMP Agent is configured for split traps and there is an HA node change - only the single 5200 trap sent as per previous functionality."

As these codes denote rare situations and have variable messages, the message in the SNMP trap does not change (unlike the ESC component messages), but the situation and resolution can be detected from the code. The 5 series code denotes an erroneous situation, and 2 series code indicates the previous situation has corrected itself.

Duplicate and Missing SNMP Traps

When the SNMP agent is constantly polling the status of all ESC components, it does not persist the ESC component status. Therefore, if the SNMP agent is restarted, it loses its previous view of the ESC component statuses. This creates two possible scenarios:

- **Duplicate SNMP Traps**: The SNMP agent can send a duplicate SNMP trap if the components are down before the SNMP agent is restarted. These duplicate SNMP traps are sent in rare situations.

  For example, if the ESC Manager is down and the SNMP agent is restarted, the following traps would be generated:

  5010 - Down, ESC Manager

  - SNMP Agent goes down
  - SNMP Agent comes up, fetches ESC component status, notes ESC Manager is down and generates a duplicate SNMP Trap

  5010 - Down, ESC Manager

- **Missing SNMP Traps**: The SNMP agent may not send out an SNMP trap which should have been generated for an ESC component status change when the SNMP agent is down. It is possible that valid SNMP traps cannot be sent in rare situations.

- For example, if ESC Manager is down and the SNMP agent is restarted, the following traps would be generated:

  5010 - Down, ESC Manager

  - SNMP Agent goes down
  - ESC Manager comes up, SNMP Agent **does not send 2010**
  - SNMP Agent comes up, fetches status, notes ESC is healthy and sends a single trap, even though it missed the ESC Manager UP trap

  2000 - Up, all ESC services

  To manage this scenario, the SNMP agent always generates a trap when it is restarted, and if the trap is for the status code "2000 - ESC Services are running.", then any previous unacknowledged traps must be cleared by the end client.

Managing Self-Signed Certificates

When ESC is deployed and the SNMP agent uses ESC Health APIs, it is recommended that a root trusted certificate is installed on the server. If the environment is a known and trusted one then it is possible to ignore these errors using the configuration parameter "ignoreSslErrors". However, if you did want to keep this setting to its more secure default it is possible to install a self-signed certificate by importing the ESC certificate into the JVM trust store. The following section describes the procedure to do so.

Step 1 Add esc as an alternative name for localhost. In the file "/etc/hosts:" add the following (or ensure that "esc" is added to the end):

Example:

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 esc
```

Step 2 In the SNMP Agent configuration file "/opt/cisco/esc/esc_database/snmp.conf" the healthUrl must point to esc.

```
"healthUrl": "https://esc:8060:/esc/health"
```

Step 3 Import the certificate into the truststore. Following is an example of importing the certificate, assuming $JAVA_HOME is/usr/lib/jvm/jre-1.8.0-openjdk.x86_64:

```
cd /opt/cisco/esc/esc-config
sudo openssl x509 -inform PEM -in server.pem -outform DER -out server.cer
sudo keytool -importcert -alias esc -keystore $JAVA_HOME/lib/security/cacerts -storepass changeit
-file server.cer
```

CHAPTER 4

ESC System Logs

Viewing ESC Log Messages

Log messages are created for ESC events throughout the VNF lifecycle. These can be external messages, messages from ESC to other external systems, error messages, warnings, events, failures and so on. The log file can be found at `/var/log/esc/escmanager_tagged.log`.

The log message format is as follows:

```
date=<time-date>] [loglevel=<loglevel>] [tid=<transactionid>] [cl=<classifications>]
[tags=<tags>] [msg=<message>
```

Sample log is as follows:

```
date=15:43:58,46022-Nov-2016]
[loglevel=ERROR ] [tid=0793b5c9-8255-47f3-81e6-fbb59f6571f7] [cl=OS ]
[tags=wf:create_vm,eventType:VM_DEPLOY_EVENT,tenant:CSCvd94541,depName:test-dep,vmGrpName:test-VNF,
vmName:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0]
[tags=wf:create_vm,eventType:VM_DEPLOY_EVENT,tenant:test,depName:test-dep,vmGrpName:test-VNF,
vmName:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0] [msg=sleepingfor5seconds
to allow vm to become ACTIVE instance id:
162344f7-78f9-4e45-9f23-34cf87377fa7
name:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0
```

When a request is received, a RequestDetails object is created which autogenerates a unique transaction id. This value is carried forward across all threads. Classifications and tags are optional. These are prefixes added to the log messages to enhance readability, and help in debugging. With classifications and tags, the log messages can be easily parsed and filtered by the log analysis tools.

The following classifications are supported:

NBI	"com.cisco.esc.rest""com.cisco.esc.filter"(North Bound Interface - Clientinterface)
SBI	"com.cisco.esc.rest"- source is a callback handler or"EventsResource"(South Bound Interface - i.e. between ESC and the VIM)
SM	"com.cisco.esc.statemachines". stands for StateMachine. This classification indicates logs in the StateMachine category.

MONITORING	"com.cisco.esc.monitoring""com.cisco.esc.paadaptor"(MONA related logs)
DYNAMIC_MAPPING	"com.cisco.esc.dynamicmapping""com.cisco.esc.db.dynamicmapping"(MONA related logs)
CONFD	"com.cisco.esc.confd"
CONFD_NOTIFICATION	"com.cisco.esc.confd.notif""com.cisco.esc.confd.ConfdNBIAdapter"
OS	"com.cisco.esc.vim.openstack"
LIBVIRT	"com.cisco.esc.vim.vagrant
VIM	"com.cisco.esc.vim"
REST_EVENT	"ESCManager_Event""com.cisco.esc.util.RestUtils". indicates REST notifications in logs.
WD	"com.cisco.esc.watchdog"
DM	"com.cisco.esc.datamodel""com.cisco.esc.jaxb.parameters"(Datamodel and resource objects)
DB	"com.cisco.esc.db"(Database related logs)
GW	"com.cisco.esc.gateway"
LC	"com.cisco.esc.ESCManager"(Start up related logs)
SEC	"com.cisco.esc.jaas"
MOCONFIG	"com.cisco.esc.moconfig"(MOCONFIG object related logs --this is internal for ESC developers)
POLICY	"com.cisco.esc.policy"(Service/VM Policy related logs)
TP	"com.cisco.esc.threadpool"
ESC	"com.cisco.esc" Any other packages not mentioned above

The following tags are supported:

- **Workflow [wf:]**—Generated using action and resource from RequestDetails object. Example "wf:create_network"
- **Event type [eventType:]**—Event that triggered the current action. Example: "eventType:VM_DEPLOY_EVENT"
- **Resource based**—These values are generated based on the type of parameter used by the event. The hierarchy, that is, the tenant, the vm group and so on is added to the log.

Tenant	[tenant:<tenant name>]
Network	[tenant:<tenant id>, network:<network name>] **Note** The tenant appears only if applicable.

Subnet	[tenant:<tenant name or id>, network:<network name or id>, subnet:<subnet name>] **Note** The tenant appears only if applicable.
User	[tenant:<tenant name>, user:<user name or id>] **Note** The tenant appears only if applicable.
Image	[image:<image name>]
Flavor	[flavor:<flavor name>]
Deployment	[tenant:<tenant name or id>, depName:<deployment name>]
DeploymentDetails	[tenant:<tenant name or id>, depName:<deployment name>, vmGroup:<vm group name>, vmName:<vm name>]
Switch	[tenant:<tenant name or id>, switch:<switch name>]
Volume	[volume:<volume name>]
Service	[svcName:<Service Registration name>]

Further, ESC logs can also be forwarded to an rsyslog server for further analysis and log management.

Filtering Logs Using Confd APIs

You can query and retrieve logs (for example, deployment logs, or error logs) in ESC using log filters introduced in the confd APIs. New filters for Tenant, Deployment Name, and VM Name are introduced. This enables you to query the ESC logs further for most recent error logs using the log filters in Confd APIs. You can also retrieve ESC logs related to the communication between ESC and the OS (by setting the classification tag to "OS").

The log format to retrieve confd API logs:

```
date=<time-date>] [loglevel=<loglevel>] [tid=<transactionid>] [cl=<classifications>]
[tags=<tags>] [msg=<message>
```

The sample log is as follows:

```
date=15:43:58,46022-Nov-2016] [loglevel=ERROR ] [tid=0793b5c9-8255-47f3-81e6-fbb59f6571f7]
 [cl=OS ]
[tags=wf:create_vm,eventType:VM_DEPLOY_EVENT,tenant:test,depName:test-dep,vmGrpName:test-VNF,
vmName:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0]
[msg=sleepingfor5seconds to allow vm to become ACTIVE instance id:
162344f7-78f9-4e45-9f23-34cf87377fa7 name:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0
```

The parameters for log level, classification and tags are dependent on each other to retrieve the logs. You can successfully retrieve the logs with the following combination.

- log_level=ERROR, classifications=OS, tags=(depName:test-dep)
- log_level=ERROR, classifications=OS, tags=(tenant: test)

The log filter returns a value when all the following conditions are met:

- Log level
- Classifications (if provided)

- Tags (if provided)

Note If there are more than one classification listed, it has to match at least one of the classifications. The same applies to the tags as well.

For example, the following log filter criteria does not return the log sample mentioned earlier:

```
log_level=ERROR, classifications=VIM, tags=(depName:test-dep)
```

It does not return any value though the log level and tags match, the classification VIM does not match.

The data model is as follows:

```
rpc filterLog {
    description "Query and filter escmanager logs using given parameters";
    tailf:actionpoint escrpc;
    input {
      leaf log_level {
        mandatory false;
        description "One of DEBUG / INFO / WARNING / ERROR / TRACE / FATAL. Results will
include all logs at and
                     above the level specified";
        type types:log_level_types;
        default ERROR;
      }
      leaf log_count {
        mandatory false;
        description "Number of logs to return";
        type uint32;
        default 10;
      }
      container classifications {
        leaf-list classification {
         description "Classification values to be used for the log filtering.  For example:
 'OS', 'SM'.
                      Logs containing any of the provided classification values will be
returned.";
          type types:log_classification_types;
        }
      }
      container tags {
        list tag {
          key "name";
          leaf name {
            mandatory true;
           description "Tag name to be used for the log filtering.  For example: 'tenant',
 'depName'.
                         Logs containing any of the provided tag name plus the tag values
will be returned.";
            type types:log_tag_types;
          }
          leaf value {
            mandatory true;
            description "Tag value pairs to be used for the log filtering.  For example:
'adminTenant', 'CSRDeployment'";
            type string;
          }
        }
      }
    }
    output {
      container filterLogResults {
        leaf log_level {
          description "Log level used to filter for the logs.";
          type types:log_level_types;
        }
        list logs {
          container classifications {
            leaf-list classification {
              description "Classifications used to filter for the logs.";
              type types:log_classification_types;
            }
          }
          container tags {
            list tag {
              key "name";
              leaf name {
                mandatory true;
                description "Tag name used to filter for the logs.";
                type types:log_tag_types;
              }
              leaf value {
                mandatory true;
                description "Tag value used to filter for the logs.";
                type string;
              }
            }
          }
          leaf log_date_time {
            description "Timestamp of the log.";
            type string;
          }
          leaf log_message {
            description "The log message.";
            type string;
          }
        }
      }
    }
  }
```

You can query for the confd API logs through the netconf console or esc_nc_cli

- Through the netconf-console, run the following query:

  ```
  /opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=admin
  --privKeyFile=/home/admin/.ssh/confd_id_dsa --privKeyType=dsa --rpc=log.xml
  ```

- Using the esc_nc_cli, run the following query:

  ```
  esc_nc_cli --user <username> --password <password> filter-log log.xml
  ```

  The sample log.xml is as follows:

  ```
  <filterLog xmlns="https://www.cisco.com/esc/esc">
    <log_level>INFO</log_level>
    <log_count>1</log_count>
    <classifications>
      <classification>OS</classification>
      <classification>SM</classification>
    </classifications>
    <tags>
      <tag>
        <name>depName</name>
        <value>CSR_ap1</value>
      </tag>
  ```

```
      <tag>
        <name>tenant</name>
        <value>admin</value>
      </tag>
    </tags>
  </filterLog>
```

The response is as follows:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <filterLogResults xmlns="https://www.cisco.com/esc/esc">
    <log_level>INFO</log_level>
    <logs>
      <classifications>
        <classification>OS</classification>
        <classification>SM</classification>
      </classifications>
      <tags>
        <tag>
          <name>depName</name>
          <value>CSR_ap1</value>
        </tag>
        <tag>
          <name>tenant</name>
          <value>admin</value>
        </tag>
      </tags>
      <log_date_time>13:06:07,575 31-Oct-2016</log_date_time>
      <log_message> No pending work flow to start.</log_message>
    </logs>
  </filterLogResults>
</rpc-reply>
```

Note The logging API responses are in XML format. If the log messages contain any XML characters, then the characters will be escaped so not to break the XML conformance.

Viewing ESC Log Files

You can find the logs of various ESC components here:

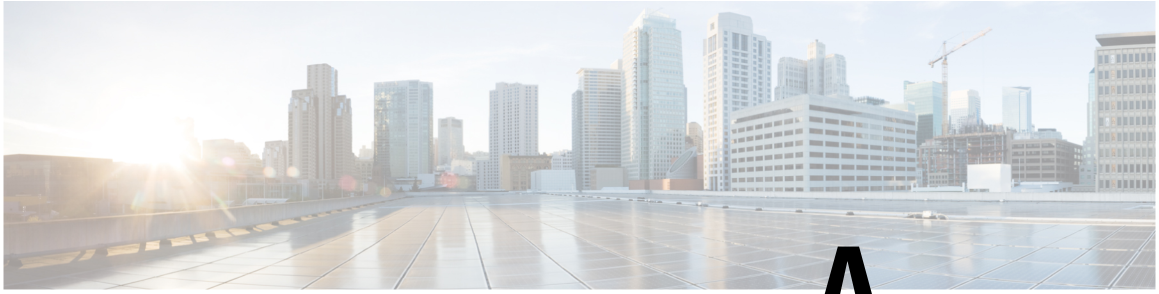
File	Component	Description	Rotation Size	Number of backup files	Active/Active deployment
/var/log/esc/escmanager.log	ESCManager	This contains logs of the ESC Manager which includes workflow, request and persistence.	150 MB	10	Available

File	Component	Description	Rotation Size	Number of backup files	Active/Active deployment
/var/log/esc/escmanager_tagged.log	ESCManager	This is the same as escmanager.log but with a format that is easier to read for netconf logging API but can be used by other parsers if needed.	150 MB	10	Available
/var/log/esc/yangesc.log	ESCManager	This contains logs related to netconf request and notifications	150 MB	10	Available
/var/log/esc/error_escmanager.log	ESCManager	All error log entries.	150 MB	10	Available
/var/log/esc/trace/event_escmanager.log	ESCManager		150 MB	10	Available
/var/log/esc/trace/escdatabase.log	ESCManager	Database related log entries			Available
/var/log/esc/trace/debug_yangesc.log	ESCManager		51 MB	2	Available
/var/log/esc/trace/esc_rest.log	ESCManager		150 MB	10	Available
/var/log/esc/mona/mona.log	MONA		150 MB	10	Available
/var/log/esc/mona/actions_mona.log	MONA		150 MB	10	Available
/var/log/esc/mona/rules_mona.log	MONA		150 MB	10	Available
/var/log/esc/vimmanager/vimmanager.log	VIM Manager Service	A detailed VIM manager log.	150 MB	10	Available

File	Component	Description	Rotation Size	Number of backup files	Active/Active deployment
/var/log/esc/vimmanager /operations_vimmanager.log	VIM Manager Service	A simplified log which only lists the VIM Manager operations been processed.	150 MB	10	Available
/var/log/esc/trace/vimmanager /vim_vimmanager.log	VIM Manager Service	Raw HTTP request/response (including header) between VIM Manager and VIM. Note, for OpenStrack to track this log, log level has to set to DEBUG for VIM. Manager.	150 MB	10	Available
/var/log/esc/<timestamp>-esc-portal-be.log	ESC UI		10 MB	4	Available
/var/log/esc/confd/audit.log	confd		10 MB	4	Available
/var/log/esc/confd/browser.log	confd		10 MB	4	Available
/var/log/esc/confd/confd.log	confd		10 MB	4	Available
/var/log/esc/confd/devel.log	confd		10 MB	4	Available
/var/log/esc/confd/netconf.log	confd		10 MB	4	Available
/var/log/esc/confd/netconf.trace	confd		10 MB	4	Available
/var/log/esc/confd/global.data			Not rotated		Available
/var/log/esc/esc_monitor.log	ESC INFRA or HA		10 MB	4	Not available
/var/log/esc/esc_monitor_output.log	ESC INFRA or HA		10 MB	4	Not available

File	Component	Description	Rotation Size	Number of backup files	Active/Active deployment
/var/log/esc/esc_confd.log	ESC INFRA or HA		10 MB	4	Not available
/var/log/esc/pgstartup.log	ESC INFRA or HA		10 MB	4	Not available
/var/log/esc/spy.log	ESC INFRA or HA		No logs (size 0)	No ESC generated logs.	Not available
/var/log/esc/catalina.out	Tomcat		Not rotated	No ESC generated logs. Only Error.	Available
/var/log/esc/esc_dbtool.log	DB tool		Not rotated		Available
/var/log/esc/snmp/snmp.log	SNMP Agent		Not rotated		Not available
/var/log/esc/etsi-vnfm/etsi-vnfm.log	ETSI-Service	This is the main log file for ETSI processing, including requests, response, payloads and general logging information where appropriate.	150MB	10	Available
/var/log/esc/etsi-vnfm/events-etsi-vnfm.log	ETSI-Service	Logs only API requests, both entering and leaving.	150MB	10	Available

File	Component	Description	Rotation Size	Number of backup files	Active/Active deployment
/var/log/esc/etsi-vnfm/event-details-etsi-vnfm.log	ETSI-Service	Logs both the API requests (entering and leaving) along with the actual JSON payloads.	150MB	10	Available
/var/log/esc/escadm.log	Escadm service	Logs to capture both manual and automated messages and errors from escadm.py. This log is useful to track startup and configuration changes to ESC.	10 MB	4	Available
/var/log/esc/elector.log	Elector service	Log entries records leadership decisions.	150 MB	10	Available for Active/Active only
/var/log/esc/consul_agent.log	Consul agent	Log entries recording ESC Consul agent with Consul server.	150MB	10	Available for Active/Active only
/var/log/esc/geo.log	GEO service	Log entries records GEO states and transitions	150MB	10	Available for GEO Active/Active only



APPENDIX A

ESC Error Conditions

Error Conditions for ESC Operations

Error Conditions for ESC Operations

If an operation fails in ESC, the user must cancel that operation. ESC will not rollback automatically to cancel any operations. The table below shows the error condition, and recovery details.

Notifications or Logging details for Error Conditions

Typically, for all error conditions, an error notification of the failed request will be sent to the NB client (ESC User) through callback if using REST interface, or through netconf notification if using NETCONF interface. An error log will be generated and sent to syslog, if syslog is configured.

Error Condition	Recovery
Failed create tenant request	NB client (ESC User) has to send in a delete tenant request before attempting to send in the same create tenant request.
Failed create network request	NB client (ESC User) has to send in a delete network request before attempting to send in the same create network request.
Failed create subnet request	NB client (ESC User) has to send in a delete subnet request before attempting to send in the same create subnet request.
Failed deployment request	NB client (ESC User) has to send in an undeploy request before attempting to send in the same deploy request If a deployment fails, ESC updates information in its database (with error state) until it receives an undeployment request. The undeployment will remove objects that are in error states.

Error Condition	Recovery
Failed Recovery	The existing deployment is not usable anymore. NB client (ESC User) has to send in an undeploy request then the same deploy request.
Failed Scale Out/In	No action required. The existing deployment is still functional. If at a later stage an undeploy was triggered, it will clean up any VMs that were affected part of the failed scale out and scale in.
Failed Service Update	No action required. The existing deployment is still functional. Any retries of that update will not be honored. If at a later stage an undeploy was triggered, it will clean up any created VMs part of the failed update.
Failed VM Operations (Start, Stop, Reboot, Enable Monitor, Disable Monitor)	No action required. The existing deployment is still functional. NB client (ESC User) can retry the failed operation.
Failed VNF/Service Operations (Start, Stop, Reboot, Enable Monitor, Disable Monitor)	No action required. The existing deployment is still functional. NB client (ESC User) can retry the failed operation.
Failed delete tenant request	Possibility of leaking resource in VIM. Manual intervention might be needed to clean up leaking resources on VIM.
Failed delete network request	Possibility of leaking resource in VIM. Manual intervention might be needed to clean up leaking resources on VIM.
Failed delete subnet request	Possibility of leaking resource in VIM. Manual intervention might be needed to clean up leaking resources on VIM.
Failed undeployment request	Possibility of leaking resource in VIM. Manual intervention might be needed to clean up leaking resources on VIM

APPENDIX B

Before Contacting Tech Support

At some point, you might need to contact your technical support representative or Cisco TAC for some additional assistance. This section outlines the steps that you should perform before you contact your next level of support in order to reduce the amount of time spent resolving the issue.

Downloading Logs from the ESC

You can download log files from the ESC for troubleshooting.

To collect log files through CLI, use the following command:

```
sudo escadm log collect
```

To collect configuration data for VMs, use the following command,

```
esc_nc_cli --user <username> --password <password> get-config esc_datamodel > <file-name>
```

For example:

```
esc_nc_cli --user <username> --password <password> get-config esc_datamodel >
/var/tmp/esc_datamodel.txt
```

To collect log files through CLI from ESC Active/Active HA, use the following command:

```
esc_nc_cli --host db.service.consul --user admin --password password get-config esc_datamodel
```

For more information of the ESC system level configuration, see Downloading Logs from the ESC Portal section in the Cisco Elastic Services Controller User Guide.

Things To Do Before Calling TAC

Answer the following questions before you contact your technical support representative:

1. Collect the system information and configuration through CLI (system log files) and through GUI. For instructions, refer Downloading the log files.
2. If an error occurs in ESC, take a screen shot of the error. In Windows, press Alt+PrintScreento capture the active window, or press PrintScreen to capture the entire desktop. Paste the screenshot into a new Microsoft Paint (or similar program) session and save the file.

3. Capture the exact error codes that you see in the message logs from either ESC or the CLI.
4. Answer the following questions before you contact your technical support representative:
   - Which ESC version, operating systems versions, and storage device firmware are in your network?
   - Were any changes made to the environment (VLANs, upgrades, or adding modules) prior to or at the time of this event?
   - Are there other similarly configured devices that could have this problem but do not?
   - Where was this problematic device connected (which device and interface)?
   - When did this problem first occur?
   - When did this problem last occur?
   - How often does this problem occur?
   - Were any traces or debug output captured during the problem time?
   - What troubleshooting steps have you attempted?
5. Answer the following questions if your problem is related to a software upgrade attempt:
   - What was the original Cisco ESC version?
   - What is the new Cisco ESC version?
   - Collect the output from the following command and forward them to your customer support representative:
     - `esc_nc_cli --user <username> --password <password> get-config esc_datamodel > <file-name>`
     - `esc_version`
     - `health.sh`
     - `escadm status`
     - `escadm vim show`