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, on page 1
- Monitoring the Health of ESC Using SNMP Trap Notifications, on page 8
- Managing SNMP Traps in ESC, on page 12
- Managing Self-Signed Certificates, on page 25

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 1: ESC Health API Status Code and Messages in Standalone and Active-Standby High Availability, on page 3 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 3: Health API Status Messages for Standalone ESC and HA, on page 5 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 1: 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 2: 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 3: 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 8.

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 20.

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 20.

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 4: 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 5: 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 6:

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 1.

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 7: 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 8:

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 9: 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
```