THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: http://www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

© 2017 Cisco Systems, Inc. All rights reserved.
# CONTENTS

## CHAPTER 1

**Preface** 1  
  Audience 1  
  Terms and Definitions 1  
  Obtaining Documentation Request 2

## CHAPTER 2

**Elastic Services Controller Overview** 3

## PART I

**Installing Cisco Elastic Services Controller on OpenStack** 5

## CHAPTER 3

**Prerequisites** 7  
  Virtual Resource Requirements 7  
  Software Requirements 7  
  Preparing for the Installation 8

## CHAPTER 4

**Installing Cisco Elastic Services Controller on OpenStack** 9  
  Installation Scenarios 9  
  Main Components of Cisco Elastic Services Controller Setup 10  
  Installing Cisco Elastic Services Controller Using the QCOW Image 10  
  Additional Installation Options 13  
  Managing Root Certificates in Cisco Elastic Services Controller 16  
  Enabling/Disabling the Root Certificate Validation 16  
  Adding a Root Certificate 16  
  Removing a Root Certificate 17  
  Managing Root Certificates During the Upgrade 17  
  Using a Bootable Volume in ESC Installation 17

## CHAPTER 5

**Configuring High Availability** 19  
  High Availability Overview 19
How High Availability Works  19
Deploying ESC High Availability  20
  Deploying ESC in High Availability Mode on Internal Storage  21
  Deploying ESC in High Availability Mode on Replicate External Storage  22
Configuring the Northbound Interface Access  23
  Configuring ESC HA with Multiple Interfaces  24
  Configuring the ESC HA Virtual IP Address  24
  Configuring the ESC HA BGP Anycast IP Address  25
Important Notes  26
Troubleshooting High Availability  27

PART II
Installing Cisco Elastic Services Controller on VMware vCenter  29

CHAPTER 6
Prerequisites  31
  Virtual Resource and Hypervisor Requirements  31
  vCenter Resources  32
  Important Notes  32

CHAPTER 7
Installing Cisco Elastic Services Controller on VMware vCenter  35
  Installing Cisco Elastic Services Controller on VMware vCenter  35
    Preparing to Install Cisco Elastic Services Controller  35
    Installing the Elastic Services Controller Using the OVA Image  36
    Installing Elastic Services Controller Using OVF Tool  38
    Powering on Cisco Elastic Services Controller Virtual Machine  41
    Next Steps: Cisco Elastic Services Controller Virtual Machine  41
      Logging in to Cisco Elastic Services Controller Portal  41
      Configuring the Virtual Machine to Automatically Power Up  41

CHAPTER 8
Configuring High Availability  43
  High Availability Overview  43
  How High Availability Works  43
  Deploying ESC High Availability  44
  Important Notes for ESC HA  45
  Troubleshooting High Availability  45
PART III

CHAPTER 9

Installing Cisco Elastic Services Controller on a Kernel-based Virtual Machine (KVM) 47

Preparing to Install Cisco Elastic Services Controller on a Kernel-based Virtual Machine 49

Installing Elastic Services Controller on a Kernel-Based Virtual Machine 50

Next Steps: Cisco Elastic Services Controller Kernel-based Virtual Machine 51

Verifying ESC installation for a Kernel-based Virtual Machine (KVM) 51

Troubleshooting Tips 52

PART IV

CHAPTER 10

Post Installation Tasks 55

Changing the ESC Password 55

Changing the Confd Netconf/CLI Administrator Password Using the Command Line Interface 55

Changing Linux Account Password 56

Changing the ESC Portal Password 56

Configuring Pluggable Authentication Module (PAM) Support for Cisco Elastic Services Controller 57

Authenticating REST Requests 57

Enabling REST Authentication 58

Changing the REST Interface Password 58

Sending an Authorized REST Request 59

Configuring Openstack Credentials 59

Reconfiguring ESC Virtual Machine 64

Reconfiguring Rsyslog 64

Reconfiguring NTP 65

Reconfiguring DNS 66

Reconfiguring Hosts 66

Reconfiguring Timezone 66

Verifying ESC Configurations and Other Post-Install Operations 67
Contents

PART V Upgrading Cisco Elastic Services Controller 71

CHAPTER 11 ESC in Maintenance Mode 73

Setting ESC in a Maintenance Mode 73
Using the escadm Tool 73
Setting ESC in an Operation Mode 74
Backup the Database from the ESC Standalone Instances 74
Backup the Database from the ESC HA Instances 75
Restoring ESC Database 77

CHAPTER 12 Upgrading Cisco Elastic Services Controller 79

Upgrading Standalone ESC Instance 80
Deploy the ESC for Upgrade 80
Restoring the ESC Database 81
Important Notes: 81
Upgrading ESC HA Instances 81
Deploying the ESC HA nodes for Upgrade 82
Restoring the ESC Database on New Master and Standby ESC Instances 82
Upgrading VNF Monitoring Rules 83
In-Service Upgrade of the ESC HA Nodes in OpenStack 84
In-Service upgrade in OpenStack using ESC RPM packages 84
In-Service upgrade in OpenStack using ESC qcow2 Image 85
In-Service Upgrade of the ESC HA Nodes in Kernel-Based Virtual Machine (KVM) 88
In-Service upgrade in KVM using ESC RPM packages 88
In-Service upgrade in KVM using ESC qcow2 Image 89
In-Service Upgrade of the ESC HA Nodes in VMware 92
In-Service upgrade in VMware using ESC RPM packages 92
In-Service upgrade in VMware using ESC qcow2 Image 93

APPENDIX A Cisco Elastic Services Controller Installer Arguments 97
Cisco Elastic Services Controller Installer File Reference 103
ESC Configuration Parameters 103
Preface

The Cisco Elastic Services Controller Install and Upgrade Guide describes the installation requirements, the installation procedures, and the upgrade procedures for Cisco Elastic Services Controller.

This preface contains the following sections:

- Audience, page 1
- Terms and Definitions, page 1
- Obtaining Documentation Request, page 2

Audience

This guide is for network administrators who are installing, provisioning, configuring, and monitoring Virtual Network Functions (VNFs). ESC can be deployed on OpenStack, VMware vCenter, and KVM. The administrator must be familiar with the VIM layer, VMware, and OpenStack resources, and the commands used.

Terms and Definitions

The below table defines the terms used in this guide.

Table 1: Terms and Definitions

<table>
<thead>
<tr>
<th>Terms</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HA</td>
<td>ESC High Availability (HA) is a solution for preventing single points of ESC failure and achieving minimum ESC downtime.</td>
</tr>
<tr>
<td>KPI</td>
<td>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.</td>
</tr>
<tr>
<td>NFV</td>
<td>Network Function Virtualization (NFV) is the principle of separating network functions from the hardware they run on by using virtual hardware abstraction.</td>
</tr>
</tbody>
</table>
Definitions

<table>
<thead>
<tr>
<th>Terms</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFVO</td>
<td>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.</td>
</tr>
<tr>
<td>NSO</td>
<td>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.</td>
</tr>
<tr>
<td>Service</td>
<td>A Service consists of single or multiple VNFs.</td>
</tr>
<tr>
<td>VIM</td>
<td>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 &amp; performance alarms.</td>
</tr>
<tr>
<td>VM</td>
<td>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.</td>
</tr>
<tr>
<td>VNF</td>
<td>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.</td>
</tr>
<tr>
<td>VNFM</td>
<td>Virtual Network Function Manager (VNFM) manages the life cycle of a VNF.</td>
</tr>
<tr>
<td>vMS</td>
<td>Cisco vMS is a Network Functions Virtualization (NFV) orchestration platform that enables fast deployment of cloud-based networking services.</td>
</tr>
</tbody>
</table>

Obtaining Documentation Request


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.
Elastic Services Controller Overview

Cisco Elastic Services Controller (ESC) is a Virtual Network Functions Manager (VNFM), which performs life cycle management of Virtual Network Functions (VNFs). ESC provides agent-less and multi vendor VNF management by provisioning virtual services, and monitoring their health and load. ESC provides the flexibility to define monitoring rules, and associate actions to be triggered based on the outcome of these rules. As a VNFM, in addition to the typical life cycle management operations, ESC also supports automatic VM recovery when a VM fails and performs automatic scaling in and out functions. ESC fully integrates with Cisco and other third party applications.

- As part of the Cisco Orchestration Suite, ESC is packaged with Cisco Network Services Orchestrator (NSO), and available within Cisco Solution, Virtual Managed Services (vMS).
- As a standalone product, ESC is available as a Virtual Network Function Manager for several Cisco VNFs such as CSR1K, ASAv, WSA and many others.

ESC is deployed in a virtual machine within OpenStack, VMware vCenter, or KVM and manages its VNFs in a Virtual Infrastructure Manager (VIM). ESC 2.0 and later supports OpenStack and VMware vCenter.

ESC fully integrates with Cisco and other third party applications. As a standalone product, the Elastic Services Controller can be deployed as a VNF Manager. ESC integrates with Network Services Orchestrator (NSO) to provide VNF management along with orchestration. Elastic Services Controller as a VNF Manager targets the virtually managed services and all service provider NFV deployments such as virtual video, WiFi, authentication and others.

ESC can manage both basic and complex VNFs. Basic VNFs include a single VM such as a vFW, vRouter and others.

Complex VNFs include multiple VMs that are orchestrated as a single entity with dependencies between them.

Elastic Services Controller provides IPv6 support on OpenStack for:

- VNF Management
- HA— ESC HA deployment is also supported on IPV6 environment. Also, while booting instances using multiple interfaces, you can assign each heartbeat or DB synch to IPv4 or IPv6.

ESC does not support IPv6 on VMware.
Elastic Services Controller provides IPv6 support for northbound interface (for example, NFVO to VNFM), and southbound interface (for example, VNFM to VNF). However, the following pre-requisites must be met:

- OpenStack cloud computing (such as Ocata) is set up and configured for ipv6, including the endpoints (that are ipv6 based).
- The OpenStack cloud computing must contain a Controller, endpoints, and a few Compute hosts, with an ipv6 management and os_api based networks.
- The ESC default security group rules support the IPv6 traffic.

Elastic Services Controller shows the IPv6 address in the GUI wherever applicable like dashboards, deployment screens, and reports.

**Note**

When you are deploying a VM, you can attach an out-of-band port of an IPv6 subnet to a VM. However, if you are deleting this VM, you cannot attach the same IPv6 address to another VM due to a known OpenStack issue.
Installing Cisco Elastic Services Controller on OpenStack

- Prerequisites, page 7
- Installing Cisco Elastic Services Controller on OpenStack, page 9
- Configuring High Availability, page 19
Prerequisites

The following sections detail the prerequisites for installing Cisco Elastic Services Controller:

- Virtual Resource Requirements, page 7
- Software Requirements, page 7
- Preparing for the Installation, page 8

Virtual Resource Requirements

The following table lists the virtual resource requirements for Cisco Elastic Services Controller:

<table>
<thead>
<tr>
<th>Requirements (for 1000 VNFs/VM management)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual CPUs</td>
<td>4 VCPUs</td>
</tr>
<tr>
<td>Memory</td>
<td>8 GB RAM</td>
</tr>
<tr>
<td>Disk Space</td>
<td>30 GB</td>
</tr>
</tbody>
</table>

Software Requirements

The following table lists the software requirements on OpenStack:

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base OS</td>
<td>CentOS 7.3.1611</td>
</tr>
<tr>
<td>Supported Browsers</td>
<td>Any of the following:</td>
</tr>
<tr>
<td></td>
<td>• Mozilla Firefox 40.x or later</td>
</tr>
<tr>
<td></td>
<td>• Google Chrome 44.x or later</td>
</tr>
</tbody>
</table>
### Preparing for the Installation

Before you perform the installation, ensure that you are prepared by reviewing this checklist:

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Version</td>
<td>Any of the following:</td>
</tr>
<tr>
<td></td>
<td>• Liberty</td>
</tr>
<tr>
<td></td>
<td>• Mitaka</td>
</tr>
<tr>
<td></td>
<td>• Newton</td>
</tr>
</tbody>
</table>

#### For Pre-installation Configuration

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Your Information/ Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCOW image location</td>
<td></td>
</tr>
<tr>
<td>QCOW image</td>
<td></td>
</tr>
<tr>
<td>VM per Instance</td>
<td></td>
</tr>
<tr>
<td>IP address</td>
<td></td>
</tr>
<tr>
<td>Subnet mask</td>
<td></td>
</tr>
<tr>
<td>Hostname</td>
<td></td>
</tr>
<tr>
<td>Domain name</td>
<td></td>
</tr>
<tr>
<td>Gateway IP address</td>
<td></td>
</tr>
<tr>
<td>Admin password</td>
<td></td>
</tr>
</tbody>
</table>

#### ESC Release Package

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Your Information/ Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC.qcow2</td>
<td>An image file for booting up the ESC instance</td>
</tr>
<tr>
<td>bootvm.py</td>
<td>The installation script compatible with python 2.7.6 and Python 3.4</td>
</tr>
</tbody>
</table>
Installation Scenarios

The following sections briefly describe some of the common deployment scenarios that are addressed by ESC. Cisco Elastic Services Controller can be installed in different modes as per the requirement. These different modes are configured during installation. The following sections briefly describe some of the common deployment scenarios that are addressed by ESC.

ESC Standalone

In the standalone scenario, a single active VM is deployed for ESC.

ESC with HA

ESC supports High Availability (HA) in the form of a Primary and Standby model. Two ESC instances are deployed in the network to prevent ESC failure and to provide ESC service with minimum interruption. If the primary ESC instance fails, the standby instance automatically takes over the ESC services. ESC HA resolves the following single point failures of ESC:

• Network failures
• Power failures
• Dead VM instance
Main Components of Cisco Elastic Services Controller Setup

The Cisco Elastic Service Controller (ESC) setup has the following components:

- **Virtual Infrastructure Manager**—Elastic Services Controller (ESC) and its VNFs are deployed in a Virtual Infrastructure Manager (VIM). This might have one or more underlying physical nodes.

- **ESC Virtual Machine**—The ESC VM is a VM that contains all the services and processes used to register and deploy services. This includes the ESC Manager and all other services. ESC provides Netconf API, REST API, and Portal as north bound interfaces to communicate with ESC. ESC VM contains CLI to interact with ESC VM. There are two CLI, one uses the REST API and the other uses Netconf API.

Installing Cisco Elastic Services Controller Using the QCOW Image

You can install Cisco Elastic Services Controller (ESC) on OpenStack by using a QCOW image. ESC will be deployed in the OpenStack as running VM instance to manage VNFs. Therefore, ESC need OpenStack environment parameters to be installed in OpenStack. The installation time varies from 10 to 20 minutes, depending on the host and the storage area network load. This procedure describes how to create ESC Virtual Machine (VM) in OpenStack.

**Before You Begin**

- All system requirements are met as specified in Prerequisites.
- You have the information identified in Preparing for the Installation.
- Copy the ESC image file on the system where you want to install ESC.
- This system must be accessible by OpenStack.

**Step 1** Log in to the system where you want to install ESC.

**Step 2** Check the compatibility of the bootvm.py and the ESC image:

```bash
./bootvm.py --version
```

For more information on the ESC installer arguments, see Appendix: A Cisco Elastic Services Controller Installer Arguments.

**Step 3** In a text editor, create a file named PROJECT-openrc.sh file and add the following authentication information. The following example shows the information for a project called admin, where the OpenStack username is also admin, and the identity host is located at controller node.
Note To set the required environment variables for the OpenStack command-line clients, you must create an environment file called an OpenStack rc file, or openrc.sh file. This project-specific environment file contains the credentials that all OpenStack services use. The ESC installation script requires these OpenStack environment parameters to perform authentication and installation on OpenStack. If all the OpenStack credentials are passed through its own arguments, the bootvm.py script doesn't require these parameters.

```bash
export OS_NO_CACHE=true
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin_pass
export OS_AUTH_URL=http://controller node:35357/v2.0
```

The other OpenStack parameters required for installation are: --os_auth_url, --os_username, --os_password, --os_tenant_name, --bs_os_user_domain_name, --bs_os_project_domain_name, --bs_os_identity_api_version, --bs_os_auth_url, --bs_os_username, --bs_os_password, --bs_os_tenant_name, --bs_os_user_domain_name, --bs_os_project_domain_name, --bs_os_identity_api_version.

For OpenStack V2 API, you need following items to be defined in your global environment variables: --os_password, --os_auth_url, --os_username, --os_tenant_name.

For OpenStack V3 API, set --os_identity_api_version=3. Other parameters required for OpenStack V3 API are: --os_user_domain_name, --os_project_domain_name, --os_project_name, --os_password, --os_auth_url, --os_username, --os_identity_api_version, --os_ca_cert, --requests_ca_bundle.

Note The arguments, --os_tenant_name, --os_username, --os_password, --os_auth_url will also by default configure the VIM connector. If you want to skip configuring the VIM connector, pass the parameter (--no_vim_credentials) with the bootvm.py. When no_vim_credentials parameter is provided, the bootvm.py arguments (os_tenant_name, os_username, os_password, os_auth_url) are ignored. For more information on configuring VIM connectors after installation, and managing VIM connectors, see Managing VIM Connectors in the Cisco Elastic Services Controller User Guide.

Note --os_ca_cert and --requests_ca_bundle arguments are only required for https connection.

**Step 4**
On any shell from which you want to run OpenStack commands, source the PROJECT-openrc.sh file for the respective project. In this example, you source the admin-openrc.sh file for the admin project.

```bash
$ source admin-openrc.sh
```

**Step 5**
Check the environment variables.

```bash
$ env | grep OS_
```

**Step 6**
Register ESC image file in the OpenStack image using the glance command:

```bash
$ glance image-create \
--name <image_name> \
--is-public=<true or false> or --visibility public or private\n--disk-format <disk_format> \n--container-format <container_format> \n--file <file>\n--progress
```

An example configuration is shown below:

```bash
$ glance image-create \
--name esc-1_0_01_11_2011-01-01 \
--is-public=<true or false> or --visibility public or private \
--disk-format qcow2 \
--container-format bare \
--file esc-1_0_01_11_2011-01-01.qcow2 \
--progress
```
The `glance image-create` command is used to create a new image. The command takes the following arguments:

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the image.</td>
</tr>
<tr>
<td>is-public</td>
<td>(Optional) Makes the image accessible to the public.</td>
</tr>
<tr>
<td>disk-format</td>
<td>Disk format of the image. ESC uses a qcow2 disk format.</td>
</tr>
<tr>
<td>container-format</td>
<td>Container format of image. ESC uses a bare container format.</td>
</tr>
<tr>
<td>file</td>
<td>Local file that contains disk image to be uploaded during creation.</td>
</tr>
<tr>
<td>progress</td>
<td>(Optional) Shows upload progress bar.</td>
</tr>
</tbody>
</table>

To verify whether the image has been registered successfully:

a) Using OpenStack Controller dashboard:
   - Log into OpenStack using your credentials.
   - Navigate to **Admin > Image**.
   - Verify if the image appears in the list.

b) Using nova CLI:

   ```bash
   $ nova image-show <image_name>
   ```

**Step 7** The standard resource requirement of ESC is 4vCPU, 8G RAM, and 30GB disk space. ESC installation script takes the pre-defined "m1.large" flavor which has the definition of 4vCPU, 8G RAM and 80G disc space. To use 30GB disk space, create a flavor with the minimum disk space requirement.

   ```bash
   $ nova flavor-create ESC_FLAVOR_NAME ESC_FLAVOR_ID 8192 30 4
   ```

**Step 8** To deploy ESC VM, do the following:

1. Ensure that the existing network have connectivity to OpenStack controller. To verify the network connectivity using the nova CLI use:

   ```bash
   $ nova net-list
   ```

2. Record the ID of the network that ESC connects to boot the ESC VM by with image and flavor created earlier. The `bootvm.py` command requires at least one `--user_pass` argument to create an admin account for linux (ssh/console access) and at least one `--user_confd_pass` to create an admin account for ConfD (netconf/cli access). The following are the syntax for these mandatory user credential arguments:

   ```
   --user_pass admin:'PASSWORD-OR-HASH'[:OPTIONAL-PUBLIC-KEY-FILE][:OPTIONAL-ROLE]
   --user_confd_pass admin:'PASSWORD-OR-HASH'[:OPTIONAL-PUBLIC-KEY-FILE]
   ```

   To generate a password hash from a password, use the following command:

   ```bash
   mkpasswd --method=SHA-512 --salt Xyz123 <<< <Password>
   ```
The following is an example to install ESC with an authorized public key. In the following example, single quotes are used to avoid conflict with shell reserved characters:

```
--user_pass admin:'$algorithm$salt$hash-of-salt-password':$HOME/.ssh/esc_rsa.pub
--user_confd_pass admin:'$algorithm$salt$hash-of-salt-password':$HOME/.ssh/esc_rsa.pub
```

The public key are generated as part of a key pair with, such as:

```
ssh-keygen -t rsa -b 1024 -C "esc" -N "" -f ~/.ssh/esc_rsa
```

Your public key and identification key are saved in /home/username/.ssh/esc_rsa and esc_rsa.pub files. For more examples on the user credential arguments, see Appendix A: Cisco Elastic Services Controller Installer Arguments.

3 To check the details of ESC VM and get the information including the IP address(es) of the ESC VM, use the following command:

```
$ nova show <esc_vm_name>
```

### Additional Installation Options

- **Deploying ESC in an OpenStack IPv6 environment:** Before deploying ESC instance in IPv6, make sure to source openrc that supports ipv6 addresses. To deploy ESC in IPv6 environment, use the following bootvm arguments:

  ```
  ./bootvm.py <esc_vm_name-ipv6> --poll --user_rest_pass <username>:<password> --image <image_name> 
  --net <ipv6_network> --ipaddr <ipv6_ip_address> --enable-http-rest --user_pass <username>:<password> 
  --user_confd_pass <username>:<password> --etc_hosts_file <hosts-file-name> --route <default routing configuration>
  ```

- **Deploying ESC in DHCP mode:** If you use the bootvm.py argument without --ipaddr, then the ESC instance will be deployed in a DHCP mode. To deploy ESC in a DHCP network, use the following configuration:

  ```
  ./bootvm.py <esc_vm_name> --image <image_name> --net <network> --flavor <flavor_name> 
  --user_pass <username>:<password> 
  --user_confd_pass <username>:<password>
  ```

  **Note**: By default, ESC only support DHCP in IPv4 networks. If IPv6 is used, you need to log in the ESC VM and run "dhclient-6 ethX" (ethX is the V6 interface name) manually to enable V6 DHCP.

- **Using a Bootable Volume in ESC Installation:** You can attach a volume to an ESC instance and launch an instance from inside the volume. For more information, see the section Using a Bootable Volume in ESC Installation.

- **Assigning Floating IP to the ESC:** If you want to associate a floating IP with the ESC instance, do the following:

  1 Check for an available floating IP address and assign it to the ESC VM:

  ```
  $ nova floating-ip-list
  $ nova floating-ip-associate esc_vm_name <ip_address>
  ```
2. Or create a new floating IP address and assign it to the ESC VM:

```bash
$ nova floating-ip-create <FLOATING_NETWORK_ID>
$ nova floating-ip-associate esc_vm_name <ip_address>
```

or

```bash
neutron floatingip-create FLOATING_NETWORK
neutron floatingip-associate floating-ip-ID port-ID
```

- **Deploying ESC with static IPs:** To use ESC in a specific network with static IPs, for example, 192.168.0.112 at esc-net1, specify `--ipaddr` and `--gateway_ip` to the `bootvm` command line, as shown below:

```
./bootvm.py <esc_vm_name> --image <image_id> --net <network> --ipaddr <ip_address> --gateway_ip <default_gateway_ip_address> --user_pass <username>:<password>
```

**Note** Before assigning static IP, make sure the static IP is available and is not being used on other machine.

- **Deploying ESC with multiple network interfaces:** To use multiple networks for ESC, for example, 192.168.0.112 at esc-net1 and 10.20.0.112 at esc-net2, specify both the IP addresses and the network names of the interfaces in the `--net` and `--ipaddr` arguments in the following command line. In addition, also choose the default gateway for ESC from the gateways of these networks. Specify the default gateway for ESC through the `--gateway_ip` argument.

```
./bootvm.py <esc_vm_name> --image <image_id> --net <network1> <network2> --ipaddr <ip_address1> <ip_address2> --gateway_ip <default_gateway_ip_address> --user_pass <username>:<password>
```

```
```

**Note** If `--flavor` is not specified, `bootvm.py` will use the default flavor "m1.large" in OpenStack.

- **Deploy ESC with log forwarding options:** To forward ESC logs to an rsyslog server, specify the IP address of the rsyslog server while creating an ESC VM. Optionally, you can also specify the port and protocol to use.

For example, if the IP address of the rsyslog server is 10.85.77.94, the port on the server to forward logs is 514, and the protocol used is UDP, the ESC installation could be

```
./bootvm.py <esc_vm_name> --image <image_id> --net esc-net1 --rsyslog_server 10.85.77.94 --rsyslog_server_port 514 --rsyslog_server_protocol udp --user_pass <username>:<password>
```

- **Disabling the ESC GUI:** To boot up ESC VM with the graphical user interface disabled, modify the `--esc_ui_startup` argument value, as shown in the command line below:

```
./bootvm.py <esc_vm_name> --image <image_id> --net <network> --user_pass <username>:<password>
```

- **Enabling REST interface for ESC:** To support the REST interface, specify the `--enable-https-rest` argument. REST interface can be activated on both https or http:

```
./bootvm.py <esc_vm_name> --image <image_id> --net <network> --user_pass
```
<username>:<password>
--user_confd_pass <username>:<password> --enable-https-rest

OR

./bootvm.py <esc_vm_name> --image <image_id> --net <network> --user_pass
<username>:<password>
--user_confd_pass <username>:<password> --enable-http-rest

• Deploying ESC with global parameters: To set the global configurations through the esc_params_file during the installation, use the arguments as shown below. These global configurations can also be changed through REST API after the installation.

Note
The default security group is applied to the tenant during tenant creation. By default, the ESC configuration parameter for the security group, openstack.DEFAULT_SECURITY_GROUP_TO_TENANT is set to true. The configuration parameter must be set at the time of installation. You can query or update the parameter on ESC VM through the REST API. If the parameter is set to true, you can create and assign default security group during tenant creation. If the parameter is set to false, you cannot create or assign default security group during tenant creation. For details on the parameters that can be configured through esc_params_file, see Appendix A: Cisco Elastic Services Controller Installer Arguments.

/.bootvm.py <esc_vm_name> --image <image_id> --net <network> --flavor <flavor_name>
--user_pass <username>:<password>:<public key file> --user_confd_pass
<username>:<password>
--esc_params_file <esc parameter configuration file>

• Deploying two instances of ESC to build an ESC HA pair: For more information on deploying ESC HA, see Configuring High-Availability in Installing ESC on OpenStack and Installing ESC on VMware chapters.

• Adding a Dynamic Mapping File: In Cisco ESC Release 2.1 and earlier, mapping the actions and metrics defined in the datamodel to the valid actions and metrics available in the monitoring agent was enabled using the dynamic_mappings.xml file. The file was stored in the ESC VM and was modified using a text editor. ESC 2.2 and later do not have an esc-dynamic-mapping directory and dynamic_mappings.xml file. If you want to add an existing dynamic_mapping xml file to the ESC VM, do the following:

1 Backup this file to a location outside of ESC, such as, your home directory.
2 Create esc-dynamic-mapping directory on your ESC VM. Ensure that the read permissions are set.
3 Install on your ESC VM using the following bootvm argument:

   --file
   root:root:/opt/cisco/esc/esc-dynamic-mapping/dynamic_mappings.xml:<path-to-local-copy-of-dynamic-mapping.xml>

The CRUD operations for mapping the actions and the metrics is available through REST API. To update an existing mapping, delete and add a new mapping through the REST API.

• Changing the confd password on an ESC VM: As an administrator, you can configure the confd password through bootvm.py, during the installation time:

   ./bootvm.py --user_pass <username>:<password> --user_confed_pass
   admin:'PASSWORD-OR-HASH':OPTIONAL-PUBLIC-KEY

To reconfigure this password after the installation, execute the following commands:

   $ /opt/cisco/esc/confd/bin/confd_cli -u admin
   $ configure
Managing Root Certificates in Cisco Elastic Services Controller

Cisco Elastic Services Controller (ESC) provides a mechanism to enable verification of SSL certificates. Currently, this feature is supported only on OpenStack. Certificate validation is enabled by default during the initial ESC boot up. However, ESC also allows you to configure these SSL certificates. This section describes how to enable/disable certificate validation, add/remove, or list the certificates for Cisco Elastic Service Controller on OpenStack. You can add a root certificate during the ESC bootup or even after ESC bootup is completed.

Enabling/Disabling the Root Certificate Validation

Cisco Elastic Services Controller by default enables certificate validation. You can also enable or disable by modifying the parameter, DISABLE_CERT_VALIDATION, available under the Openstack category in the esc_params.conf file, or through the REST interface, or using the escadm tool. On ESC master node, use the command, escadm enable-certificate or escadm disable-certificate to enable and disable the certificate validation, respectively.

Adding a Root Certificate

You can add a root certificate during the ESC bootup or even after ESC bootup is completed. Before adding certificates, ensure the OpenStack environment file, OpenStack RC file has parameters to perform authentication and installation on OpenStack. The --os_auth_url must be specified while passing the parameters. --os_auth_url specifies the secure (https) or unsecured (http) keystone URL used by OpenStack for authentication.

- Add certificate for standalone (only) during the bootup time, i.e., during the ESC VM installation:

```bash
./bootvm.py test-vm --image <image_name> --net <network> --cert_file
/home/cisco/mitaka.crt
--user_pass <username>:<password> --user_confd_pass <username>:<password>
```

Currently, ESC does not support adding a certificate for HA during the installation as the keepalived service is not running when a certificate is added.

- Add certificate for standalone/HA after booting up the ESC instance. The escadm tool has an add-certificate option which has the following arguments: The --file argument refers to the CA certificate file. Using this argument you can import any file format supported by the java keytool: X.509 v1, v2, and v3 certificates, and PKCS#7. The --alias argument is unique and refers to the name this specific CA certificate is given.

1. Copy/Transfer CA Certificate file to ESC master VM.

For the ease of future upgrades, make sure that you keep a copy of all the commands and arguments that are used while installing ESC using the bootvm.py file.
2 Add certificate to ESC truststore. To do this, execute the following command:

```
escadm add-certificate --alias [ca cert alias] --file [file path]
```

3 Verify the certificate is added.

```
escadm list-certificate
```

### Removing a Root Certificate

The `escadm` tool has a 'delete-certificate' option which only takes --alias argument. The --alias argument refers to the name of the CA certificate to be deleted. Use this argument on the standalone/HA ESC VM:

**Step 1** On (master) ESC use `escadm` to delete certificate from ESC truststore.

```
escadm delete-certificate --alias [ca cert alias]
```

**Step 2** Verify the certificate is removed.

```
escadm list-certificate
```

### Managing Root Certificates During the Upgrade

- **Image Upgrade**: If you are backing up the ESC DB for upgrade, then no other action is required, the ESC truststore will be restored once the ESC DB is restored. If you are not backing up the ESC DB for upgrade, then each CA certificate needs to be added again to the ESC truststore.

- **RPM Upgrade**: This upgrade method keeps the ESC truststore as is, i.e. all CA certificates in the ESC truststore should remain there after upgrade.

### Using a Bootable Volume in ESC Installation

A volume in OpenStack is a detachable block storage device that can be attached to an ESC instance. You can store and also run ESC instances from a volume.

> **Note**

- Only one ESC instance can be launched from one volume at a time.

- ESC installation with a combination of bootable volume and high availability on cinder is not supported.
To launch an ESC instance from a bootable volume, do the following:

**Step 1** Create a bootable volume in OpenStack based on an ESC image or from a bootable volume. The bootable volume must be at least 30 GB disk size. For more information, see OpenStack documentation.

**Step 2** Deploy ESC VM using the `bootvm.py` command and choose the `--boot_volume` argument instead of the `--image` argument, as shown below:

```
./bootvm.py <esc_vm_name> --boot_volume <volume_name_or_id> --net <network> --user_pass <username>:<password> --user_confd_pass <username>:<password> --flavor <flavor_name>
```

**Note**
- Only one of these arguments, `--image` or `--boot_volume` must be passed to the `bootvm.py` command. The installation will fail, if both or none of the arguments are used.
- When launching an ESC instance from a bootable volume, volume disk size is considered over the flavor disk size.
- If an ESC instance is deleted, the volume attached to it will not be deleted, as the volume was created out-of-band.
Configuring High Availability

This chapter contains the following sections:

- High Availability Overview, page 19
- How High Availability Works, page 19
- Deploying ESC High Availability, page 20
- Configuring the Northbound Interface Access, page 23
- Important Notes, page 26
- Troubleshooting High Availability, page 27

High Availability Overview

ESC supports High Availability (HA) in the form of a Primary and Standby model. Two ESC instances are deployed in the network to prevent ESC failure and provide ESC service with minimum service interruption. If the primary ESC instance fails, the standby instance automatically takes over the ESC services. ESC HA resolves the following single point failures:

- Network failures
- Power failures
- Dead VM instance
- Scheduled downtime
- Hardware issues
- Internal application failures

How High Availability Works

ESC HA network can be either set up as a single installation of a ESC HA pair or deployed as two standalone ESC nodes that are converted into HA pair after re-configuring these nodes post deployment. A HA deployment consists of two ESC instances: a primary and a standby. Under normal circumstances, the primary ESC
instance provides the service. The corresponding standby instance is passive. The standby instance is in constant communication with the primary instance and monitors the primary instances' status. If the primary ESC instance fails, the standby instance automatically takes over the ESC services to provide ESC service with minimum interruption.

The standby also has a complete copy of the database of the primary, but it does not actively manage the network until the primary instance fails. The KeepAliveD service monitors both primary and standby instances activity status. When the primary instance fails, the standby takes over automatically. The standby instance takes over primary instance to manage the services while primary instance restoration is taking place.

When the failed instance is restored, if required you can manually initiate a switch-over and resume network management via the primary instance.

Both primary and standby ESC instances are connected to the northbound orchestration system through an IPv4 or IPv6 network. The router uses BGP or VIP routing to route Anycast IP to the primary ESC instance. For the northbound system, a unique virtual IP address is assigned to access the current primary ESC High Availability instance. The deployed VNFs are connected to both ESC primary and standby instances through another IPv6 network.

ESC HA nodes are managed by KeepAliveD and DRBD (Replication tool to keep the ESC database synchronized) sync network services. While the KeepAliveD service monitors both primary and standby instances status, the DRBD service monitors primary instance DB and sync the changes to the standby instance DB. These two services can be co-located on same VIP network or in two separate networks. VM handshake between ESC instances occurs through the KeepAliveD over the IPv4 or IPv6 network.

**Deploying ESC High Availability**

To deploy Cisco Elastic Services Controller (ESC) High Availability (HA), ESC standalone instances can be installed on two separate nodes - Primary and Standby. For more information see, *How High Availability Works*, on page 19. You can connect the Primary and Standby instances to either a Cinder volume or Replication based volume (DRBD).

The following deployment mechanisms can be used to deploy ESC HA:

- **Internal Storage**—When ESC HA is configured with Internal storage, the Primary and the Standby instances have individual databases which are always synchronized. In this solution, ESC HA is designed with database replication and DRBD is used as the tool for disk-level replication. The database in the Primary instance simultaneously propagates the data to the database in the Standby instance thus requiring no external storage. In the event of a Primary instance failing, the Standby instance get assigned the role of the Primary instance along with its own synchronized database.

  ESC HA is deployed using Internal storage, the ESC instances reply on the virtual IP address (that is kad_vip argument), and the interface of vrrp instance (that is kad_vif argument) to select the Primary ESC instance. To establish a reliable heartbeat network, it is recommended that the Primary and Standby ESC instances are on different physical hosts. The reliability of the physical links between the ESC instances (such as, network interface bonding) can also be taken into consideration.

- **Replicate External-Storages** — In this type of architecture, ESC HA is configured with DRBD and both Primary and Standby instance store their data in two external storages (OpenStack Cinder volumes). Each ESC node is attached by a Cinder volume and ESC data files are stored in the cinder volume. The data in two ESC node are synchronized through the database replication mechanism provided by DRBD.

The table lists the differences between the HA options:
<table>
<thead>
<tr>
<th></th>
<th>Internal Storage Based ESC HA</th>
<th>Replicate External Storage Based ESC HA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data sharing method</strong></td>
<td>Data replication between HA nodes</td>
<td>Data replication between two external storages (cinder volume)</td>
</tr>
<tr>
<td><strong>Installation Method</strong></td>
<td>Post-installation Configuration</td>
<td>Bootvm Installation</td>
</tr>
<tr>
<td><strong>VIM Support</strong></td>
<td>OpenStack, VMware, KVM</td>
<td>OpenStack only</td>
</tr>
<tr>
<td><strong>Dependency</strong></td>
<td>VIM independent</td>
<td>Rely on OpenStack cinder</td>
</tr>
</tbody>
</table>
| **Advantages**           | • No dependency on specific VIM components.  
• Flexible to build of HA clusters from commodity hardware, without the requirement for shared-storage. | • Use database replication mechanism for data synchronization  
• Two cinder volumes are used as external storage and are attached to ESC node. |
| **Limitations**          | The data consistency may be affected in a double fault condition (occurs when both ESC nodes have problems). | The data consistency may be affected in a double fault condition (occurs when both ESC nodes have problems). |

### Deploying ESC in High Availability Mode on Internal Storage

When you boot ESC instances on Primary and Standby instances, you need to specify the following `bootvm.py` command arguments to deploy ESC HA on an internal storage:

- `kad_vip`

**Note**

When ESC HA is deployed, the `kad_vip` argument allows end users to access the Primary ESC instance.

- `kad_vif`
- `ha_node_list`

These arguments enable the `bootvm.py` command to automatically set up the internal storage on the OpenStack. For more information on using the `bootvm.py` command arguments, see Appendix A: Cisco Elastic Services Controller Installer Arguments.

To deploy ESC HA instances, use the bootvm script on both the nodes with the following arguments:

**ON HA NODE 1:**
Deploying ESC High Availability

Replicate external storage ESC HA requires two cinder volumes for database storage.

Before You Begin

- Networks and IP addresses that both ESC instances will connect to
- Keepalive interface and virtual IP for HA switchover

**Step 1**
Create two cinder volumes in OpenStack. The configured cinder volume size should be 3GB.

```
$ cinder create --display-name cindervolume_name_a[SIZE]
$ cinder create --display-name cindervolume_name_b[SIZE]
```

**Step 2**
Check the status of the created cinder volume and find the uuids for deployment.

```
$ cinder list
```
Step 3  Deploy ESC HA instances. Use the `bootvm` script on both the nodes with the following arguments:

ON HA NODE 1:

```
$ ./bootvm.py <ESC_HA_Node1>
  --user_pass <username>:<password>
  --user_confd_pass <username>:<password>
  --gateway_ip <default gateway IP address>
  --net <network name1>
  --ipaddr <static ip address>
  --image <image_name>
  --avail_zone nova:<openstack zone>
  --kad_vip <virtual IP address>
  --kad_vif <VRRP_Interface_Instance>
  --ha_node_list=<ESC_HA_NODE1_IP> <ESC_HA_NODE2_IP>
  --db_volume_id <cinder volume id>
  --ha_mode drbd_on_cinder
```

ON HA NODE 2:

```
$ ./bootvm.py <ESC_HA_Node2>
  --user_pass <username>:<password>
  --user_confd_pass <username>:<password>
  --gateway_ip <default gateway IP address>
  --net <network name1>
  --ipaddr <static ip address>
  --image <image_name>
  --avail_zone nova:<openstack zone>
  --kad_vip <virtual IP address>
  --kad_vif <VRRP_Interface_Instance>
  --ha_node_list=<ESC_HA_NODE1_IP> <ESC_HA_NODE2_IP>
  --db_volume_id <cinder volume id>
  --ha_mode drbd_on_cinder
```

Step 4  After both VMs are rebooted; the keepalived state on one of ESC VM should be one of ESC VM should be in MASTER state and the other one should be in BACKUP state. You can check ESC HA state by using following command: $

```
escadm status --v.
```

Configuring the Northbound Interface Access

When you configure ESC HA, you can also specify a virtual or BGP Anycast IP address to the HA pair. The northbound interface as well as the service portal uses virtual or BGP Anycast IP address to access the ESC Primary HA instance. When deploying ESC HA, use the following arguments with the `./bootvm.py` script:

- `--ha_node_list`
- `--kad_vip`
- `--kad_vif`
For more details on these arguments, see section Appendix A: Cisco Elastic Services Controller Installer Arguments.

The following section explains how to configure ESC HA with multiple interfaces and to configure the virtual and BGP Anycast IP address.

**Configuring ESC HA with Multiple Interfaces**

You can configure ESC HA with DRDB synchronization and VRRP heartbeat broadcasting on a network interface for data synchronization and VNF monitoring. You can use an additional network interface to allocate Virtual IP for the northbound access. To configure the multiple interfaces on ESC HA nodes, use --ha_node_list, --kad_vip, --kad_vif arguments to specify these multiple network interfaces configuration. For details on these arguments, see section Appendix A: Cisco Elastic Services Controller Installer Arguments.

Example configuration steps are shown below:

```bash
./bootvm.py <esc_ha1> --user_pass <username>:<password> --user_confd_pass <username>:<password> --image <image_id> --net <net-name> --gateway_ip <default_gateway_ip_address> --ipaddr <ip_address1> <ip_address2> --ha_node_list < IP addresses HA nodes1> < IP addresses for HA nodes2> --kad_vip <keepaliveD VIP of the HA nodes and the interface for keepaliveD VIP> (for example: --kad_vip 192.0.2.254:eth2) --kad_vri <virtual router id of vrrp instance> --kad_vif <virtual IP of the HA nodes or the interface of the keepalived VRRP> (for example: --kad_vif eth1 --ha_mode drbd --route 10.85.103.0/24:192.168.0.1:eth1 --avail_zone nova:zonename
```

Similarly, a three network interface can be configured for ESC HA nodes. An example three interfaces configuration is shown below with the following assumptions:

- Network 1 is an IPv6 network used for northbound connection. ESC VIP is allocated in this network and the Orchestrator send requests to ESC through ESC VIP.
- Network 2 is an IPv4 network used for ESC sync traffic (DRDB synchronization) and VRRP heartbeat. This network is also used for OpenStack connection and VNF monitoring.
- Network 3 is another IPv4 network used for management. The SA, rsyslog, etc. can use this network to manage ESC.

```bash
./bootvm.py esc-ha-0 --image ESC-2_2_x_yyy --net esc-v6 esc-net --gateway_ip 192.168.0.1 --ipaddr 2001:cc0:2020::fa 192.168.0.239 192.168.5.239 --ha_node_list 192.168.0.239 192.168.0.243 --kad_vip 192.0.2.254:eth0 --kad_vif eth1 --ha_mode drbd --route 10.85.103.0/24:192.168.0.1:eth1 --avail_zone nova:zonename
```

**Configuring the ESC HA Virtual IP Address**

In this option, the value of kad_vip argument should be a virtual IP, which allows the service portal and the northbound to access the Primary ESC and send requests to ESC HA service through virtual IP (VIP).
If northbound and both ESC HA nodes are located in the same network, you can connect directly through the virtual IP (VIP). If northbound doesn't sit on the same network as ESC HA, assign a floating IP to ESC HA VIP using the procedure below:

1. Create a port with the VIP address (kad_vip) in the same network as ESC's kad_vip connects.
   
   ```bash
   neutron port-create esc-net --name esc_vip --fixed-ip subnet_id=esc-subnet,ip_address=192.168.0.87
   ```

2. Deploy ESC HA. See Configuring High-Availability section in Installing ESC on OpenStack.

   Note
   
   Make sure the kad_vip using the same IP address as the port created above.

3. Associate a floating IP with the port created above. The first uuid is the floating ip id and the second one is the port id.

   ```bash
   neutron floatingip-associate <floating IP> <port ID>
   ```

   Access ESC HA through the floating IP and it will connect to the ESC Primary node.

4. For the portal access, make sure the keepalive network is accessible by your browser and the virtual IP is the IP address to access the portal of the Primary node.

   For example, if the VIP is 192.0.2.254, access ESC HA portal with https://192.0.2.254:9001/.

### Configuring the ESC HA BGP Anycast IP Address

In this option, ESC HA uses BGP advertisement to provide unique IP access for the northbound to connect to the ESC HA.

To configure BGP for ESC HA, the following network parameters are required:

- BGP remote IP
- IP of the interface for BGP anycast routing
- BGP local AS number for routing configuration
- BGP remote AS number for routing configuration
- BGP routing configuration

To configure BGP for ESC HA, use the escadm tool in the ESC Virtual Machine, as shown below:

```bash
$ sudo bash
$ escadm bgp set --local_ip LOCAL_IP --anycast_ip ANYCAST_IP --remote_ip REMOTE_IP
   --local_as LOCAL_AS --remote_as REMOTE_AS
   --local_router_id LOCAL_ROUTER_ID --remote_ip REMOTE_IP
$ sudo escadm restart
```
You must configure the BGP router, if BGP Anycast IP is configured for the northbound access.

**Configuring BGP Router**

**Step 1**  
Log on to the BGP router.

**Step 2**  
Configure BGP Anycast IP command:

- For IPv6, run the following commands:

```plaintext
configure
router bgp <Router_AS_#>
neighbor <Esc_ip_address>
remote-as <ESC_AS_#>
address-family ipv6 unicast
route-policy anycast-in in
route-policy anycast-out out
route-policy anycast-in
pass
end-policy
route-policy anycast-out
drop
end-policy
commit
```

- For IPv4, run the following commands:

```plaintext
configure
router bgp <Router_AS_#>
neighbor <Esc_ip_address>
remote-as <ESC_AS_#>
route-policy anycast-in in
route-policy anycast-out out
route-policy anycast-in
pass
end-policy
route-policy anycast-out
drop
end-policy
commit
```

**Step 3**  
Ping the Anycast IP or SSH to the Anycast IP to check whether the configuration has been successfully associated with the ESC Master instance.

**Important Notes**

- **ESC HA**
  
  * An HA failover takes about 2 to 5 minutes. The ESC service will not be available during the switchover time.
• When the switchover is triggered during transactions, all incomplete transactions will be dropped. The requests should be re-sent by northbound if it does not receive any response from ESC.

• **External Storage**

  • If the Primary ESC instance is suspended by OpenStack command, the switch over will be triggered but the cinder volume won't be attached to the new Primary ESC instance. This is not a valid use case for ESC HA.

• **Internal Storage**

  • Two ESC instances have to be deployed to establish the HA solution. The ESC HA will start to work when both ESC instances are successfully deployed and are able to connect to each other. If you just deploy one ESC instance with HA parameters, the ESC instance keeps Switching-to-Master state and will not be able to provide any service until it reaches its peer.

  • Split-brain scenario can still happen in this ESC HA solution, although the chance is very low.

---

**Troubleshooting High Availability**

• Check for network failures. If a network problem occurs, you must check the following details:

  • The IP address assigned is correct, and is based on the OpenStack configuration.

  • The gateway for each network interface must be pingable.

  • The BGP configuration between the installation arguments and ASR configuration must match.

• Check the logs for troubleshooting:

  • The ESC Admin logs at `/var/log/esc/escadm.log`

  • The ESC manager log at `/var/log/esc/escmanager.log`

  • The ESC HA log at `/var/log/esc/esc_haagent.log`

  • The exabgp log at `/var/log/exabgp.log`

  • The KeepAliveD log at `/var/log/messages` by grep keepalived

• If BGP is used, check the BGP configuration settings in the following file location:

  `/opt/cisco/esc/esc-scripts/bgp-sa/exabgp/neighbor-init.conf`

• Check for DRBD (Replication based ESC HA) for Internal Storage solution:

  • Check the DRBD configuration file at

    `/etc/drbd.d/esc.res`

  • Access the DRBD log

    `/var/log/messages` | grep drbd
PART II

Installing Cisco Elastic Services Controller on VMware vCenter

- Prerequisites, page 31
- Installing Cisco Elastic Services Controller on VMware vCenter, page 35
- Configuring High Availability, page 43
Prerequisites

The following sections detail the prerequisites for installing Cisco Elastic Services Controller:

- Virtual Resource and Hypervisor Requirements, page 31
- vCenter Resources, page 32
- Important Notes, page 32

Virtual Resource and Hypervisor Requirements

The following table lists the prerequisites to install Cisco Elastic Services Controller on VMware vCenter or vSphere:

See the VMware Compatibility Guide to confirm that VMware supports your hardware platform.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Requirements</strong></td>
<td></td>
</tr>
<tr>
<td>Virtual CPUs</td>
<td>4 VCPUs</td>
</tr>
<tr>
<td>Memory</td>
<td>8 GB RAM</td>
</tr>
<tr>
<td>Disk Space</td>
<td>30 GB</td>
</tr>
<tr>
<td><strong>Hypervisor Requirements</strong></td>
<td></td>
</tr>
<tr>
<td>VMWare vSphere</td>
<td>5.5 or 6.0 with VMware ESXi (English only)</td>
</tr>
<tr>
<td>VMWare vCenter</td>
<td>ESC 2.1 and later supports both vCenter versions 5.5 or 6.0 (English only). ESC 2.0 supports only vCenter version 5.5.</td>
</tr>
</tbody>
</table>
vCenter Resources

Resources to be created/installed on vCenter:

- **Datacenters**: At least one datacenter. For more details, see the **Important Notes** below.

- **Hosts**: Host configuration based on your targeted performance objectives. Each Host under the single vDS must have at least two physical Network Interface Card (NIC) connected, (one for vCenter Management Interface by default, and the other used to assign to VDS's uplink portgroup). This setup is required for data access across hosts.

- **Compute Clusters**: Clusters can be created to group several hosts together.

- **Datastores**: Shared datastore is required if user wants to leverage DRS.

- **Distributed Switches**: At least one distributed switch that will contains all the VNF supporting networks.

**Important Notes**

Keep in mind the following important notes while installing ESC on a VMware:

- A single ESC instance will only supports:
  - Multiple Datacenter supported deployment, network, image, subnet creation
  - One vSphere Distributed Switch (VDS)

- DPM, HA, and vMotion must be off.

- If DRS is enabled, it has to be in the "Manual Mode".

- Fault Tolerance is not supported.

- Datastore Cluster is not supported, only flat datastore(s) structure under the cluster or under the datacenter are supported.

- ESC only supports a default resource pool. Adding and creating resource pools are not supported.

- Image (Template) created through ESC are stored under `/esc-ovas` folder.

- Day-0, smart license, and other supported files are packed into a ISO file, and uploaded to the same folder where the VM rest, then mount it as a CD-ROM to the VM.

- ESC/VIM does not respond for the name and file content passed in for generating ISO file. They have to be provided according to each template's requirements. e.g. for ASAv, the day-0 config has to be named as "day0-config", and smart license token has to be named as "idtoken".

- When you see the error message "Networking Configuration Operation Is Rolled Back and a Host Is Disconnected from vCenter Server", it is due to a vCenter's limitation. See the [Troubleshooting guide](#), page 91 to increase the timeout for rollback.

- The following VM features and operations are not supported in all versions of the Cisco CSR 1000V. If still these operations are used or performed, there may be risk of encountering dropped packets, dropped connections, and other error statistics.

  1. DRS
  2. Suspend
• Although deployments can be processed without Shared Storage, ESC does not guarantee optimized computing resource. Shared storage(s) should associate with as many as possible hosts, which will give more opportunity to DRS to balance resources.

• Every time a re-deploy happens as part of recovery on VMware, VM’s interface(s) will have different MAC addresses.

• All the VM group defined in a datamodel must accompany with a "zone-host" placement policy, meaning the deployment has to be either host-targeted or cluster-targeted.

• Recovery may fail, if a VM has PCI/PCIe passthrough device(s) attached, when it's recovered to a computing-host (picked based on ESC placement algorithm) which doesn't have any PCI/PCIe passthrough enabled device available.

• For PCI/PCIe passthrough working, DRS has to be off.

• If you experience a PowerOn error on a VM that has PCI/PCIe passthrough device(s) attached to it, update the VM or the image (template) the VM is cloned from, using the solution described here.
CHAPTER 7

Installing Cisco Elastic Services Controller on VMware vCenter

This chapter describes how to install Cisco Elastic Services Controller on VMware vCenter and includes the following sections:

- Installing Cisco Elastic Services Controller on VMware vCenter, page 35
- Next Steps: Cisco Elastic Services Controller Virtual Machine, page 41

Installing Cisco Elastic Services Controller on VMware vCenter

Cisco Elastic Services Controller can be installed in a VMware ESXi hypervisor and can be accessed or managed using vSphere client of VMware. You can install Cisco Elastic Services Controller in a VMware environment using an Open Virtual Appliance (OVA) package.

The VMware vSphere client can be connected directly to your ESXi installation, or it can be connected to a vCenter server which in turn is connected to your vSphere installation. Connecting through vCenter provides a number of capabilities that connecting directly to ESXi does not. If a vCenter server is available and associated with the ESXi installation, it should be used.

Preparing to Install Cisco Elastic Services Controller

In order to install Cisco Elastic Services Controller and configure its network connection, you have to answer several questions. Some of these questions concern the networking environment in which the virtual machine is being installed, and some of them concern values which are unique to the particular virtual machine being installed.

Before you perform the installation, ensure that you are prepared by reviewing this checklist:

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Your Information/ Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVA image location</td>
<td></td>
</tr>
<tr>
<td>OVA image</td>
<td></td>
</tr>
<tr>
<td>VSphere Web Client</td>
<td></td>
</tr>
</tbody>
</table>
Installing the Elastic Services Controller Using the OVA Image

To install Cisco Elastic Services Controller, you must first download the correct installation file. Using vSphere, connect directly to the ESXi installation or the vCenter server, and select the ESXi installation where the OVA is to be deployed.

This procedure describes how to deploy the Elastic Services Controller OVA image on VMware.

Before You Begin

• Set your keyboard to United States English.
• Confirm that the Elastic Services Controller OVA image is available from the VMware vSphere Client.
• Make sure that all system requirements are met as specified in the Chapter 6: Prerequisites.
• Gather the information identified in Preparing to Install Cisco Elastic Services Controller.

Step 1
Using the VMware vSphere Client, log in to the vCenter server.

Step 2
Choose vCenterHome > Hosts and Clusters. Right click the host where you want to deploy ESC, and then choose Deploy OVF Template.

Step 3
In the wizard, provide the information as described in the following table:

<table>
<thead>
<tr>
<th>Screen</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select source</td>
<td>Select the Elastic Services Controller OVA.</td>
</tr>
<tr>
<td>Screen</td>
<td>Action</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Review details</td>
<td>Review OVF Template details.</td>
</tr>
<tr>
<td>Select name and folder</td>
<td>Enter a name and select a folder for the VM.</td>
</tr>
<tr>
<td>Select configuration</td>
<td>Select any one of the following deployment configurations:</td>
</tr>
<tr>
<td></td>
<td>• Large, 1 Network</td>
</tr>
<tr>
<td></td>
<td>• Large, 2 Networks</td>
</tr>
<tr>
<td></td>
<td>• Large, 3 Networks</td>
</tr>
<tr>
<td>Select a resource</td>
<td>Select the host or cluster to run the ESC template.</td>
</tr>
<tr>
<td>Select storage</td>
<td>Select a location to store the files for the VM and a provision type.</td>
</tr>
<tr>
<td></td>
<td>The storage can be local or shared remote, such as NFS or SAN.</td>
</tr>
<tr>
<td></td>
<td>You can choose either Thin provisioned format or Thick provisioned format to store the VM virtual disks.</td>
</tr>
<tr>
<td>Select networks</td>
<td>Based on the Network configuration for the deployment selected in the Select configuration option, you can allocate the pre-configured networks in vCenter to ESC network interfaces.</td>
</tr>
<tr>
<td>Customize template</td>
<td>Bootstrap Properties</td>
</tr>
<tr>
<td>Username</td>
<td>Administrator username for remote login.</td>
</tr>
<tr>
<td>Password</td>
<td>Administrator password.</td>
</tr>
<tr>
<td>Host name</td>
<td>VM Hostname.</td>
</tr>
<tr>
<td>Network IP</td>
<td>VM IP address.</td>
</tr>
<tr>
<td>Network Gateway</td>
<td>Gateway IP address.</td>
</tr>
<tr>
<td>Enable Https Rest</td>
<td>Enable external REST interface over HTTPS on port 8443.</td>
</tr>
<tr>
<td>Enable Portal startup</td>
<td>Enable Portal startup at port 9001 (for https).</td>
</tr>
<tr>
<td>VIM Settings of vCenter Server</td>
<td>IP address of vCenter server for VNF deployment .</td>
</tr>
<tr>
<td>vCenter IP</td>
<td>Port of the vCenter server.</td>
</tr>
</tbody>
</table>
### Screen | Action
--- | ---
vCenter Username | Username to access vCenter server.
vCenter Password | Password to access vCenter server.
Datacenter Name | Name of the Datacenter in target vCenter for VNF deployment (Default VDC after Multi-VDC supported)
Datastore Name | The destination datastore of all the image (template) create through ESC.
Datastore Host | The destination computing-host of all the image (template) create through ESC.
Ready to Complete | Review the deployment settings.
Caution | Any discrepancies can cause VM booting issues. Carefully review the IP address, subnet mask, and gateway information for accuracy.
Public Key | Administrator authorized public key for remote login.
ConfD Username | Administrator username for netconf and ConfD CLI.
ConfD Password | Administrator password for netconf and ConfD CLI.
ConfD Public Key | Administrator authorized public key for netconf and ConfD CLI.

**Step 4** Check the **Power on after deployment** check box to power on the VM after deployment.

**Step 5** Click **Finish**.
A progress indicator shows the task progress until Elastic Services Controller is deployed.

**Step 6** After Elastic Services Controller is successfully deployed, click **Close**.

**Step 7** Power on the Elastic Services Controller VM.

---

## Installing Elastic Services Controller Using OVF Tool

In addition to installing the Elastic Services Controller using the OVA image, you can use the VMware OVF Tool, a command-line client, to install Elastic Services Controller on VMware vCenter or vSphere.

To install Elastic Services Controller (ESC) from the command line, do the following:

**Step 1** Use the probe mode to learn the properties of the OVA package. The probe mode allows you to investigate the contents of a source.
To invoke the probe mode, use the ovftool command with only a source and no target.

>ovftool <source locator>
The following example shows the result of probing the ESC OVA.

```
ESC_OVA=(Path to the OVA Package)

ESC_HOSTNAME="$USER"
ESC_GATEWAY="192.0.2.1"
ESC_NET1_IP="192.0.2.0/24" #
ESC_NET2_IP="192.51.100.0/24"
ADMIN_USERNAME="admin"
ADMIN_PASSWORD="password"
HTTPS_REST="True"

VMWARE_VCENTER_PORT='80'
VMWARE_VCENTER_IP='192.0.2.0.xx'
VMWARE_DATASTORE_HOST='192.0.2.0.xx'
VMWARE_DATACENTER_NAME='OTT-ESC-1'
VMWARE_DATASTORE_NAME='cluster-datastore1'
VMWARE_COMPUTE_CLUSTER_NAME='OTT-CLUSTER-1'
VMWARE_VCENTER_USERNAME='root'
VMWARE_VCENTER_PASSWORD='password'
VMWARE_VCENTER_FOLDER="$USER"

# All valid deployment options:
# 4CPU-8GB  (default)
# 4CPU-8GB-2Net
# 4CPU-8GB-3Net
DEPLOYMENT_OPTION="4CPU-8GB-2Net"
```

**Step 2**

Before you deploy the ESC OVA, configure the properties of the OVA packages. Ensure the following OVA package properties are updated for the ESC OVA: ESC_OVA, ESC_HOSTNAME, VMWARE_VCENTER_FOLDER, ESC_NET1_IP, ESC_NET2_IP, and VMWARE_VCENTER_FOLDER.

The OVA descriptors contain configuration properties for the OVA package. You can set only one property at a time, but you can have multiple instances of the option per command. For multiple property mappings, repeat the option, separating them with a blank, for example --prop:p1=v1 --prop:p2=v2 --prop:p3=v3.

```
>ovftool/ovftool
--powerOn
--acceptAllEulas
--noSSLVerify
--datastore=$VMWARE_DATASTORE_NAME
--diskMode=thin
--name=$ESC_HOSTNAME
--deploymentOption=$DEPLOYMENT_OPTION
--vmFolder=$VMWARE_VCENTER_FOLDER
--prop:admin_username=$ADMIN_USERNAME --prop:admin_password=$ADMIN_PASSWORD
--prop:admin_username=admin
--prop:admin_password='Strong4Security!' 
--prop:confd_admin_username=admin 
--prop:confd_admin_password='Strong4Security!' 
--prop:esc_hostname=$ESC_HOSTNAME 
--prop:vmware_vcenter_port=$VMWARE_VCENTER_PORT 
--prop:vmware_vcenter_ip=$VMWARE_VCENTER_IP 
```
Following are some advanced examples of passing user credentials through properties.

Advanced usage with password hash:

```bash
--prop:admin_username=admin
--prop:admin_password='$6$wnOi$UDOQmkKm2tQtr2jDVNhoo4wS42ffYYmz8o8GLDugfzTBbJXmMQDw14Gx2pxQvMumea125.ag9HEZUq8L.AdM2v0'
--prop:confd_admin_username=admin
--prop:confd_admin_password='$6$wnOi$UDOQmkKm2tQtr2jDVNhoo4wS42ffYYmz8o8GLDugfzTBbJXmMQDw14Gx2pxQvMumea125.ag9HEZUq8L.AdM2v0'
--prop:confd_admin_public_key='ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAu+nkTtu2pShVbTYL+mmKxtmz2M5dNXFy8IeX/JHSfXsODH1EAYs1zHGFxq36RT5vIG/
+c2u/V8rSa7xzXDrdfGICxfkPuEj1UQH2Mq2yFjMfcaSAT56hsqE= admin@esc'
--prop:confd_admin_username=admin
--prop:confd_admin_password='$6$wnOi$UDOQmkKm2tQtr2jDVNhoo4wS42ffYYmz8o8GLDugfzTBbJXmMQDw14Gx2pxQvMumea125.ag9HEZUq8L.AdM2v0'
--prop:confd_admin_public_key='ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAu+nkTtu2pShVbTYL+mmKxtmz2M5dNXFy8IeX/JHSfXsODH1EAYs1zHGFxq36RT5vIG/
+c2u/V8rSa7xzXDrdfGICxfkPuEj1UQH2Mq2yFjMfcaSAT56hsqE= admin@esc'
```

**Step 3**

To deploy the OVA package with the VMware OVF Tool, use the following command syntax:

```bash
>ovftool <source locator> <target locator>
```

where `<source locator>` is the path to the OVA package and `<target locator>` is the path target for the virtual machine, OVA package or VI. A VI location refers to any location on a VMware product, such as vSphere, VMware Server or ESXi. For more information on the VMware OVF Tool, see the VMware OVF Tool user documentation.

The ESC VM is deployed on VMware and powered on automatically.
Powering on Cisco Elastic Services Controller Virtual Machine

To power on the Cisco Elastic Services Controller virtual machine (VM):

You must set the memory and CPUs based on the requirements prior to clicking the power on. Once you start the VM you cannot change the memory or CPU settings until you shut down.

**Step 1**
After deploying the VM, select the virtual machine name in vSphere, right-click on it and select **Open Console**.

**Step 2**
Click the **Power on** button ( ). During the initial boot of the newly deployed machine, you will be prompted to enter a root (system) password, which is not the Cisco Elastic Services Controller portal password.

**Note**
This is the root password for the underlying Linux operating system on which the Cisco Elastic Services Controller portal is installed. You will be asked to enter this password twice. You will need root access to the underlying Linux operating system at various times in the future, so make sure that you remember this password.

The boot process can take a while.

The End User License Agreement window appears on the first boot. Read the license agreement in its entirety, and only if you understand and accept the license terms, enter y (Yes).

Next Steps: Cisco Elastic Services Controller Virtual Machine

**Logging in to Cisco Elastic Services Controller Portal**

To log in to the ESC Portal, see the **Logging in to the ESC Portal**, on page 68

**Configuring the Virtual Machine to Automatically Power Up**

You can configure the ESXi hypervisor to automatically power up the ESC VM when power is restored to the ESXi hypervisor layer.
You must manually power up the VM.

---

**Step 1**
In the vSphere client, select the ESXi machine to which you are connected. It is not a specific VM that you have to select but the ESXi hypervisor on which they reside.

**Step 2**
Select the **Configuration** tab.

**Step 3**
Click the **Virtual Machine Startup/Shutdown** link under the **Software** area. You should see the VM in the list shown in window.

**Step 4**
Click the **Properties...** link present at the top right corner of the page. If you do not see that, resize the window until you do.
The Virtual Machine Startup and Shutdown page is displayed.

**Step 5**
Check the **Allow virtual machines to start and stop automatically with the system** check box.

**Step 6**
Select the virtual machine running ESC and use the **Move Up** button on the right to move it up into the group labeled **Automatic Startup**

**Step 7**
Click **OK**
This ensures that whenever power is restored to the ESXi hypervisor, the ESC VM powers up automatically.
Configuring High Availability

This chapter contains the following sections:

- High Availability Overview, page 43
- How High Availability Works, page 43
- Deploying ESC High Availability, page 44
- Important Notes for ESC HA, page 45
- Troubleshooting High Availability, page 45

High Availability Overview

ESC supports High Availability (HA) in the form of a Primary and Standby model. Two ESC instances are deployed in the network to prevent ESC failure and provide ESC service with minimum service interruption. If the primary ESC instance fails, the standby instance automatically takes over the ESC services. ESC HA resolves the following single point failures:

- Network failures
- Power failures
- Dead VM instance
- Scheduled downtime
- Hardware issues
- Internal application failures

How High Availability Works

A High Availability deployment consists of two ESC instances: a primary and a standby. Under normal circumstances, the primary ESC instance provides the services. The corresponding standby instance is passive. The standby instance is in constant communication with the primary instance and monitors the primary instances' status. If the primary ESC instance fails, the standby instance automatically takes over the ESC services to provide ESC service with minimum interruption.
The standby also has a complete copy of the database on the primary, but it does not actively manage the network until the primary instance fails. When the primary instance fails, the standby takes over automatically. Standby instance takes over primary instance to manage the services while primary instance restoration taken place.

When the failed instance is restored, failback operations can be initiated to resume network management via the original primary instance.

ESC instances are managed by using KeepAliveD. The VM handshake between ESC instances occurs through the KeepAliveD over the IPv4 network.

### Deploying ESC High Availability

To deploy ESC HA on VMware vCenter or vSphere, two separate standalone nodes need to be installed first. After the standalone ESC instances are installed, reconfigure these nodes to turn them into Primary and Standby using the following:

- `kad_vip`
- `kad_vif`
- `ha_node_list`

**Note**

- On each ESC VM, we need to run `escadm` tool to configure ESC HA parameters and then reboot the VM.
- When you are deploying ESC HA, the `kad_vip` argument allows end users to access the Primary ESC instance.

### Before You Begin

- Cisco Elastic Services Controller (ESC) High Availability (HA) requires a network to keep alive and replicate database between primary and standby nodes. Both ESC VMs must have at least one network interface connecting to the same network and must be able to communicate to each other through the network.
- Ensure the two ESC VMs are located in different hosts and datastores so that single point failures can be prevented.

### Step 1

Log in to the ESC Standalone instances.

### Step 2

As an admin user, run the `escadm` tool on both the Primary and Standby instances and provide the corresponding arguments.

- `kad_vip`— Specifies the IP address for Keepalived VIP (virtual IP) plus the interface of Keepalived VIP [ESC-HA]
- `kad_vif`— Specifies the interface for Keepalived virtual IP and keepalived VRRP [ESC-HA]. You can also use this argument to only specify the interface for keepalived VRRP, if the VIP interface is already specified using the `kad_vip` argument.
• **ha_node_list**— Specifies list of IP addresses for HA nodes in the Primary/Standby cluster for DRDB synchronization. This argument is utilized for replication-based HA solution only. For ESC instances with multiple network interfaces, the IP addresses should be within the network that --kad_vif argument specifies.

```
$ escadm ha set --kad_vip= <ESC_HA_VIP> --kad_vif= <ESC_KEEPALIVE_IF> --ha_node_list= <ESC_NODE_1_IP> <ESC_NODE_2_IP>
$ escadm reload
```

**Step 3**  
After the restart, one ESC VM should be in Primary state and the other one should be in Standby state.

**Step 4**  
Add the VIP to the allowed address pairs for both VMs so that the VIP is reachable from outside.

**Step 5**  
Verify the status of each ESC instance.

```
# escadm status
```

The following table lists few other command to check the status:

<table>
<thead>
<tr>
<th>Status</th>
<th>CLI Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC HA Role</td>
<td>cat /opt/cisco/esc/keepalived_state</td>
</tr>
<tr>
<td>ESC Health</td>
<td>health.sh</td>
</tr>
</tbody>
</table>
| ESC Service Status      | If you want to see more details (such as status of the VIM manager, SNMP, portal, ESC manager, keepalived status and so on), add '–v':  
  escadm status --v  
  To check the detailed status, check the /var/log/esc/escadm.log |

---

**Important Notes for ESC HA**

- The HA failover takes about 2 to 5 minutes based on the number of managed VNFs to be operational. ESC service will not be available during the switchover time.

- When the switchover is triggered during transactions, all incomplete transactions will be dropped. The requests should be re-sent by Northbound interface if it does not receive any response from ESC.

**Troubleshooting High Availability**

- Check for network failures. If a network problem occurs, you must check the following details:
  - The IP address assigned is correct, and is based on the OpenStack configuration.
  - The gateway for each network interface must be pinged.
• The BGP configuration between the installation arguments and ASR configuration must match.

• Check the logs for troubleshooting:
  - The ESC manager log at /var/log/esc/escmanager.log
  - The ESC HA log at /var/log/esc/vsoc_haagent.log
  - The exabgp log at /var/log/exabgp.log
  - The KeepAliveD log at /var/log/messages by grep keepalived
  - The ESC service status log at /var/log/esc/escadm.log
PART III

Installing Cisco Elastic Services Controller on a Kernel-based Virtual Machine (KVM)

- Installing Cisco Elastic Services Controller on a Kernel-based Virtual Machine, page 49
This chapter describes how to install Cisco Elastic Services Controller on a Kernel-based Virtual Machine and includes the following sections:

- Installing Cisco Elastic Services Controller in a Kernel-based Virtual Machine, page 49
- Next Steps: Cisco Elastic Services Controller Kernel-based Virtual Machine, page 51

### Installing Cisco Elastic Services Controller in a Kernel-based Virtual Machine

Cisco Elastic Services Controller can be installed in a Kernel-based Virtual Machine. You can install Cisco Elastic services controller in a Kernel-based Virtual Machine using libvirt.

### Preparing to Install Cisco Elastic Services Controller on a Kernel-based Virtual Machine

If you plan to run Cisco Elastic Services Controller on a kernel-based virtual machine, make sure the following are setup:

<table>
<thead>
<tr>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python 2.7 or later</td>
</tr>
<tr>
<td>Installed by default on Linux</td>
</tr>
<tr>
<td>python-setuptools</td>
</tr>
<tr>
<td>Installed by default on Linux</td>
</tr>
</tbody>
</table>
On RHEL:
# easy_install pip
Since the installation using pip compiles source files, the gcc and python development packages are also required on RHEL. To install these packages on RHEL:
# yum install gcc python-devel

On Ubuntu: Installed by default. Since the installation using pip compiles source files, the gcc and python development packages are also required on Ubuntu. To install these packages on Ubuntu:
# apt-get install python-dev

# pip install python-keystoneclient
# pip install python-cinderclient
# pip install python-novaclient
# pip install python-neutronclient

OpenStack clients

# pip install python-keystoneclient
# pip install python-cinderclient
# pip install python-novaclient
# pip install python-neutronclient

On RHEL:
# yum install genisoimage

On Ubuntu:
# apt-get install genisoimage

libvirt and virtinst

On RHEL 6.x:
# yum install libvirt-python python-virtinst

On RHEL 7.x:
# yum install libvirt-python virt-install

On Ubuntu:
# apt-get install libvirt-dev
# pip install libvirt-python

libvirt will create the default network automatically.

Installing Elastic Services Controller on a Kernel-Based Virtual Machine

To install standalone Elastic Services Controller (ESC) on a kernel-based virtual machine, do the following:

Step 1
Load the variables from the openrc file that contains OpenStack credentials:
cat ./openrc.sh
export OS_TENANT_NAME='<OS tenant username>'
export OS_USERNAME='<OS username>'
export OS_PASSWORD='<OS password>'
export OS_AUTH_URL='http://<Openstack Host>:5000/v2.0/

source ./openrc.sh
Step 2 Copy the ESC qcow2 image and the bootvm.py into the kernel-based VM.

Step 3 Boot ESC on a kernel-based VM on the default network that was created when libvirt was installed, use one of the following command:

```
./bootvm.py --user_pass <username>:<password> --user_confd_pass <username>:<password> --libvirt --image <image_name> esc-vm --net <default network>
```

Step 4 Boot ESC on a kernel-based VM on the default network with static IP, using the following command:

```
./bootvm.py --user_pass <username>:<password> --user_confd_pass <username>:<password> --libvirt --image <image_name> esc-vm --net <network> --ipaddr <ip_address>
```

Step 5 Get a list of used IP addresses in your network. Use IP addresses that are not in the list for both HA bootvm.py command and for kad_vip. Determine the first 3 octets of your network (i.e. 192.168.122) and pass it in the below command:

```
ar -an | grep 192.168.122
```

Step 6 To install ESC on a kernel-based VM in high availability, use the following command twice for both the HA nodes:

**Note** For the second bootvm.py command, use the other HA instance name.

```
./bootvm.py --user_pass <username>:<password> --user_confd_pass <username>:<password> --libvirt --image <image_name> --ha_mode drbd --gateway_ip <default_gateway_ip_address> --ipaddr <ip_address> --ha_node_list <ha peer ip addresses separated by comma> --kad_vip <vip address> esc-ha-1 --net <network>
```

---

Next Steps: Cisco Elastic Services Controller Kernel-based Virtual Machine

**Logging in to Cisco Elastic Services Controller Portal**

To log in to the ESC Portal, see the [Logging in to the ESC Portal](#) on page 68

**Verifying ESC installation for a Kernel-based Virtual Machine (KVM)**

After deploying ESC on a Kernel-based virtual machine, use the following procedure to verify the deployment.

---

**Step 1** Check that the ESC VMs have booted using the following command:

```
$ virsh list
```

**Step 2** Get the IP address of the ESC VM, using the following command:

```
$ arp -an | grep <ip_address>
```

**Step 3** Connect to ESC using SSH and verify the processes are running:

```
$ ssh USERNAME@ESC_IP
```
Troubleshooting Tips

When SSH access is not available, due to network conditions or ESC startup failures, you can connect to ESC through console (if enabled in ESC VM image) or VNC access. To access ESC VM through VNC, do the following:

1. Identify the vnc port.
   ```bash
touch dumpxml 10 | fgrep vnc
```
2. Create a ssh tunnel to the local vnc port to allow connection from your remote VNC client.
Post Installation Tasks
Post Installation Tasks

This chapter contains the following sections:

- Changing the ESC Password, page 55
- Configuring Pluggable Authentication Module (PAM) Support for Cisco Elastic Services Controller, page 57
- Authenticating REST Requests, page 57
- Configuring Openstack Credentials, page 59
- Reconfiguring ESC Virtual Machine, page 64
- Verifying ESC Configurations and Other Post-Install Operations, page 67
- Logging in to the ESC Portal, page 68

Changing the ESC Password

You will be forced to change the default password on first time login. Portal will not let you bypass this step and will keep returning you to this page until you change the default password. After the first time password change, you can change your password using the procedures described in this section. Also, if the user has multiple browsers or tabs or the SAME user is logged on by 2 or more computers and one of the user changes the password then everyone will be logged off and asked to re-enter the new password. The user session has an expiry of 1 hour so if the user is inactive on the portal for an hour then portal will expire the session and the user will have to re-login. If you forgot your password, you can also reset the password.

This section discusses how to change the passwords.

Changing the ConfD Netconf/CLI Administrator Password Using the Command Line Interface

After you install ESC, to change the Confd admin password, do the following:

**Step 1**
Log in to the ESC VM.

```
$ ssh USERNAME@ESC_IP
```
Step 2  Switch to the admin user.

```
[admin@esc-ha-0 esc]$ sudo bash
[sudo] password for admin:
```

Step 3  Load the ConfD CLI:

```
$ /opt/cisco/esc/confd/bin/confd_cli -u admin
```

Step 4  Set the new admin password:

```
$ configure
$ set aaa authentication users user admin password <new password>
```

Step 5  Save the changes.

```
$ commit
```

### Changing Linux Account Password

**Step 1**  Log in to ESC VM.

```
$ ssh USERNAME@ESC_IP
```

**Step 2**  To change the default password, use the following command:

```
passwd
[admin@esc-vm ~]$ passwd
Changing password for user admin.
Changing password for admin.
(current) UNIX password: ****
New password: *****
Retype new password: *****
```

### Changing the ESC Portal Password

**Step 1**  Log in to ESC VM.

**Step 2**  Switch to the root user.

**Step 3**  To reset to the default password (admin/cisco123), use one of the following method:

- Using escadm utility:
  
  ```
  escadm portal set
  escadm portal set --username admin --password <new password>
  ```

- Using the bootvm command line:
  
  ```
  --user_portal_pass admin:<new password>
  ```

- Using the ESC Portal:
  
  1. Log in to ESC portal using your username and password.
2 Choose Accounts Setting on the Navigation menu.
3 Enter the old password in the Old password field, then enter a new password in the New Password and Confirm Password fields.
4 Click Update Password.

Configuring Pluggable Authentication Module (PAM) Support for Cisco Elastic Services Controller

You can configure ESC services to use Pluggable Authentication Modules (PAM) for user authentication in ESC. With Cisco Elastic Services supporting PAM, you can also enable LDAP authentication in ESC. If PAM is not configured, ESC will continue to use the default authentication method for each ESC service. The following table lists the commands to enable PAM authentication for each ESC service.

Table 2: Configuring PAM for ESC Services

<table>
<thead>
<tr>
<th>ESC Service/Component</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC Manager (REST interface)</td>
<td>sudo escadm escmanager set --auth PAM:[[:&lt;pam_service_name&gt;]]</td>
</tr>
<tr>
<td>ESC Monitor (Health API)</td>
<td>sudo escadm monitor set --auth PAM:[[:&lt;pam_service_name&gt;]]</td>
</tr>
<tr>
<td>Confd</td>
<td>sudo escadm confd set --auth PAM:[[:&lt;pam_service_name&gt;]]</td>
</tr>
<tr>
<td>Portal</td>
<td>sudo escadm portal set --auth PAM:[[:&lt;pam_service_name&gt;]]</td>
</tr>
</tbody>
</table>

Note

- The SSHD service that runs inside the ESC VM already uses PAM authentication by default.
- If any component sets PAM authentication without specifying the PAM service, ESC defaults to the PAM service 'system-auth'.

Authenticating REST Requests

ESC REST API uses http basic access authentication where the ESC client will have to provide a username and password when making ESC REST requests. The user name and password will be encoded with Base64.
in transit, but not encrypted or hashed. HTTPS will be used in conjunction with Basic Authentication to provide the encryption.

This section discusses how to enable the REST authentication, change the default password of the REST interface, and how to send authorized REST request from the ESC client.

**Enabling REST Authentication**

By default, the REST authentication is disabled. To enable it, pass the argument `--enable-auth` to ESC bootvm.py, ESC installation script.

ESC also supports https communication over port 8443. ESC will generate a self-signed certificate that the client will need to trust to get the https communication going. By default, the REST https is disabled and restricted to localhost. To enable it, pass the argument `--enable-https-rest` to ESC bootvm.py, ESC installation script.

**Note**

Make sure to pass either `--enable-https-rest` or `--enable-http-rest` or both the arguments to the bootvm.py script along with the `--enable-auth` argument, to enable the authentication. To enable https or http after ESC VM is booted, use the escadm command specified below.

```
escadm escmanager set --url http://127.0.0.1:8080/ESCManager,https://0.0.0.0:8443/ESCManager
```

**Changing the REST Interface Password**

The REST interface has only one default username/password (admin/cisco123). The password can be updated after the bootup using escadm tool from the ESC VM CLI. You can also update the password through the REST API.

**Step 1** Log in to ESC VM.
**Step 2** Switch to the root user.
**Step 3** To reset to the default password, use the following command:

```
escadm rest set
```

**Step 4** To replace the existing password with a new one, use one of the below options:

- Using the escadm tool from the ESC VM CLI:
  
  ```
  escadm rest set --username admin --password <new password>
  ```

- Using the REST API:
  
  ```
  http://[ESCVM_IP]:8080/ESCManager/v0/authentication/setpassword?userName=admin&password=yourPassword
  ```
  or
  ```
  https://[ESCVM_IP]:443/ESCManager/v0/authentication/setpassword?userName=admin&password=yourPassword
  ```

Cisco Elastic Services Controller 3.1 Install and Upgrade Guide
Sending an Authorized REST Request

To send an authorized request, the ESC client should send the request with the following header:

```
Authorization: Basic YWRtaW46Y2lzY28xMjM=
```

where `YWRtaW46Y2lzY28xMjM=` is the Base64 encoded string of the default username/password.

Most libraries and web clients have an interface for providing the username/password and the application will encode the username/password and add the HTTP Basic Auth header.

Example using the default credentials:

For HTTP:
```
http://[ESCVM_IP]:8080/ESCManager/v0/tenants/
```

For HTTPS:
```
https://[ESCVM_IP]:8443/ESCManager/v0/tenants/
```

Configuring Openstack Credentials

If ESC was deployed without passing VIM credentials, you can set VIM credentials through ESC VIM and through VIM User APIs (REST or Netconf API).

Note
ESC will accept the northbound configuration request only if the following conditions are met:

- ESC has VIM or a VIM user configured through APIs (REST/Netconf).
- ESC has VIM or a VIM user configured, and ESC is able to reach the VIM.
- ESC has VIM or a VIM user configured, and ESC is able to authenticate the user.

Configuring using Netconf API

- Passing VIM credential using Netconf:

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <!--- represents a vim-->
    <vim_connector>
      <!-- unique id for each vim-->
      <id>my-ucs-30</id>
      <!-- vim type [OPENSTACK|VMWARE_VSPHERE|LIBVIRT|AWS|CSP]-->
      <type>OPENSTACK</type>
      <properties>
        <property>
          <name>os_auth_url</name>
          <value>http://<os_ip:port>/v3</value>
        </property>
        <!-- The project name for openstack authentication and authorization -->
        <property>
          <name>os_project_name</name>
          <value>vimProject</value>
        </property>
        <!-- The project domain name is needed for openstack v3 identity api -->
        <property>
          <name>os_project_domain_name</name>
        </property>
      </properties>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```
In 3.0, multiple VIM connectors are supported but within one ESC instance, all the VIM connectors have to in one VIM type (e.g. OpenStack, VMware).

One VIM is chosen as the default VIM which supports all pre 3.0 config requests and datamodels.

Deployments can be done on the VIM that is not the default VIM. The deployment to a non default VIM has to have all out-of-band resources (except ephemeral volumes). No other configurations like image, flavor, network, and so on can be done on the VIM that is not the default VIM.

The default VIM connector will be auto provisioned and does not need to be configured in the following scenarios:

- If VIM credentials have been passed during ESC boot up.
- If upgrading from 2.3.x to 3.0.

The change in the datamodel for Openstack create VIM connector would be handled during upgrade by migration. The 'os_tenant_name' and 'os_project_domain_name' properties would be moved to the VIM Connector properties and 'os_tenant_name' will be renamed to 'os_project_name'.

For the default VIM Connector, once it is properly authenticated, its properties cannot be updated.

VIM user can be deleted, recreated, or its properties can be updated at anytime.

**Note**

- In 3.0, multiple VIM connectors are supported but within one ESC instance, all the VIM connectors have to in one VIM type (e.g. OpenStack, VMware).
- One VIM is chosen as the default VIM which supports all pre 3.0 config requests and datamodels.
- Deployments can be done on the VIM that is not the default VIM. The deployment to a non default VIM has to have all out-of-band resources (except ephemeral volumes). No other configurations like image, flavor, network, and so on can be done on the VIM that is not the default VIM.

The default VIM connector will be auto provisioned and does not need to be configured in the following scenarios:

- If VIM credentials have been passed during ESC boot up.
- If upgrading from 2.3.x to 3.0.

The change in the datamodel for Openstack create VIM connector would be handled during upgrade by migration. The 'os_tenant_name' and 'os_project_domain_name' properties would be moved to the VIM Connector properties and 'os_tenant_name' will be renamed to 'os_project_name'.

For the default VIM Connector, once it is properly authenticated, its properties cannot be updated.

VIM user can be deleted, recreated, or its properties can be updated at anytime.

**Updating VIM Connector using Netconf:**

```xml
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <vim_connector nc:operation="replace">
      <id>example_vim</id>
      <type>OPENSTACK</type>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```
• **Updating VIM user using Netconf:**

```xml
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <vim_connector id="example_vim">
      <users>
        <user nc:operation="replace">
          <id>my_user</id>
          <credentials>
            <properties>
              <property>
                <name>os_password</name>
                <value>cisco123</value>
              </property>
            </properties>
          </credentials>
        </user>
      </users>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

• **Deleting VIM connector using Netconf:**

```xml
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <vim_connector nc:operation="delete">
      <id>example_vim</id>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

• **Deleting VIM Connector using command:**

```
$./opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli delete-vim-connector <vim connector id>
```

• **Deleting VIM user using command:**

```
$./opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli delete-vim-user <vim connector id> <vim user id>
```
Configuring using REST API

• Adding VIM using REST:

```xml
POST /ESCManager/v0/vims/
HEADER: content-type, callback

<?xml version="1.0"?>
<vim_connector xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <id>example_vim</id>
  <type>OPENSTACK</type>
  <properties>
    <property>
      <name>os_auth_url</name>
      <value>{auth_url}</value>
    </property>
    <property>
      <name>os_project_name</name>
      <value>vimProject</value>
    </property>
    <!-- The project domain name is only needed for openstack v3 identity api -->
    <property>
      <name>os_project_domain_name</name>
      <value>default</value>
    </property>
    <property>
      <name>os_identity_api_version</name>
      <value>3</value>
    </property>
  </properties>
</vim_connector>
```

• Adding VIM user using REST:

```xml
POST /ESCManager/v0/vims/{vim_id}/vim_users
HEADER: content-type, callback

<?xml version="1.0"?>
<user xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <id>my_user</id>
  <credentials>
    <properties>
      <property>
        <name>os_password</name>
        <value>cisco123</value>
      </property>
      <!-- The user domain name is only needed for openstack v3 identity api -->
      <property>
        <name>os_user_domain_name</name>
        <value>default</value>
      </property>
    </properties>
  </credentials>
</user>
```

• Update VIM using REST:

```xml
PUT /ESCManager/v0/vims/{vim_id}
HEADER: content-type, callback

<?xml version="1.0"?>
<vim_connector xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <!--unique id for each vim-->
  <id>example_vim</id>
  <type>OPENSTACK</type>
  <properties>
    <property>
      <name>os_auth_url</name>
      <value>{auth_url}</value>
    </property>
    <property>
      <name>os_project_name</name>
      <value>vimProject</value>
    </property>
  </properties>
</vim_connector>
```
Update VIM user using REST:

PUT /ESCManager/v0/vims/{vim_id}/vim_users/{vim_user_id}
HEADER: content-type, callback

<?xml version="1.0"?>
<user xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <id>my_user</id>
  <credentials>
    <properties>
      <property>
        <name>os_password</name>
        <value>cisco123</value>
      </property>
      <!-- The user domain name is only needed for openstack v3 identity api -->
      <property>
        <name>os_user_domain_name</name>
        <value>default</value>
      </property>
    </properties>
  </credentials>
</user>

Delete VIM using REST:

DELETE /ESCManager/v0/vims/{vim_id}

Delete VIM user using REST:

DELETE /ESCManager/v0/vims/{vim_id}/vim_users/{user_id}

Notification example after each VIM or VIM user configuration is done:

<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-10-06T16:24:05.856+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Created VIM connector successfully</status_message>
    <vim_connector_id>my-ucs-30</vim_connector_id>
    <event>
      <type>CREATE_VIM_CONNECTOR</type>
    </event>
  </escEvent>
</notification>

Important Notes:

- In ESC 3.0, you can add multiple VIM Connector for Openstack VIM. Each VIM Connector can have only one VIM User.
- VIM username and password can be updated at anytime. VIM endpoint will not be able to update after a resource is created through ESC.
- After VIM is connected and VIM user is authenticated, VIM can no longer be deleted or updated, only VIM user can be deleted or updated.
Reconfiguring ESC Virtual Machine

This section covers the following:

- Reconfiguring Rsyslog
- Reconfiguring NTP
- Reconfiguring DNS
- Reconfiguring Hosts
- Reconfiguring Timezone

Reconfiguring Rsyslog

Rsyslog parameters are optional. If there is a need for customization after booting an ESC VM, you can edit the files in ESC VM (/etc/rsyslog.d/).

Step 1  Editing the Rsyslog file:

- If you haven't specified the log forwarding configuration at the bootup time, you may create a file under /etc/rsyslog.d/ like /etc/rsyslog.d/log-forwarding.conf.

- If you have specified the log forwarding through installation, you may just need to edit the file. The file could be /etc/rsyslog.d/20-cloud-config.conf. In this file, to forward logs to multiple rsyslog servers, edit the following line:

  *.*  @[server_ip]:port

Note

- Use '@@' before specifying server ip address (if TCP is the protocol used to forward logs to the rsyslog server).

- Use '@' before specifying server ip address (if UDP is the protocol used to forward logs to the rsyslog server).

- server_ip can either be ipv4/ipv6 address of the rsyslog server.

- '[' around the server_ip is required to separate it from ':port#', if an ipv6 server address is specified.

For further information on Rsyslog configuration, see the Red Hat documentation.

Step 2  Configuring the ESC log file: Configure which ESC log files you want to forward to the rsyslog server:

a) Navigate to /etc/rsyslog.d/ Create or modify a configuration file, such as log-esc.conf. Make a copy of sample log-esc.conf.

b) Specify the following block for every file you want to forward to rsyslog server.

```bash
$InputFileName /var/log/esc/escmanager.log
$InputFileTag esc-manager:
$InputFileStateFile stat-esc-manager
```
Step 3  Restart the rsyslog service

   # service rsyslog restart

Step 4  Configure the server side to receive forwarded logs.

   a) On a designated server, go to /etc/rsyslog.conf, and uncomment the lines shown below, depending on if you want to listen to logs from clients based on TCP or UDP:

   #$ModLoad imudp
   #$UDPServerRun 514

   b) Exit the file. Run this command as the last step.

   sudo service rsyslog restart
   Now, the server is listening for logs on port 514, using TCP/UDP.

Reconfiguring NTP

Step 1  Open the NTP configuration file /etc/ntp.conf in a text editor such as vi, or create a new one if it does not already exist:

   # vi /etc/ntp.conf

Step 2  Add or edit the list of public NTP servers. If you don't specify the NTP server through the installation, the file should contain the following default lines, but feel free to change or expand these according to your needs:

   server 0.rhel.pool.ntp.org iburst
   server 1.rhel.pool.ntp.org iburst
   server 2.rhel.pool.ntp.org iburst
   server 3.rhel.pool.ntp.org iburst
   server <your_ntp_server_ip> iburst

   The iburst directive at the end of each line speeds up the initial synchronization.

Step 3  Once you have the list of servers complete, in the same file, set the proper permissions, giving the unrestricted access to localhost only. Make sure those lines are there in your configure file.

   restrict default kod nomodify notrap nopeer noquery
   restrict -6 default kod nomodify notrap nopeer noquery
restrict 127.0.0.1
restrict -6 ::1

Step 4  Save all changes, exit the editor, and restart the NTP daemon:
# service ntpd restart

Step 5  Make sure that ntpd is started at boot time:
# chkconfig ntpd on

---

Reconfiguring DNS

Step 1  The /etc/resolv.conf file contains the configuration for the DNS client (resolver). It typically looks something like this:

```
search domain.com
nameserver 8.8.4.4
```

Step 2  You may modify the IP address of the "nameserver" item or add new nameserver records.

```
search domain.com
nameserver <your_first_dns_ip>
nameserver <your_second_dns_ip>
```

Step 3  Restart Network Service.
          service network restart

---

Reconfiguring Hosts

The /etc/hosts file allows you to add, edit, or remove hosts. This file contains IP addresses and their corresponding hostnames. If your network contains computers whose IP addresses are not listed in DNS, it is recommended that you add them to the /etc/hosts file.

Step 1  Add the IP addresses that are not listed in DNS to the /etc/hosts file.

Step 2  Restart your network for the changes to take effect.
          service network restart

---

Reconfiguring Timezone

For ESC VM, in /etc the file "localtime" is a link to or copy of a file containing information about your time zone. Access your zone information files from /usr/share/zoneinfo. To change the time zone, find your country, your city or a city in the same time zone from zone information files in /usr/share/zoneinfo and link it to the localtime in the /etc file.

```
$ ln -sf /usr/share/zoneinfo/America/Los_Angeles /etc/localtime
```
Verifying ESC Configurations and Other Post-Install Operations

This section covers various post-install checks and operations using the escadm tool.

Verifying Existing ESC Configurations

You can use `escadm dump` command for displaying current ESC configurations in yaml format. The output will show the various services in ESC.

```
$ escadm dump
resources:
  confd:
    init_aaa_users:
      - name: admin
        passwd:
          option: start-phase0
  esc_service:
    group:
      - confd
      - mona
      - vimmanager
      - postgres
      - escmanager
      - portal
      - monitor
      - snmp
      type: group
    escmanager: {}
    mona: {}
    monitor: {}
    postgres: {}
    portal: {}
    snmp:
      run_forever: true
    vimmanager: {}
```

Verifying VIM configurations

You can use `escadm vim show` command to verify the vims settings are correctly populated:

```
$ escadm vim show
[
  {
    "status": "CONNECTION_SUCCESSFUL",
    "status_message": "Successfully connected to VIM",
    "type": "OPENSTACK",
    "id": "default_openstack_vim",
    "properties": {
      "property": {
        "name": "os_auth_url",
        "value": "http://10.85.103.143:35357/v3"
      }
    }
  }
]
```

Troubleshooting ESC Services Startup Issues

**Problem:** Issues encountered while verifying ESC services status at the installation time using `escadm status`.

**Causes:** Some services take time to start or have trouble starting.
Solution:

1. Identify the issues using one of the following method:
   - Check the log `/var/log/esc/escadm.log`
     ```
     $ cat /var/log/esc/escadm.log
     2017-06-01 20:35:02,925: escadm.py(2565): INFO: promote drbd to master...
     2017-06-01 20:35:02,934: escadm.py(2605): INFO: Waiting for at least one drbd to be UptoDate...
     2017-06-01 20:35:02,942: escadm.py(2616): INFO: Waiting for peer drbd node to be demoted...
     2017-06-01 20:35:14,017: escadm.py(1755): INFO: Starting filesystem service: [OK]
     2017-06-01 20:35:15,039: escadm.py(1755): INFO: Starting vimmanager service: [OK]
     2017-06-01 20:35:17,163: escadm.py(1755): INFO: Starting mona service: [OK]
     2017-06-01 20:35:18,440: escadm.py(1755): INFO: Starting snmp service: [OK]
     2017-06-01 20:35:21,397: escadm.py(1770): INFO: Starting confd service: [FAILED]
     2017-06-01 20:35:28,504: escadm.py(1755): INFO: Starting pgsql service: [OK]
     2017-06-01 20:35:29,331: escadm.py(1755): INFO: Starting escmanager service: [OK]
     2017-06-01 20:35:30,354: escadm.py(1755): INFO: Starting portal service: [OK]
     2017-06-01 20:35:31,523: escadm.py(1755): INFO: Starting esc_service service: [OK]
     ```
   - Add `-v` to escadm status to show the verbose output of the ESC services.
     ```
     $ escadm status --v
     0 ESC status=0 ESC HA Master Healthy
     pgsql (pgid 61397) is running
     vimmanager (pgid 61138) is running
     monitor (pgid 61162) is running
     mona (pgid 61190) is running
     drbd is master
     snmp (pgid 61541) is running
     filesystem (pgid 0) is running
     bgp is dead
     keepalived (pgid 60838) is running
     portal (pgid 61524) is running
     confd (pgid 61263) is running
     escmanager (pgid 61491) is running
     ```

2. Confirm the status of the identified services that has issues and manually start these services.
   ```
   $ escadm bgp status // If the status is stopped or dead, manually start the services using the next command.
   $ escadm bgp start --v
   ```

Logging in to the ESC Portal

- The ESC portal is enabled by default. You must ensure that the ESC portal is not disabled during installation. For more information on enabling or disabling the ESC portal, see Installing Cisco Elastic Services Controller Using the QCOW Image, on page 10.
- When you log in to the ESC portal for the first time you are prompted to change the default password.

To log in to the ESC portal, do the following:
Before You Begin

- Register an instance of ESC. For more information on registering the ESC instance see, Installing Cisco Elastic Services Controller Using the QCOW Image, on page 10
- Ensure that you have the username and password.

---

**Step 1**
Using your web browser, enter the IP address of ESC and port 443.

**Example:**
For example, if the IP address of ESC is 192.0.2.254, enter:
```
https://192.0.2.254:443
```
A Security Alert message is displayed.

**Step 2**
Click Yes to accept the security certificate. The Login page is displayed.

**Step 3**
Enter the username and password and click Login.
If you are logging in for the first time, the login page reappears, prompting you to change your password.

**Step 4**
Enter the old password in the Old Password field, then enter a new password in the New Password and Confirm Password fields.

**Step 5**
Click Update Password or press Enter.

**Note**
- If the UI becomes unresponsive, restart the UI by executing the `escadm portal restart` from the ESC shell prompt.
- ESC Portal only supports one user.
- Currently, a pre-installed self-signed certificate supports HTTPS. The user must confirm the self-signed certificate before proceeding with the ESC Portal.
- In HTTPS communication mode, if the URL protocol type returned by OpenStack is not HTTPS, the access to the VNF Console may be disabled. For security reasons, while running in HTTPS more non-secure communication will be rejected.
PART V

Upgrading Cisco Elastic Services Controller

• ESC in Maintenance Mode, page 73
• Upgrading Cisco Elastic Services Controller, page 79
Setting ESC in a Maintenance Mode

ESC must be put to maintenance mode to backup and restore ESC database. To do so, use the escadm tool as specified in the below section.

Before You Begin

During maintenance mode,

- Northbound requests are blocked by ESC and ESC responds with maintenance mode notification.
- Only REST requests receive response that ESC is unavailable temporarily. ConfD requests get the maintenance mode rejection message, or an OK message for all idempotent request such as create tenant request when the tenant already exists.
- Monitoring actions are paused.
- All ongoing requests and transactions continue to progress.

Using the escadm Tool

ESC can be put to maintenance mode using the escadm tool.

Step 1

Put ESC to maintenance mode from the VM shell:

```
sudo escadm op_mode set --mode=maintenance
```
Backup the Database from the ESC Standalone Instances

- The following assumptions should be taken into consideration:
  - A third machine is required to store the database and log backups.
  - ESC does not support database schema downgrade. Restoring database to the older ESC version could cause unexpected problems.
  - Before you start the backup process, ensure you have an external storage space (could be in the OpenStack controller or any system accessible by ESC). The backup/restore could be expressed in a generic format which will be used by the escadm tool: scp://<username>:<password>@<backup_ip>:<filename>. In this format, the credentials, IP address and file storage path of the third machine are required. You may also use localhost IP as the backup IP to backup database in a location of ESC VM and then copy the files to the external storage.

To backup the ESC database from a standalone ESC or a HA (master node):

Step 1
Log in to ESC VM and set it to maintenance mode, run:

$ escadm op_mode set --mode=maintenance

Step 2
To make sure ESC is in maintenance mode, run:

$ escadm op_mode show

Setting ESC in an Operation Mode

Put ESC in operation mode using the escadm tool:

sudo escadm op_mode set --mode=operation

Response is as follows:

Set mode to OPERATION
Operation Mode = OPERATION

Verify ESC’s operation mode at any time using the following command:

sudo escadm op_mode show

Step 2
To query operation mode at any time,

sudo escadm op_mode show

Example:

Operation Mode = OPERATION

Step 3
Set maintenance mode when there is no in-flight transaction. Using the ipt_check flag with the escadm tool, you can choose to set ESC in the maintenance mode only if there are no ongoing transactions in ESC. Set the flag to true, if you do not want ESC to set in the maintenance mode, if there are ongoing transactions in ESC.

sudo escadm op_mode set --mode=maintenance --ipt_check=true

With the ipt_check option set to true, escadm tool checks if there is any ongoing operation, if so, the escadm tool will not set ESC to maintenance mode.

Backup the Database from the ESC Standalone Instances

- The following assumptions should be taken into consideration:
  - A third machine is required to store the database and log backups.
  - ESC does not support database schema downgrade. Restoring database to the older ESC version could cause unexpected problems.

Before you start the backup process, ensure you have an external storage space (could be in the OpenStack controller or any system accessible by ESC). The backup/restore could be expressed in a generic format which will be used by the escadm tool: scp://<username>:<password>@<backup_ip>:<filename>. In this format, the credentials, IP address and file storage path of the third machine are required. You may also use localhost IP as the backup IP to backup database in a location of ESC VM and then copy the files to the external storage.

To backup the ESC database from a standalone ESC or a HA (master node):

Step 1
Log in to ESC VM and set it to maintenance mode, run:

$ escadm op_mode set --mode=maintenance

Step 2
To make sure ESC is in maintenance mode, run:

$ escadm op_mode show
Step 3  Backup database using the commands below:

```
$ escadm backup --file /tmp/db_file_name.tar.bz2
scp://<username>:<password>@<backup_vm_ip>:<filename>
```

Step 4  To put ESC back to operation mode, run:

```
$ escadm op_mode set --mode=operation
$ escadm op_mode show
```

Step 5  Collect all the logs from the old ESC VM and back it up.

```
$ escadm log collect
```

A timestamped log file will be generated in: /var/tmp/esc_log<timestamp>.tar.bz2

**Note**  If a dynamic mapping file is used by ESC service, the dynamic mapping file should be backed up at the same time with ESC logs. The default path of the dynamic mapping file is /opt/cisco/esc/esc-dynamic-mapping/dynamic_mappings.xml.

Step 6  After a successful database back-up, shut down the old ESC VM using Horizon/Kilo or Nova commands. For ESC VM instances based in VMware vSphere, shutdown the primary instance through VMware client dashboard. An example of shutting down a VM in OpenStack is shown below:

```
$ nova stop OLD_ESC_ID
```

Step 7  Detach the old port from the old VM and rename the old ESC node. Examples of detaching and renaming the VM in OpenStack is shown below:

```
nova interface-detach ESC_NAME port-id-of-ESC_NAME
nova rename ESC_NAME ESC_NAME.old
```

In VMware, assign a different IP address to the old VM and then rename the old VM.

---

**Backup the Database from the ESC HA Instances**

- The following assumptions should be taken into consideration:
  - A third machine is required to store the database and log backups.
  - ESC does not support database schema downgrade. Restoring database to the older ESC version could cause unexpected problems.

- Before you start the backup process, ensure you have an external storage space available (could be in the OpenStack controller or any system accessible by ESC). The backup/restore could be expressed in a generic format which will be used by the escadm tool:

  ```
  scp://<username>:<password>@<backup_ip>:<filename>.
  ```

  In this format, the credentials, IP address and file storage path of the third machine are required. You may also use localhost IP as the backup IP to backup database in a location of ESC VM and then copy the files to the external storage.

To backup the ESC database from a standalone ESC or a HA (master node):

---

**Step 1**  Perform the following steps on the Standby ESC node.
a) Connect to the standby ESC instance using SSH:
   $ ssh <username>@<backup_vm_ip>
b) Verify that the ESC instance is standby and note the name of the standby ESC HA instance:
   $ escadm status --v
   If the output value shows "BACKUP", the node is the standby ESC node.
c) Change access to an admin user.
   sudo bash
d) Collect all the logs from the standby ESC VM and back it up.
   $ escadm log collect
   A timestamped log file will be generated in: /var/tmp/esc_log<timestamp>.tar.bz2
e) Shutdown the standby ESC instance through OpenStack Kilo/Horizon using Nova command or VMware client. An example of shutting down the VM on OpenStack is shown below:
   $ nova stop OLD_ESC_STANDBY_ID

Step 2  Perform the following steps on the Master ESC node.
a) Connect to the primary ESC instance using SSH:
   $ ssh <username>@<master_vm_ip>
b) Change access to an admin user.
   $ sudo bash
c) Verify that the ESC instance is Master and note the name of the Master ESC HA instance
   $ escadm status --v
   If the output value shows "MASTER", the node is the Primary ESC node.
d) Back up the database files from the master node of ESC HA:
   $ escadm backup --file /tmp/db_file_name.tar.bz2
   scp://<username>:<password>@<backup_vm_ip>:<filename>
e) Collect the logs from the master ESC VM and back it up.
   $ escadm log collect
   A timestamped log file will be generated in: /var/tmp/esc_log<timestamp>.tar.bz2
   Note  If a dynamic mapping file is used by ESC service, the dynamic mapping file should be backed up at the same time with ESC logs. The default path of the dynamic mapping file is /opt/cisco/esc/esc-dynamic-mapping/dynamic_mappings.xml.

Step 3  Shutdown the primary ESC instance through OpenStack Kilo/Horizon using Nova command. For ESC VM instances based in VMware vSphere, shutdown the primary instance through VMware client dashboard. An example of shutting down the VM on OpenStack is shown below:
   $ nova stop OLD_ESC_MASTER
   Use the nova list command or the nova show OLD_ESC_STANDBY command, to verify if the ESC HA instances have been successfully shut down.

Step 4  (Only for OpenStack) Detach the port from the old ESC VM and rename the old VM.
   If upgraded ESC VM needs to operate with same IP addresses and same instance names as old instances, detach the ports from each instance, shutdown the old ESC VMs and then rename the old ESC instances.
If you intend to use old VMware primary instance, assign a different IP address and rename the VM name. If not, you can delete the old VM and use the same IP address for the new upgraded VMware primary instance. After deleting the old VM, you can continue with the old instance name and the IP address.

OpenStack commands for detaching the ports and renaming the old VMs are shown below:

```bash
nova interface-list ESC_NAME
nova interface-detach ESC_NAME port-id-of-ESC_NAME
nova rename ESC_NAME ESC_NAME.old
```

## Restoring ESC Database

**Before You Begin**

To restore the database,

- In standalone ESC instance, stop ESC services. Run `# escadm stop`.
- In HA type instances, stop KeepAliveD on the Backup first, and later on the Master ESC HA instance. Run `# escadm stop`.
- All the services must be stopped. To check the status, run `# escadm status --v`.

**Step 1**

Restore the database:

```bash
$ escadm restore --file /tmp/db_file_name.tar.bz2
scp://<username>:<password>@<backup_vm_ip>:<filename>
```

**Step 2**

Enter the ESC password in the URL, or manually enter it after executing the above command.

**Step 3**

Restart the ESC service to complete the database restore by running the following command:

```bash
$ sudo escadm restart
```

**Note**

ESC maintenance mode blocks the northbound request and VNF monitoring. However, if there are some ongoing transactions because of northbound requests before ESC entering maintenance mode, those transactions may have following restriction with backup and restore:

- ESC reports an error for the deployment, network creation, and subnet creation requests, if these transactions are interrupted by backup and restore. Northbound handles these error messages but it may cause network or subnet leakage in some cases (For example, ESC is interrupted before getting the UUID from OpenStack).
- ESC reports an error for service chain upgrade, and requires service chain undeployment and deployment (rather than downgrade and upgrade) to re-create the service.
Upgrading Cisco Elastic Services Controller

Cisco Elastic Service Controller supports two type of upgrades:

- **Backup and Restore Upgrade**: This upgrade process involves stopping the ESC keepalived daemon (for ESC HA), backing up the database, stopping and renaming (or deleting) the ESC instances, re-installing the ESC instances, and restore database. For information on the supported ESC versions for ESC 2.2 upgrade, see the table below.

- **In-service upgrade**: ESC supports in-service upgrade for high-availability nodes with a minimum downtime.

You can upgrade the ESC instance as a standalone instance or as a high availability pair. The upgrade procedure is different for standalone and high availability pair.

This chapter lists separate procedures on how to upgrade ESC standalone and ESC High Availability instance. You must review these instructions before you decide to upgrade the ESC instance. See the Installation Scenarios, on page 9 for more information on the installation scenarios.

- ESC only support direct upgrade from previous two minor releases. For example, ESC 2.3 will support direct upgrade from ESC 2.1 and ESC 2.2. For any release older than the supported versions for direct upgrade, you need to perform the staged upgrade.

- Upgrading ESC using RPM Package (referred to as RPM Upgrade in this chapter) only applies to the ESC upgrade between ESC maintenance releases with the same minor release number. For Example, the upgrade from ESC 2.3.1 to ESC 2.3.2. If you want to upgrade ESC between minor releases (for example, upgrade from ESC 2.2.9 to ESC 2.3.1) or major releases (for example, upgrade from ESC 2.2 to ESC 2.3 ), you can upgrade through Backup and Restore upgrade process using qcow2 image.

- For ESC upgrade, you should be familiar with ESC installation process.
  - For OpenStack, refer to the OpenStack installation procedures, see Chapter 4: Installing Cisco Elastic Services Controller in OpenStack.
  - For VMware, refer to the VMware Installation installation procedures, see Chapter 7: Installing Cisco Elastic Services Controller in VMware vCenter.
  - For ESC HA, please refer to the ESC HA installation procedures, see Chapter 5: Configuring High Availability for OpenStack and Chapter 8: Configuring High Availability for VMware.
### Table 3: Supported ESC Versions for Upgrading to ESC 3.0

<table>
<thead>
<tr>
<th>Virtual Infrastructure Manager</th>
<th>Supported Versions for Backup and Restore Upgrade</th>
<th>Supported Versions for In-Service Upgrade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Infrastructure Manager</td>
<td>Supported Versions for Backup and Restore Upgrade</td>
<td>Supported Versions for In-Service Upgrade</td>
</tr>
<tr>
<td>OpenStack</td>
<td>2.1, 2.2 (2.2.8 and 2.2.9), 2.3 (2.3.1 and 2.3.2)</td>
<td>2.2 (2.2.8 and 2.2.9), 2.3 (2.3.1 and 2.3.2)</td>
</tr>
<tr>
<td>VMware</td>
<td>2.1, 2.2 (2.2.8 and 2.2.9), 2.3 (2.3.1 and 2.3.2)</td>
<td>-</td>
</tr>
</tbody>
</table>

**IMPORTANT NOTES**

- ESC portal now displays the notification data that was present in the database, even after the upgrade. This feature is supported only from ESC 2.1. If you are upgrading from 1.1 to 2.1 or later, you will not be able to see the notifications from the 1.1 release on the ESC portal as this data was not present in the database.

- After upgrading to the new ESC version, ESC service will manage the life cycle of all VNFs deployed in the previous release. To apply any new features (with new data models) to the existing VNFs, you must undeploy and redeploy these VNFs.

- **Upgrading Standalone ESC Instance, page 80**
- **Upgrading ESC HA Instances, page 81**
- **In-Service Upgrade of the ESC HA Nodes in OpenStack, page 84**
- **In-Service Upgrade of the ESC HA Nodes in Kernel-Based Virtual Machine (KVM), page 88**
- **In-Service Upgrade of the ESC HA Nodes in VMware, page 92**

## Upgrading Standalone ESC Instance

To upgrade standalone ESC instance, perform the following tasks:

1. Back up the ESC database. For more information, see *Backup the Database from the ESC Standalone Instances*.
2. Redeploy the ESC instance. For more information, see the below section, *Deploy the ESC for Upgrade*.
3. Restore the ESC database on the new ESC instance. For more information, see the below section, *Restoring the ESC Database*.

### Deploy the ESC for Upgrade

After backing up and shutting down of the old ESC VM, a new/upgraded (based on new ESC package) ESC VM should be installed. All parameters for ESC installation should be the same as the old ESC VM deployment.

- For OpenStack, you need to register the new ESC qcow2 image using the Glance command with a new image name and then use new bootvm.py script and new image name to install ESC VM.
In OpenStack, if an old ESC VM was assigned with floating IP, the new ESC VM should be associated with the same floating IP after the installation.

- For VMWare, you need to use the new ESC OVA file to install ESC VM. All other configurations and property values should be the same as the old VM.

**Restoring the ESC Database**

Restore the ESC database on the new ESC instance, using the following procedure:

**Step 1** Connect to the new ESC instance using SSH.

```
$ ssh USERNAME@NEW_ESC_IP
```

**Step 2** Switch to the root user.

```
$ sudo bash
```

**Step 3** Stop the ESC service.

```
$ escadm stop
```

**Step 4** Check ESC service status to make sure all the services are stopped.

```
$ escadm status
```

**Step 5** Restore the database files.

```
$ escadm restore --file /tmp/db.tar.bz2
```

```
$ scp://<username>:<password>@<backup_ip>:<filename>
```

**Note** If a dynamic mapping file (dynamic_mapping.xml) is used by ESC service, the dynamic mapping file should be restored into the ESC VM. Before starting ESC service, you need to copy the backup dynamic mapping file (dynamic_mapping.xml) to the path `/opt/cisco/esc/esc-dynamic-mapping/`

**Step 6** Restart the ESC service:

```
$ escadm restart
```

After ESC service is started, the standalone ESC upgrade is complete. You can check the health of the new ESC service by running `$ escadm status` in the new ESC VM.

**Step 7** In Openstack, after restoring the database successfully, delete the old ESC instance:

```
$ nova delete OLD_ESC_ID
```

**Important Notes:**

After upgrading to the new ESC version, ESC service will keep doing life cycle management of all VNFs deployed by the old version. However, to apply any new features (with new data models) to the VNFs deployed by the ESC with old version is not guaranteed. If you want to apply any new feature of the new ESC version to existing VNFs, you have to undeploy and redeploy those VNFs.

**Upgrading ESC HA Instances**

To upgrade ESC HA nodes, perform the following tasks:
1. Back up the database from an old ESC HA primary instance. For more information, see Backup the Database from the ESC HA Instances.

2. Deploy new ESC HA nodes based on new ESC version. For more information, see the below section, Deploy the ESC HA nodes for Upgrade.

3. Restore the Database on Primary ESC instance (Standby ESC instance will sync with the Primary ESC instance). For more information, see the below section, Restoring the ESC Database on New Master and Standby Instances.

Deploying the ESC HA nodes for Upgrade

After backing up and shutting down the two old ESC VMs, based on new ESC package install the new ESC VMs.

- For OpenStack, you need to register the new ESC qcow2 image using the Glance command with a new image name and then to use new bootvm.py script and new image name to install ESC VM. All other bootvm.py arguments should be the same as used to setup an old VMs.

- For VMWare, there are two steps to bring up HA pair in VMware: 1) setup two standalone instances 2) reconfigure each instance with HA info. All other configurations and property values should be the same as the old VMs.

- If VIP is used for Northbound access, keep VIP the same for the new deployment as used to reconfigure the old HA pair. If BGP is used for northbound access, keep the BGP parameters the same as the old HA pair.

Restoring the ESC Database on New Master and Standby ESC Instances

Shut down the Standby ESC instance.

Step 1
Connect to the standby ESC instance using SSH.

$ ssh USERNAME@ESC_STANDBY_IP

Step 2
Verify that the ESC instance is standby and note the name of the standby ESC HA instance:

$ escadm status

If the output value shows "BACKUP", the node is the standby ESC node.

Note
If a dynamic mapping file (dynamic_mapping.xml) is used by ESC service, the dynamic mapping file should be restored into the backup ESC VM. Before power off the standby ESC node, you need to copy the backup dynamic mapping file (dynamic_mapping.xml) to the path /opt/cisco/esc/esc-dynamic-mapping/.

Step 3
Shutdown the standby ESC instance through OpenStack Kilo/Horizon using Nova command. For ESC VM instances based in VMware vSphere, shutdown the primary instance through VMware client dashboard. An example of shutting down the standby ESC instance in OpenStack is shown below:

$ nova stop NEW_ESC_STANDBY_ID

Restore the database on the new Master ESC instance.

Step 4
Connect to the primary ESC instance using SSH.

$ ssh USERNAME@ESC_MASTER_IP

Step 5
Switch to the root user.

$ sudo bash
Step 6 Verify that the ESC instance is primary.
$ escadm status
If the output value shows 'MASTER', the node is the master ESC node.

Step 7 Stop the ESC services on the master node and verify the status to ensure the services are stopped.
$ escadm stop
$ escadm status

Step 8 Restore the database files.
$ escadm restore --file /tmp/db.tar.bz2
$ scp://<username>:<password>@<backup_ip>:<filename>

Note If a dynamic mapping file (dynamic_mapping.xml) is used by ESC service, the dynamic mapping file should be restored into the ESC VM. Before starting the ESC node, you need to copy the backup dynamic mapping file (dynamic_mapping.xml) to the path /opt/cisco/esc-dynamic-mapping/.

Step 9 Reboot the VM to restart the full ESC service:
$ escadm restart

Step 10 Use the $ escadm status to check the status of the ESC service.

Step 11 Start the standby ESC node.
Power on the standby ESC node through OpenStack Nova/Horizon or VMware client. After starting the standby node, ESC HA upgrade process should be complete.

Step 12 Delete the old HA instance through OpenStack Nova/Horizon or VMware client. An example of deleting the VM on OpenStack is shown below:
$ nova delete OLD_ESC_MASTER_RENAMED OLD_ESC_STANDBY_RENAMED

Upgrading VNF Monitoring Rules

In ESC 2.1 and earlier, mapping the actions and metrics defined in the datamodel to the valid actions and metrics available in the monitoring agent is enabled using the dynamic_mappings.xml file. The file is stored in the ESC VM and can be modified using a text editor. ESC 2.2 and later do not have an esc-dynamic-mapping directory and dynamic_mappings.xml file. The CRUD operations for mapping the actions and the metrics is available through REST API.

To upgrade the VNF monitoring rules, you must back up the dynamic_mappings.xml file and then restore the file in the upgraded ESC VM. For more information, see the backup and restore procedures. For upgrade of HA instance, see Upgrading ESC HA Instances. For upgrade of the standalone instance, see Upgrading Standalone ESC Instance.
In-Service Upgrade of the ESC HA Nodes in OpenStack

In-Service upgrade in OpenStack using ESC RPM packages

Use this procedure to upgrade the ESC high-availability nodes one node at a time with a minimum service interruption. This process leverages the ESC HA replication and failover capability to smoothly move ESC service to the new upgraded node without the manual database restore.

Step 1  Backup ESC database and log files.
  a) Perform ESC database backup from primary node. For more information on backing up the database, see Backup the Database from the ESC HA Instances.
  b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

        $ escadm log collect

       Note  A timestamped file will be generated in: /var/tmp/esc_log-<timestamp>.tar.bz2
  c) Copy the database backup file and logs files (generated in /tmp/esc_log-.tar.bz2)* out of ESC VMs.
  d) Copy the preupgrade.sh script and RPM files to both Primary ESC VM and Secondary ESC VMs. Execute the following command in both ESC VMs:

        $ sudo bash preupgrade.sh

         Expect output:
        Success

Step 2  Log into the ESC HA secondary VM and stop the services.

        $ sudo escadm stop

Step 3  Make sure the secondary ESC VM is in STOP state.

        $ escadm status --v
       If ESC status=0 esc ha is stopped.

Step 4  Copy rpm file into secondary ESC VM and execute the rpm command for upgrade:

        $ sudo rpm -Uvh /home/admin/cisco-esc-2.2.9-50.rpm

Step 5  Log into the primary instance, set ESC primary node into maintenance mode.

        $ escadm op_mode set --mode=maintenance
       Make sure there are no in-flight transactions and no new transactions during the upgrade. From ESC 2.3, you may use following commands to check in-flight transactions.

        $ escadm ip_trans
       For releases older than ESC 2.3, you may need to check escmanager log and make sure no new transactions are recorded in this log file. The log file can be located at (/var/log/esc/escmanager.log).

Step 6  In OpenStack controller, power off ESC primary node and make sure it is completely shut down by OpenStack.

        $ nova stop <primary_vm_name>

        $ nova show <primary_vm_name>
Step 7 Log in to the upgraded ESC instance (previous secondary one), and start the ESC service. The upgraded VM will take over primary role and provide ESC service.

```
$ sudo escadm restart
```

Step 8 Check the ESC version on the new primary instance to verify the upgraded version is correct. Once it is in the Primary state, make sure ESC service is running properly in the new Primary VM.

```
$ escadm status
Expected output:
  0 ESC status=0 ESC Master Healthy

$ esc_version
```

Step 9 Power on the old primary instance. In OpenStack controller, execute the following command:

```
$ nova start <primary_vm_name>
$ nova show <primary_vm_name>
```

Step 10 Log into the VM which is still with old ESC version and repeat step 2, 3, 4, and 7 in the VM.

**Note** For a Quick rollback: For a quick rollback to the previous version, you can just power off the upgraded primary instance. You can get the old ESC service back immediately. Run the following command in the old primary instance.

```
sudo bash preupgrade.sh --revoke
```

You can then redeploy the secondary instance to fully rollback the HA.

---

**In-Service upgrade in OpenStack using ESC qcow2 Image**

**Step 1** Backup ESC database and log files.

a) Perform ESC database backup from primary node. For more information on backing up the database, see *Backup the Database from the ESC HA Instances*.

b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

```
$ escadm log collect
```

**Note** A timestamped file will be generated in: /var/tmp/esc_log-<timestamp>.tar.bz2

c) Copy the database backup file and logs files (generated in /tmp/esc_log-.tar.bz2) out of ESC VMs.

d) Copy the preupgrade.sh script to both Primary ESC VM and Secondary ESC VMs. Make sure the file has execution mode set. Execute the following command in both ESC VMs:

```
chmod +x preupgrade.sh
$ sudo bash preupgrade.sh
```

**Expect output:**

```
Success
```

**Step 2** Redeploy secondary ESC instance. Register new ESC image on the secondary instance, and wait for the data to be synchronized.
a) Delete the secondary instance through Horizon/Kilo using OpenStack Nova client. In OpenStack controller, running following command through nova client.

```
nova delete <secondary_vm_name>
```

b) Register new ESC image into OpenStack Glance for redeployment usage.

```
glance image-create --name <image_name> --disk-format qcow2 --container-format bare --file <esc_qcow2_file>
```

c) Redeploy the secondary ESC VM instance based on newer image version. Re-install new the secondary instance by using the new ESC package (bootvm.py and new registered image). All other installation parameters should be the same as the former ESC VM deployment. For example, hostname, ip address, gateway_ip, ha_node_list, kad_vip, kad_vif have to use the same values. Once the new ESC instance with upgraded version is up, it will be in secondary state.

d) Log into the new instance and run the following command to check the synchronization state of the new ESC node.

```
# drbd-overview
```

Wait until the output of drbd-overview show both nodes are "UpToDate" like the output below. It means the new ESC instance has completed the data synchronization from the primary instance.

```
esc/0 Connected Secondary/Primary UpToDate/UpToDate
```

**Step 3**

Stop keepalived service on Secondary instance, Power off primary instance, and then start Secondary keepalived service.

a) Log into the primary instance, set ESC primary node into maintenance mode.

```
$ escadm op_mode set --mode=maintenance
```

Make sure there is no in-flight transaction ongoing before moving to the next step. To verify there are no in-flight transactions, use the following command:

```
For ESC 2.3:
$ escadm ip_trans
For versions older than ESC 2.3, check escmanager log at (/var/log/esc/escmanager.log) and make sure there are no new transaction in escmanager log.
```

b) Log in to the upgraded secondary instance and shut down the ESC service.

```
$ escadm stop
```

c) Power off the primary instance through OpenStack Nova client/Horizon and make sure it is off. In OpenStack Controller, run:

```
$ nova stop <primary_vm_name>
$ nova list | grep <primary_vm_name>
```

d) Log into the previously upgraded secondary instance which is in stopped state and restart the ESC service. The secondary ESC instance will take the primary role (switchover will be triggered) and start providing services with new version.

```
$ sudo escadm restart
```

**Step 4**

Check the ESC version on the new primary instance to verify the version is upgraded correctly.

```
$ escadm status (check ha status)
```

Expected output:

```
0 ESC status=0 ESC Master Healthy
```

```
$ esc_version (check esc version)
version : 3.x.x
release : xxx
```

**Step 5**

Re-deploy the old primary instance with the new ESC image.

Delete the old primary instance and redeploy it by using the new ESC package (bootvm.py and new registered image).
a) Log in to the new deployed instance and check ha status. The new instance should be in secondary state:
   `$ escadm status --v`

b) Run the following command to check the synchronization state of the new ESC secondary node:
   `# drbd-overview`
   Wait until the output of drbd-overview shown as UpToDate.

c) For the new ESC secondary node, make sure the health check is passed and the ESC version are upgraded correctly.

   `$ escadm status (check ha status)`
   Expected output:
   0 ESC status=0 ESC Master Healthy
   `$ esc_version (check esc version) version : 2.x.x`
   release : xxx
   `$ health.sh` Expected output:
   ESC HEALTH PASSED

   **Step 6** Go back in to the first upgraded primary instance and check the health and keepalived state.
   `$ drbd-overview`
   Expected output:
   1:esc/0 Connected Primary/Secondary UpToDate/UpToDate /opt/cisco/esc/esc_database ext4 2.9G 52M 2.7G 2%

   `$ escadm status (check ha status)`
   Expected output:
   0 ESC status=0 ESC Master Healthy

   `$ esc_version (check esc version) Expected output:
   version : 2.x.x`
   release : xxx

   `$ health.sh (check esc health)` Expected output:
   ESC HEALTH PASSED

   **Note** Quick rollback: In case of an upgrade failure, shutdown the upgraded instance and start the old primary instance to have a quick rollback. Run the following command in the old primary instance.
   `sudo bash preupgrade.sh --revoke`
   Then redeploy the upgraded instance with old esc version to have a full rollback.

   **Rollback Procedure for In-service Upgrade**
   1. Copy the database and log backup files to a location out of ESC VMs.
   2. Delete any remaining ESC instance and redeploy ESC HA VMs using qcow2 image with old version.
   3. Restore the database. Follow the procedures in the section, Upgrading ESC HA Instance with Backup and Restore for HA database restore.
   4. After database restore, you should have ESC service back with the old version.
In-Service Upgrade of the ESC HA Nodes in Kernel-Based Virtual Machine (KVM)

In-Service upgrade in KVM using ESC RPM packages

Use this procedure to upgrade ESC high-availability nodes with a minimum service interruption on a Kernel-based virtual machine.

Step 1
Backup ESC database and log files.

a) Perform ESC database backup from primary node. For more information on backing up the database, see Backup the Database from the ESC HA Instances.

b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

   $ escadm log collect

   Note: A timestamped log file will be generated in: /var/tmp/esc_log-<timestamp>.tar.bz2

c) Copy the database backup file and logs files (generated in /tmp/esc_log-.tar.bz2)* out of ESC VMs.

d) Copy the preupgrade.sh script and RPM files to both Primary ESC VM and Secondary ESC VMs. Execute the following command in both ESC VMs:

   $ sudo bash preupgrade.sh

   Expect output:
   Success

Step 2
Log into the ESC HA secondary VM and stop the ESC service.

$ sudo escadm stop

Step 3
Make sure the secondary ESC VM is in STOP state.

$ escadm status --v

If ESC status=0 esc ha is stopped.

Step 4
In secondary VM, execute the rpm command for upgrade:

$ sudo rpm -Uvh /home/admin/cisco-esc-<latest rpm filename>.rpm

Step 5
Log into the primary instance, set ESC primary node into maintenance mode.

$ escadm op_mode set --mode=maintenance

Make sure there are no in-flight transactions and no new transactions during the upgrade. From ESC 2.3, you may use following commands to check in-flight transactions.

$ escadm ip_trans

For any build older than ESC 2.3, you may need to check escmanager log for transactions at (/var/log/esc/escmanager.log).

Step 6
Power off ESC primary node and make sure it is completely shut down. In KVM ESC controller, execute the following commands:

$ virsh destroy <primary_vm_name>

$ virsh list --all
Step 7  Log in the upgraded ESC instance (previous secondary one), start the ESC service. The upgraded VM will take over primary role and provide ESC service.

$ sudo escadm restart
$ start esc_monitor

Step 8  Check the ESC version on the new primary instance to verify the upgraded version is correct. Once it is in the Primary state, make sure ESC service is running properly in the new Primary VM.

$ escadm status
Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version

$ health.sh
Expected output:
ESC HEALTH PASSED

Step 9  Power on the old primary instance. In KVM ESC controller, execute the following commands:

$ virsh start <primary_vm_name>

Step 10 Log into the VM which is still with old ESC version and repeat step 2, 3, 4, and 7 in the VM.

Note  For a Quick rollback: For a quick rollback to the previous version, you can just power off the upgraded primary instance. You can get the old ESC service back immediately. Run the following command in the old primary instance.

    sudo bash preupgrade.sh --revoke

You can then redeploy the secondary instance to fully rollback the HA.

---

**In-Service upgrade in KVM using ESC qcow2 Image**

**Step 1**  Backup ESC database and log files.

a) Perform ESC database backup from primary node. For more information on backing up the database, see [Backup the Database from the ESC HA Instances](#).

b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

    $ escadm log collect

Note  A timestamped log file will be generated in: /var/tmp/esc_log-<timestamp>.tar.bz2

c) Copy the database backup file and logs files (generated in /tmp/esc_log-.tar.bz2)* out of ESC VMs.

d) Copy the preupgrade.sh script to both Primary ESC VM and Secondary ESC VMs. Execute the following command in both ESC VMs:

    $ sudo bash preupgrade.sh

Expect output:
Success

**Step 2**  Redeploy secondary ESC instance. Register new ESC image on the secondary instance.
a) Delete the secondary instance through libvirt Virsh commands. On KVM host, run the following command:

```
$ virsh destroy the <secondary_vm_name>
$ virsh undefine --remove-all-storage <secondary_vm_name>
```

b) Copy the new ESC image into Kvm Host for redeployment usage:

```
sshpass -p 'host Password' scp /scratch/BUILD-2_x_x_x/BUILD-2_x_x_x/ESC-2_x_x_x.qcow2 root@HOSTIP:
```

c) Redeploy the secondary ESC VM instance based on newer image version. Re-install new the secondary instance by using the new ESC package (bootvm.py and new registered image). All other installation parameters should be the same as the former ESC VM deployment. For example, hostname, ip address, gateway_ip, ha_node_list, kad_vip, kad_vif have to use the same values. Once the new ESC instance with upgraded version is up, it will be in secondary state.

d) Log into the new instance and run the following command to check the synchronization state of the new ESC node.

```
$ drbd-overview
```

Wait until the output of `drbd-overview` show both nodes are "UpToDate" like the output below. It means the new ESC instance has completed the data synchronization from the primary instance.

```
esc/0 Connected Secondary/Primary UpToDate/UpToDate
```

### Step 3

Stop keepalived service on Secondary instance, Power off primary instance, and then start Secondary keepalived service.

a) Log into the primary instance, set ESC primary node into maintenance mode.

```
$ escadm op_mode set --mode=maintenance
```

Make sure there is no in-flight transaction ongoing before moving to the next step. To verify there are no in-flight transactions, use the following command:

For ESC 2.3:

```
$ escadm ip_trans
```

For versions older than ESC 2.3, check escmanager log at (/var/log/esc/escmanager.log) and make sure there are no new transaction in escmanager log.

b) Log in to the upgraded secondary instance and shut down the keepalived service.

```
$ sudo escadm stop
```

c) Power off the primary instance and make sure it has been completely turned off. In KVM ESC Controller, run:

```
$ virsh destroy <primary_vm_name>
$ virsh list --all
```

d) Log into the previously upgraded secondary instance which is in stopped state and start the ESC service. The secondary ESC instance will take the primary role (switchover will be triggered) and start providing services with new version.

```
$ sudo escadm restart
```

### Step 4

Check the ESC version on the new primary instance to verify the version is upgraded correctly.

```
$ escadm status (check ha status)
```

Expected output:

```
0 ESC status=0 ESC Master Healthy
```

```
$ esc_version (check esc version)
version : 3.x.x
release : xxx
```

```
$ health.sh (check esc health)
```

Expected output:

```
ESC HEALTH PASSED
```
Step 5 Re-deploy the old primary instance with the new ESC image. Delete the old primary instance and redeploy it by using the new ESC package (bootvm.py and new registered image). All other installation parameters should be the same as the old ESC VM deployment. For example, hostname, ip address, gateway_ip, ha_node_list, kad_vip, kad_vif have to be the same values.

a) Log in to the new deployed instance and check ha status. The new instance should be in secondary state:

   $ escadm status

b) Run the following command to check the synchronization state of the new ESC secondary node:

   $ drbd-overview
   Wait until the output of drbd-overview shown as UpToDate.

c) For the new ESC secondary node, make sure the health check is passed and the ESC version are upgraded correctly.

   $ escadm status (check ha status)
   Expected output:
   0 ESC status=0 ESC Master Healthy
   $ esc_version (check esc version)
   version : 2.x.x
   release : xxx
   $ health.sh
   Expected output:
   ESC HEALTH PASSED

Step 6 Go back in to the first upgraded primary instance and check the health and keepalived state.

   $ drbd-overview
   Expected output:
   1:esc/0 Connected Primary/Secondary UpToDate/UpToDate /opt/cisco/esc/esc_database ext4 2.9G 52M 2.7G 2%

   $ escadm status (check ha status)
   Expected output:
   0 ESC status=0 ESC Master Healthy

   $ esc_version (check esc version)
   Expected output:
   version : 2.x.x
   release : xxx

   $ health.sh (check esc health)
   Expected output:
   ESC HEALTH PASSED

Note Quick rollback: In case of an upgrade failure, shutdown the upgraded instance and start the old primary instance to have a quick rollback. Run the following command in the old primary instance.

   sudo bash preupgrade.sh --revoke

   Then redeploy the upgraded instance with old esc version to have a full rollback.

Rollback Procedure for In-service Upgrade

1 Copy the database and log backup files to a location out of ESC VMs.
2 Delete any remaining ESC instance and redeploy ESC HA VMs using qcow2 image with old version.
3 Restore the database. Follow the procedures in the section, Upgrading ESC HA Instance with Backup and Restore for HA database restore.
4 After database restore, you should have ESC service back with the old version.
In-Service Upgrade of the ESC HA Nodes in VMware

**In-Service upgrade in VMware using ESC RPM packages**

Use this procedure to upgrade the ESC high-availability nodes one node at a time with a minimum service interruption. This process leverages the ESC HA replication and failover capability to smoothly move ESC service to the new upgraded node without the manual database restore.

**Step 1**
Backup ESC database and log files.

a) Perform ESC database backup from primary node. For more information on backing up the database, see Backup the Database from the ESC HA Instances.

b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

   ```
   $ escadm log collect
   ```

c) Copy the database backup file and logs files (generated in /tmp/esc_log-.tar.bz2)* out of ESC VMs.

d) Copy the preupgrade.sh script and RPM files to both Primary ESC VM and Secondary ESC VMs. Execute the following command in both ESC VMs:

   ```
   $ sudo bash preupgrade.sh
   ```

   Expect output:

   ```
   Success
   ```

**Step 2**
Log into the ESC HA secondary VM and stop the keepalived service.

   ```
   $ sudo escadm stop
   ```

**Step 3**
Make sure the secondary ESC VM is in STOP state.

   ```
   $ escadm status --v
   ```

   If ESC status=0 esc ha is stopped.

**Step 4**
In secondary VM, execute the rpm command for upgrade:

   ```
   $ sudo rpm -Uvh /home/admin/cisco-esc-2.2.9-50.rpm
   ```

**Step 5**
Log into the primary instance, set ESC primary node into maintenance mode.

   ```
   $ escadm op_mode set --mode=maintenance
   ```

   Make sure there are no in-flight transactions and no new transactions during the upgrade. From ESC 2.3, you may use following commands to check in-flight transactions.

   ```
   $ escadm ip trans
   ```

   For build older than ESC 2.3, you may need to check escmanager log and make sure no new transactions are recorded in this log file. The log file can be located at (/var/log/esc/escmanager.log).

**Step 6**
Power off ESC primary node. In VMware vSphere Client, select **Home > Inventory > VMs and Templates**, right click the primary instance name from the left panel, and select **Power > Power Off**.

**Step 7**
Log in to the upgraded ESC instance (previous secondary one), and start the keepalived service. The upgraded VM will take over primary role and provide ESC service.

   ```
   $ sudo escamd restart
   ```

**Step 8**
Check the ESC version on the new primary instance to verify the upgraded version is correct. Once it is in the Primary state, make sure ESC service is running properly in the new Primary VM.

   ```
   $ escadm status
   ```
In-Service Upgrade of the ESC HA Nodes in VMware

Step 9  Power on the old primary instance. In VMware vSphere Client, select Home > Inventory > VMs and Templates, right click the primary instance name from the left panel, then select Power > Power On.

Step 10  Log into the VM which is still with old ESC version and repeat step 2, 3, 4, and 7 in the VM.

Note  For a Quick rollback: For a quick rollback to the previous version, you can just power off the upgraded primary instance. You can get the old ESC service back immediately. Run the following command in the old primary instance.

```
sudo bash preupgrade.sh --revoke
```

You can then redeploy the secondary instance to fully rollback the HA.

In-Service upgrade in VMware using ESC qcow2 Image

Step 1  Backup ESC database and log files.

a) Perform ESC database backup from primary node. For more information on backing up the database, see Backup the Database from the ESC HA Instances.

b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

```
$ escadm log collect
```

Note  A timestamped log file will be generated in: /var/tmp/esc_log-<timestamp>.tar.bz2

c) Copy the database backup file and logs files (generated in /tmp/esc_log-<timestamp>*.tar.gz) out of ESC VMs.

d) Copy the preupgrade.sh script to both Primary ESC VM and Secondary ESC VMs. Execute the following command in both ESC VMs:

```
$ sudo bash preupgrade.sh
```

Expect output:
Success

Step 2  Redeploy secondary ESC instance. Register new ESC image on the secondary instance, and wait for the data to be synchronized.

a) Delete the secondary instance. To delete the secondary ESC instance, you need to first "Power Off" the instance through vSphere Client and then use the Delete from Disk option. In VMware vSphere Client, select Home > Inventory > VMs and Templates, right click the instance name from the left panel, then select Power > Power Off. Now to delete the secondary instance, select Home > Inventory > VMs and Templates, right click the instance name from the left panel, then select Delete from Disk.

b) Redeploy the secondary ESC VM instance based on newer image version. Re-install new the secondary instance by using the new ESC package (bootvm.py and new registered image). Once the new ESC instance with upgraded version is up, it will be in secondary state.
c) Log into the new instance and run the following command to check the synchronization state of the new ESC node.

```
$ drbd-overview
```

Wait until the output of drbd-overview show both nodes are "UpToDate" like the output below. It means the new ESC instance has completed the data synchronization from the primary instance.

```
esc/0 Connected Secondary/Primary UpToDate/UpToDate
```

**Step 3**

Stop keepalived service on Secondary instance, Power off primary instance, and then start Secondary keepalived service.

a) Log into the primary instance, set ESC primary node into maintenance mode.

```
$ escadm op_mode set --mode=maintenance
```

Make sure there is no in-flight transaction ongoing before moving to the next step. To verify there are no in-flight transactions, use the following command:

```
For ESC 2.3:
$ escadm ip_trans
```

For versions older than ESC 2.3, check escmanager log at (/var/log/esc/escmanager.log) and make sure there are no new transaction in escmanager log.

b) Log in to the upgraded secondary instance and shut down the keepalived service.

```
$ sudo escadm stop
```

c) Power off the primary instance and make sure the primary instance has been powered off. In VMware vSphere Client, select Home > Inventory > VMs and Templates, right click the instance name from the left panel, then select Power > Power Off.

d) Log into the previously upgraded secondary instance which is in stopped state and start the keepalived service. The secondary ESC instance will take the primary role (switchover will be triggered) and start providing services with new version.

```
$ sudo escadm start
```

**Step 4**

Check the ESC version on the new primary instance to verify the version is upgraded correctly.

```
$ escadm status --v(check ha status)
```

Expected output:

```
0 ESC status=0 ESC Master Healthy
```

```
$ esc_version (check esc version)
version : 3.x.x
release : xxx
```

```
$ health.sh (check esc health)
```

Expected output:

```
ESC HEALTH PASSED
```

**Step 5**

Re-deploy the old primary instance with the new ESC image.

Delete the old primary instance and redeploy it by using the new ESC package (bootvm.py and new registered image). All other installation parameters should be the same as the old ESC VM deployment. For example, hostname, ip address, gateway_ip, ha_node_list, kad_vip, kad_vif have to be the same values. To delete, in the VMware vSphere Client, access, Home > Inventory > VMs and Templates, right click the instance name from the left panel, then select Delete from Disk.

a) Log in to the new deployed instance and check ha status. The new instance should be in secondary state:

```
$ escadm status
```

b) Run the following command to check the synchronization state of the new ESC secondary node:

```
$ drbd-overview
```
Wait until the output of `drbd-overview` shown as UpToDate.

c) For the new ESC secondary node, make sure the health check is passed and the ESC version are upgraded correctly.

```
$ escadm status (check ha status)
Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version (check esc version)
version : 3.x.x
release : xxx

$ health.sh
Expected output:
ESC HEALTH PASSED
```

**Step 6**  
Go back in to the first upgraded primary instance and check the health and keepalived state.

```
$ drbd-overview
Expected output:
1:esc/0 Connected Primary/Secondary UpToDate/UpToDate /opt/cisco/esc/esc_database ext4 2.9G 52M 2.7G 2%

$ escadm status (check ha status)
Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version (check esc version)
Expected output:
version : 3.x.x
release : xxx

$ health.sh (check esc health)
Expected output:
ESC HEALTH PASSED
```

**Note**  
Quick rollback: In case of an upgrade failure, shutdown the upgraded instance and start the old primary instance to have a quick rollback. Run the following command in the old primary instance.

```
sudo bash preupgrade.sh --revoke
```

Then redeploy the upgraded instance with old esc version to have a full rollback.

**Rollback Procedure for In-service Upgrade**

1. Copy the database and log backup files to a location out of ESC VMs.
2. Delete any remaining ESC instance and redeploy ESC HA VMs using qcow2 image with old version.
3. Restore the database. Follow the procedures in the section, Upgrading ESC HA Instance with Backup and Restore for HA database restore.
4. After database restore, you should have ESC service back with the old version.
Cisco Elastic Services Controller Installer Arguments

You need to specify the following *bootvm.py* script arguments to boot ESC instances.

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>esc_hostname</td>
<td>Specifies the host name of the ESC VM instance.</td>
</tr>
<tr>
<td>--image</td>
<td>Specifies the image id used in the OpenStack glance to boot up the ESC instance.</td>
</tr>
<tr>
<td>--boot_volume</td>
<td>Specify the volume name or id of the external bootable volume from where you want to launch ESC instance.</td>
</tr>
<tr>
<td>--net</td>
<td>Specifies the Network IDs or names in OpenStack that ESC connects to.</td>
</tr>
<tr>
<td>--ipaddr</td>
<td>(Optional) Specifies the IP addresses that ESC will be assigned in the network.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> The IP address must correspond to the net_id in the --net argument.</td>
</tr>
<tr>
<td>--gateway_ip</td>
<td>(Optional) Specifies the default gateway IP address of ESC.</td>
</tr>
<tr>
<td>--os_auth_url</td>
<td>(Optional) Specifies the OpenStack keystone url used by os_auth_url for authentication.</td>
</tr>
<tr>
<td>--os_username</td>
<td>(Optional) Specifies the OpenStack keystone username used by os_username for authentication.</td>
</tr>
<tr>
<td>--os_password</td>
<td>(Optional) Specifies the OpenStack keystone password used by os_password for authentication.</td>
</tr>
<tr>
<td>--os_tenant_name</td>
<td>(Optional) Specifies the OpenStack tenant name used by os_tenant_name for ESC deployment.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>--bs_os_auth_url</code></td>
<td>(Optional) Specifies the OpenStack keystone url used by bs_os_auth_url for authentication.</td>
</tr>
<tr>
<td><code>--bs_os_username</code></td>
<td>(Optional) Specifies the OpenStack keystone username used by bs_os_username for authentication.</td>
</tr>
<tr>
<td><code>--bs_os_password</code></td>
<td>(Optional) Specifies the OpenStack keystone password used by bs_os_password for authentication.</td>
</tr>
<tr>
<td><code>--bs_os_tenant_name</code></td>
<td>(Optional) Specifies the OpenStack tenant name used by bs_os_tenant_name for ESC deployment.</td>
</tr>
<tr>
<td><code>--flavor</code></td>
<td>(Optional) Specifies the OpenStack flavor id to boot the ESC VM.</td>
</tr>
<tr>
<td><code>--security_rules_file</code></td>
<td>(Optional) Specifies the file to define security rules (IP, Port security) for ESC VM.</td>
</tr>
<tr>
<td><code>--etc_hosts_file</code></td>
<td>(Optional) Specifies the file for adding more entries to the ESC vm's hosts file (/etc/hosts).</td>
</tr>
<tr>
<td><code>--avail_zone</code></td>
<td>(Optional) Specifies the OpenStack zone used for ESC deployment.</td>
</tr>
<tr>
<td><code>--esc_params_file</code></td>
<td>(Optional) Specifies the default parameter file for ESC deployment.</td>
</tr>
<tr>
<td><code>--db_volume_id</code></td>
<td>(Optional) Specifies the cinder volume id to mount for database storage in ESC HA [ESC-HA].</td>
</tr>
<tr>
<td><code>--ha_node_list</code></td>
<td>(Optional) Specifies list of IP addresses for HA nodes in the Primary/Standby cluster. For ESC nodes with multiple network interfaces, these IPs should be the addresses in the network used for data synchronization. <strong>Note</strong> This argument is utilized for replication-based HA solution only.</td>
</tr>
<tr>
<td><code>--kad_vip</code></td>
<td>(Optional) Specifies the IP address for Keepalived VIP (virtual IP) plus the interface of Keepalived VIP [ESC-HA]. An example format for specifying the interface of VIP is --kad_vip 192.0.2.1:eth2 or --kad_vip [2001:cc0:2020::fc]:eth2</td>
</tr>
<tr>
<td><code>--kad_vif</code></td>
<td>(Optional) Specifies the interface for Keepalived virtual IP and keepalived VRRP [ESC-HA]. You can also use this argument to only specify the interface for Keepalived VRRP, if the VIP interface is already specified using the kad_vip argument.</td>
</tr>
<tr>
<td><code>--kad_vri</code></td>
<td>Specified the virtual router id of vrrp instance. Accepted values for kad_vri are 0 to 254. ESC VMs in the same HA should use the same kad_vri number. If kad_vip is not used for L3 HA, the kad_vir has to be used, otherwise, you can skip kad_vri argument.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>--bgp_remote_ip</td>
<td>(Optional) Specifies the bgp remote ip, should be the ip of ASR/BGP Router [ESC-HA]</td>
</tr>
<tr>
<td>--bgp_local_as</td>
<td>(Optional) Specifies the BGP AS number for routing configuration [ESC-HA].</td>
</tr>
<tr>
<td>--bgp_remote_as</td>
<td>(Optional) Specifies the BGP AS number routing configuration and this number is configured in BGP Router [ESC-HA].</td>
</tr>
<tr>
<td>--bgp_local_router_id</td>
<td>(Optional) Specifies the BGP routing configuration [ESC-HA]. Note: The BGP routing configuration must be unique for the ESC.</td>
</tr>
<tr>
<td>--bgp_local_ip</td>
<td>(Optional) Specifies the IP of the interface for BGP anycast routing [ESC-HA].</td>
</tr>
<tr>
<td>--bgp_md5</td>
<td>(Optional) Specifies the md5 code for BGP broadcasting [ESC-HA].</td>
</tr>
<tr>
<td>--route</td>
<td>Specifies the routing configuration for ESC VM.</td>
</tr>
<tr>
<td>--ntp_server</td>
<td>(Optional) Specifies the NTP server address.</td>
</tr>
<tr>
<td>--rsyslog_server</td>
<td>(Optional) Specifies the IP address of rsyslog server that ESC sends the log to</td>
</tr>
<tr>
<td>--rsyslog_server_port</td>
<td>(Optional) Specifies the port of rsyslog server that ESC sends the log to.</td>
</tr>
<tr>
<td>--rsyslog_server_protocol</td>
<td>(Optional) Specifies the protocol to be used by the ESC to forward logs to the server.</td>
</tr>
<tr>
<td>--secure</td>
<td>(Optional) Enables secure configuration. You can specify the following values:</td>
</tr>
<tr>
<td></td>
<td>• A—Root is completely locked out. You cannot login as a root even from the console.</td>
</tr>
<tr>
<td></td>
<td>• B—SELinux runs in the enforcing mode.</td>
</tr>
<tr>
<td></td>
<td>• C—IPv4/IPv6 tables are started.</td>
</tr>
<tr>
<td></td>
<td>• D—SSH password authentication is disabled. You need the private key to ssh into ESC vm.</td>
</tr>
<tr>
<td></td>
<td>• E—host keys for confd will be re-created.</td>
</tr>
<tr>
<td>--host_mapping_file</td>
<td>(Optional) Specifies the host mapping file for VNF deployment.</td>
</tr>
<tr>
<td>--version</td>
<td>(Optional) Prints the version of bootvm.py and exits.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>--user_pass</td>
<td>This along with --user_conf_pass are mandatory arguments from 3.0 onwards.</td>
</tr>
<tr>
<td></td>
<td>This argument adds a user to access the ESC VM. Use this argument to specify a user without administrative privileges, i.e, a non-admin/non-root user. Use the following format: user_name:password. The bootvm.py command requires at least one --user_pass argument to create an admin account for linux (ssh/console access). The following is the syntax for the mandatory user credential argument:</td>
</tr>
<tr>
<td></td>
<td>--user_pass argv: admin:'PASSWORD-OR-HASH'[:OPTIONAL-PUBLIC-KEY-FILE][:OPTIONAL-ROLE]</td>
</tr>
<tr>
<td></td>
<td>This user can only do the following:</td>
</tr>
<tr>
<td></td>
<td>• Login to ESC through SSH.</td>
</tr>
<tr>
<td></td>
<td>• Access and drive the Netconf CLI, such as, esc_ne_cli, netconf-console, and so on.</td>
</tr>
<tr>
<td></td>
<td>• Read ESC-related logs from /var/logs/esc</td>
</tr>
<tr>
<td></td>
<td>• Access REST interface through localhost</td>
</tr>
<tr>
<td></td>
<td>This user cannot:</td>
</tr>
<tr>
<td></td>
<td>• Access the ESC DB and reconfigure ESC system.</td>
</tr>
<tr>
<td></td>
<td>• Access the system-level logs</td>
</tr>
<tr>
<td></td>
<td>• Configure the system level components, such as: Rsyslog, Keepalived, DRDB, and so on.</td>
</tr>
<tr>
<td></td>
<td>• Access the encryption keys and values from REST interface or ESC logs.</td>
</tr>
<tr>
<td></td>
<td>Following is an example of --user_pass for admin account and stronger clear text passwords. Use single quotes to avoid conflict with shell reserved characters:</td>
</tr>
<tr>
<td></td>
<td>--user_pass admin:'Strong4Security!'.</td>
</tr>
<tr>
<td></td>
<td>Another example to install ESC using a password hash for both admin accounts. Use single quotes to avoid conflict with shell reserved characters:</td>
</tr>
<tr>
<td></td>
<td>--user_pass admin:'$algorithm$salt$hash-of-salt-password'.</td>
</tr>
<tr>
<td></td>
<td>ESC 2.1 and later, accepts the public key for this attribute. For example, the following will generate 'admin321' as the password for user 'admin' and use /tmp/abc.pub as the key file to inject the public key for it:</td>
</tr>
<tr>
<td></td>
<td>--user_pass argv: admin:admin321:/tmp/abc.pub</td>
</tr>
<tr>
<td>Arguments</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| --user_confd_pass         | Used to change confd users. The bootvm.py command requires at least one --user_confd_pass to create an admin account for ConfD (netconf/cli access). The following is the syntax for the mandatory user credential argument:  

    --user_confd_pass  
    admin:'PASSWORD-OR-HASH'[:OPTIONAL-PUBLIC-KEY-FILE]

Following is an example of --user_confd_pass for admin account and stronger clear text passwords. Use single quotes to avoid conflict with shell reserved characters:  

    --user_confd_pass:'Strong4Security!' .

Another example, to install ESC using a password hash for both admin accounts. Use single quotes to avoid conflict with shell reserved characters:  

    --user_confd_pass:'$algorithm$salt$hash-of-salt-password'.

ESC 2.1 and later, accepts the public key for this attribute. For example, the following will generate 'admin321' as the password for user 'admin' and use /tmp/abc.pub as the key file to inject the public key for it: --user_confd_pass:admin321:/tmp/abc.pub |

| --esc_portal_startup      | (Optional) Starts the ESC portal.                                                                                                                                                                                                                                                                                                      |
| --log                     | (Optional) Specifies the log file. By default, logs to stdout.                                                                                                                                                                                                             |
| --esc_monitor_check_ips   | (Optional) Specifies the IP addresses that must be monitored by esc_monitor (for HA failover).                                                                                                                                                                             |
| --enable-https-rest       | (Optional) Enables a secure REST Interface for the created ESC VM.                                                                                                                                                                                                        |
| --enable-http-rest        | (Optional) Enables an unsecured REST Interface for the created ESC VM.                                                                                                                                                                                                   |
| --ha_mode                 | Specifies the ESC HA mode for HA installation. Specify one of the following available options for HA: **no_ha**: No HA, **cinder**: Shared Cinder Volume, **drbd**: Built-in DRBD, **drbd_on_cinder**: DRBD over Cinder Volume  | |
| --encrypt_key             | Specifies the key for encryption                                                                                                                                                                                                                                                                                                     |
| --proxy                   | Uses the proxy on a given port.                                                                                                                                                                                                                                                                                                      |
| --noproxy                 | Lists the hosts which do not use proxy.                                                                                                                                                                                                                                                                                             |
| --kad_unicast_src_ip      | Specifies the source IP address of unicast. Should be the IP address of interface that ESC VM uses for unicast (L3) VRRP communication. Example: --kad_unicast_src_ip 10.20.0.1                                                                                     |
## Arguments

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--kad_unicast_peer</td>
<td>Specified the peer IP addresses of unicast. Should be the ip address of interface that ESC peer VM uses for unicast (L3) VRRP communication. Example: --kad_unicast_peer 10.20.0.1</td>
</tr>
<tr>
<td>--placement_hint</td>
<td>Use this argument to specify the placement of ESC HA virtual machines using the server group, samehost, differenthost filters. Example:</td>
</tr>
<tr>
<td></td>
<td>• --placement_hint different_host=2b299428-e7a7-4528-8566-9a4970183c6a [ID should be the VM uuid]</td>
</tr>
<tr>
<td></td>
<td>• --placement_hint same_host=2b299428-e7a7-4528-8566-9a4970183c6a [ID should be the VM uuid]</td>
</tr>
<tr>
<td></td>
<td>• --placement_hint group=4c7758ab-e9cb-4cf0-8f02-344ec666365b [ID should be the server group uuid]</td>
</tr>
<tr>
<td>--format {json}</td>
<td>Use this argument to capture the success and failure message in the output. Example: $ ./bootvm.py --image ESC-2_3_0_8 --net esc-net --format json --test-0</td>
</tr>
<tr>
<td></td>
<td>{ &quot;status&quot; : &quot;Success&quot; , &quot;vm_uuid&quot; : &quot;UUID&quot; }</td>
</tr>
<tr>
<td>--user_rest_pass</td>
<td>Adds a user to access the Rest API. Format is username: password. This option can be repeated.</td>
</tr>
<tr>
<td>--user_portal_pass</td>
<td>Add a portal user. Format username: password. This option can be repeated.</td>
</tr>
<tr>
<td>--no_vim_credentials</td>
<td>Use this argument to deploy ESC without passing VIM credential. If this argument is used, following parameters will not be passed during the installation:</td>
</tr>
<tr>
<td></td>
<td>• --os_auth_url</td>
</tr>
<tr>
<td></td>
<td>• --os_username</td>
</tr>
<tr>
<td></td>
<td>• --os_password</td>
</tr>
<tr>
<td></td>
<td>• --os_tenant_name</td>
</tr>
<tr>
<td></td>
<td>After the deployment is complete, the user can set these VIM credential through ESC's VIM/VIM User APIs (REST/Netconf). For more information on configuring through REST APIs and Netconf, see Configuring VIM credentials after installing ESC in the Post Installation Tasks chapter.</td>
</tr>
</tbody>
</table>
Table 4:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>security_rules_file</td>
<td>The file contains the following:</td>
</tr>
<tr>
<td></td>
<td>• Security rules to create a security group for the tenant.</td>
</tr>
<tr>
<td></td>
<td>• Configurations to allow traffic for the tenant.</td>
</tr>
<tr>
<td>etc_hosts_file</td>
<td>The file contains one or more entries that you want to include in the /etc/hosts file.</td>
</tr>
<tr>
<td>esc_params_file</td>
<td>The file contains information to configure various parameters of ESC. For details on parameters that can be configured in the esc_params_file are described in table below.</td>
</tr>
<tr>
<td>host_mapping_file</td>
<td>The file contains information to map a network based on the hosts.</td>
</tr>
</tbody>
</table>

ESC Configuration Parameters

Using this file, you can configure various ESC parameters during the installation. The parameters that can be configured are shown in the table.

Below is an example configuration using this file:

```
openstack.endpoint=adminURL
affinity.filter=ServerGroupAffinity
```

Table 5: ESC Configuration Parameters

<table>
<thead>
<tr>
<th>esc_param.conf</th>
<th>Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default_vm_recovery_retries_max</td>
<td>Int</td>
<td>3</td>
<td>Number of recovery attempts allowed per VM.</td>
</tr>
</tbody>
</table>
**ESC Configuration Parameters**

<table>
<thead>
<tr>
<th><code>esc_param.conf</code></th>
<th>Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>openstack.endpoint</td>
<td>String</td>
<td>adminURL</td>
<td>The parameter to set up the keystone endpoint value of ESC. Options: adminURL, publicURL</td>
</tr>
<tr>
<td>log.level</td>
<td>String</td>
<td>INFO</td>
<td>Level of logging. Options: INFO, Trace, DEBUG</td>
</tr>
<tr>
<td>affinity.filter</td>
<td>String</td>
<td>SameHostFilter</td>
<td>A constant string used to build PolicyEngine and initializing VM policy table. Options: SameHostFilter, ServerGroupAffinity</td>
</tr>
<tr>
<td>anti_affinity.filter</td>
<td>String</td>
<td>DifferentHostFilter</td>
<td>A constant string used to build PolicyEngine and initializing VM policy table. Options: DifferentHostFilter</td>
</tr>
</tbody>
</table>

**Note**
ESC uses SameHostFilter and DifferentHostFilter for ESC policy engine by default but OpenStack may not configure those filters by default. You may need to add SameHostFilter and DifferentHostFilter to the following scheduler options in the `/etc/nova/nova.conf` file of the nova service in your OpenStack.

`scheduler_default_filters = RetryFilter, AvailabilityZoneFilter, RamFilter, ComputeFilter, ComputeCapabilitiesFilter, ImagePropertiesFilter, ServerGroupAntiAffinityFilter, ServerGroupAffinityFilter, DifferentHostFilter, SameHostFilter`
ESC System Logs

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-af21-084e3b029a0] [msg=sleepingfor5seconds to allow vm to become ACTIVE instance id: 162344f7-78f9-4e45-9f23-34cf87377fa7 name:test-dep_test_0_dc3f406c-05ca-43b3-af21-084e3b029a0]```

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:

<table>
<thead>
<tr>
<th>NBI</th>
<th>&quot;com.cisco.esc.rest&quot;&quot;com.cisco.esc.filter&quot; (North Bound Interface - Clientinterface)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBI</td>
<td>&quot;com.cisco.esc.rest&quot; - source is a callback handler or&quot;EventsResource&quot; (South Bound Interface - i.e. between ESC and the VIM)</td>
</tr>
<tr>
<td>SM</td>
<td>&quot;com.cisco.esc.statemachines&quot;. stands for StateMachine. This classification indicates logs in the StateMachine category.</td>
</tr>
<tr>
<td>MONITORING</td>
<td>&quot;com.cisco.esc.monitoring&quot;&quot;com.cisco.esc.paadaptor&quot; (MONA related logs)</td>
</tr>
<tr>
<td>DYNAMIC_MAPPING</td>
<td>&quot;com.cisco.esc.dynamicmapping&quot;&quot;com.cisco.esc.db.dynamicmapping&quot; (MONA related logs)</td>
</tr>
<tr>
<td>CONFD</td>
<td>&quot;com.cisco.esc.confda&quot;</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Tenant</th>
<th>[tenant:&lt;tenant name&gt;]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>[tenant:&lt;tenant id&gt;, network:&lt;network name&gt;]</td>
</tr>
</tbody>
</table>

**Note**
The tenant appears only if applicable.
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:
162344f7-78f9-4e45-9f23-34cf873771fa? 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)

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:

```plaintext
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 {
```
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 filter-log log.xml

The sample log.xml is as follows:

```xml
<filterLog xmlns="http://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:

```xml
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <filterLogResults xmlns="http://www.cisco.com/esc/esc">
  </filterLogResults>
</rpc-reply>
```
<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.