CONTENTS

CHAPTER 1

Preface 1

Purpose 1

Audience 1

Terms and Definitions 1

Obtaining Documentation Request 2

CHAPTER 2

Elastic Services Controller Overview 5

Key Features of Elastic Services Controller 5

ESC Architecture 6

CHAPTER 3

Elastic Services Controller Interfaces 9

Elastic Services Controller NB APIs 9

NETCONF/YANG Northbound API 9

REST Northbound API 11

Cisco Elastic Services Controller Portal 12

Logging into the Web Portal 12

Getting Started with ESC Portal 13

Cisco Elastic Services Controller Portal Dashboard 17

Resources 17

Deployments 17

Infrastructure 17

Incoming Requests 17

Notifications 17

Systems 18

CHAPTER 4

Virtual Network Functions Life Cycle Management 19

Deploying VNFs in the OpenStack Environment 20

Deploying VNFs in the VMware Environment 21
CHAPTER 5

Managing VM Resources in ESC

Managing Tenants

- Adding Tenants Using Northbound APIs

Managing Networks

- OpenStack Network
- VMware Network

- Adding Networks in ESC Portal (VMware Only)
- Deleting Networks in ESC Portal
- Adding Networks Using Northbound APIs

Managing Subnets

- Adding Subnet Definitions Using Northbound APIs

Managing Flavors

- Adding Flavors Using Northbound APIs

Managing Images

- Adding Images in ESC Portal (VMware Only)
- Creating Image from a Form
- Deleting Images in ESC Portal
- Adding Images Using Northbound APIs

Managing Volumes (OpenStack Only)

CHAPTER 6

Deploying Virtual Network Functions

Deploying Virtual Network Functions in the OpenStack Environment

Deploying Virtual Network Functions in the VMware Environment

- Multi-Virtual Data Center Support (VMware Only)
- Passing OVF Properties to a VM

Deploying Virtual Network Functions in ESC Portal (VMware Only)

- Deploy Using a File (deployment datamodel)
- Deploying Using a Form

Day-0 Configuration

Unified Deployment Request

Applying Affinity and Anti-affinity Rules

- Affinity and Anti-Affinity Rules in the OpenStack Environment
- Intra Group Affinity Policy
Preface

• Purpose, page 1

Purpose

This guide will help you to perform tasks such as deploying, monitoring, scaling and healing of VNFs. To setup, provision, and configure Elastic Services Controller (ESC), see the Cisco Elastic Services Install and Upgrade Guide.

Audience

This guide is designed for network administrators responsible for provisioning, configuring, and monitoring VNFs. Elastic Services Controller (ESC) and its VNFs are deployed in a Virtual Infrastructure Manager (VIM). Currently, OpenStack (two platforms with 3 versions) vCenter (5.5 and 6.0) are the supported VIMs. The administrator must be familiar with the VIM layer, vCenter and OpenStack resources, and the commands used.

Cisco ESC is targeted for Service Providers (SPs) and Small-to-Medium Businesses (SMBs). ESC helps SPs reduce cost of operating the networks by providing effective and optimal resource usage. For SMBs ESC automates provisioning, configuring and monitoring of network functions. Small-to-medium businesses can benefit by leveraging some of the basic services such as DNS, DHCP and NTP.

Terms and Definitions

The below table defines the terms often used in this guide.

Table 1: Terms and Definitions

<table>
<thead>
<tr>
<th>Terms</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC</td>
<td>Elastic Services Controller (ESC) is a Virtual Network Function Manager (VNFM), performing life cycle management of Virtual Network Functions.</td>
</tr>
<tr>
<td>Terms</td>
<td>Definitions</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>HA</td>
<td>ESC High Availability (HA) is a solution for preventing single points of ESC failure and achieving minimum ESC shutdown time.</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>
<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>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>Virtual Managed Services (vMS) solution shifts the deployment of managed services away from the manual configuration of the latest network devices to the creation of a software abstraction to represent the service definition.</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) and it performs life cycle management of Virtual Network Functions (VNFs). ESC provides agentless and multi vendor VNF management by provisioning the virtual services and monitoring their health and promotes agility, flexibility, and programmability in Network Function Virtualization (NFV) environments. It provides the flexibility to define rules for monitoring, and associate actions that are triggered based on the outcome of these rules. Based on the monitoring results, ESC performs scale in or scale out operations on the VNFs. In the event of a VM failure ESC also supports automatic VM recovery.

ESC fully integrates with Cisco and other third party applications. As a standalone product, the ESC can be deployed as a VNF Manager. ESC integrates with Cisco Network Services Orchestrator (NSO) to provide VNF management along with orchestration. ESC as a VNF Manager targets the virtually managed services and all service provider NFV deployments such as virtual video, WiFi, authentication and so on. Complex services include multiple VMs that are orchestrated as a single service with dependencies between them. These multiple VMs are managed as a single entity, such as, Virtually Managed Services (vMS) 1.0 and later.

- Key Features of Elastic Services Controller, page 5
- ESC Architecture, page 6

Key Features of Elastic Services Controller

- Provides open and modular architecture, which allows multi-vendor OSS, VNF and VIM support.
- Provides end-to-end dynamic provisioning and monitoring of virtualized services using a single point of configuration.
- Provides customization across different phases of life cycle management; while monitoring the VM, service advertisement, and custom actions.
- Provides agentless monitoring with an integrated Monitoring Actions (MONA) engine. The monitoring engine provides simple and complex rules, to decide scale in and scale out of VMs.
- Provides scale in and scale out options based on the load of the network.
- Deploys or removes VMs based on the monitoring errors and threshold conditions detected as part of healing (also called as recovery).
• Supports service agility by providing faster VNF deployment and life cycle management.
• Supports multi-tenant environments.
• Supports REST and NETCONF / YANG interfaces to provide better hierarchical configuration and data modularity.

ESC Architecture

Cisco Elastic Services Controller (ESC) is built as an open and modular architecture, that allows OSS, and multi-vendor support. It performs life cycle management of the VNFs, that is, VNF onboarding, configuring the VNFs, monitoring them, and making VNF level life cycle decisions such as healing and scaling based on the KPI requirements. ESC and its managed VNFs are deployed as VMs running within a Virtual Infrastructure Manager (VIM). Currently, OpenStack and vCenter 6.0 are the supported VIMs. The ESC core engine manages transactions, validations, policies, workflows, VM state machines and rollbacks. The monitoring and actions service engine in ESC performs monitoring based on several monitoring methods. Events are triggered based on the monitoring actions. The monitoring engine also supports custom monitoring plugins.

ESC can be configured for High Availability. For details, see the Cisco Elastic Services Controller Install and Upgrade Guide.

ESC interacts with the top orchestration layer using the REST and NETCONF/YANG NB APIs. The orchestration layer can be a Cisco NSO or any third party OSS. Confd enables integration with NSO by adding NETCONF/YANG northbound interface support. A configuration template, Virtual Network Function Descriptor (VNFD) file is used to describe the deployment parameters and operational behaviors of the VNFs.
The VNFD file is used in the process of onboarding a VNF and managing the life cycle of a VNF instance. Figure 1: Cisco Elastic Services Controller Architecture represents the Elastic Services Controller architecture.

Figure 1: Cisco Elastic Services Controller Architecture
Elastic Services Controller Interfaces

Cisco Elastic Services Controller (ESC) can be deployed in one of the following ways:

- As part of the Cisco Orchestration suite—ESC is packaged with Cisco Network Service Orchestrator (NSO), and available within Cisco Solutions such as Virtually Managed Services (vMS) and Cisco Intercloud Services (CIS).
- As a standalone product, ESC is available as a VNFM bundled with Cisco VNFs such as VPN, vRouter, vSecurity and many others.

When ESC is deployed as a part of the vMS, CIS, VPN, vRouter and so on, these applications interface with ESC through the Northbound APIs. ESC supports both REST and NETCONF northbound interfaces for operations and transactions. In addition, in ESC 2.0 onwards, the ESC portal supports CRUD operations for some of the task for Virtual Network Function lifecycle management.

This chapter contains information about the Northbound APIs and the ESC portal.

- Elastic Services Controller NB APIs, page 9
- Cisco Elastic Services Controller Portal, page 12

Elastic Services Controller NB APIs

Elastic Services Controller (ESC) supports REST and NETCONF northbound interfaces for operations and transactions. The northbound interfaces interact with the NB client, NSO or any OSS. For REST interface interactions, callbacks are triggered, and for NETCONF/YANG interface interactions, NETCONF notifications are triggered.

NETCONF/YANG Northbound API

ESC uses NETCONF to configure and manage the network and its devices. NETCONF is a network management protocol to install, manipulate, operate and delete the configuration of network devices. Cisco NSO communicates with ESC using the open NETCONF protocol and YANG based data models. ESC manages Virtual Network Functions at a device level, and NSO manages the entire network service lifecycle. Together, they make it a complete orchestration solution that spans across both physical and virtual infrastructure.
ConfD enables integration with NSO by adding NETCONF/YANG northbound interface support. Along with NETCONF notifications, the NETCONF/YANG model also provides operational data. You can run query to get details such as list of all tenants, networks, and deployments in ESC.

Starting from ESC Release 2.1, you can create a single NETCONF request to perform multiple actions. The following is a NETCONF request to delete two tenants and subnets or networks simultaneously:

```xml
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant nc:operation="delete">
      <name>hezh-mix-tenant1</name>
    </tenant>
    <tenant nc:operation="delete">
      <name>hezh-mix-tenant2</name>
    </tenant>
  </tenants>
</esc_datamodel>
```

Examples of NETCONF/YANG API are as follows:

**NETCONF request to create a Tenant,**

```xml
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <source>
      <running/>
    </source>
    <config>
      <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
        <tenants>
          <tenant>
            <name>mytenant</name>
          </tenant>
        </tenants>
      </esc_datamodel>
    </config>
  </edit-config>
</rpc>
```

An escEvent of type CREATE_TENANT with a status of SUCCESS is sent to NETCONF subscribers once the configuration activation is completed. This indicates that the activation workflow is complete and the configuration resource is successfully created in the VIM.

**NETCONF notification after a tenant is successfully created:**

```xml
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-05-05T19:38:27.71+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_message>Tenant successfully created</status_message>
    <tenant>mytenant</tenant>
    <vm_source/>
    <vm_target/>
  </escEvent>
</notification>
```

The operational data (Opdata) for the tenant shows the name and tenant_id. NETCONF request,

```xml
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter select="esc_datamodel/opdata/tenants/tenant[name='mytenant']" type="xpath"/>
  </get>
</rpc>
```
NETCONF response,

```xml
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
      <opdata>
        <tenants>
          <tenant>
            <name>mytenant</name>
            <tenant_id>dccd22a13cc64e388a4b8d39e6a8fa7f</tenant_id>
          </tenant>
        </tenants>
      </opdata>
    </esc_datamodel>
  </data>
</rpc-reply>
```

For more details on series of notifications, event failure notifications, and Opdata, see the Cisco Elastic Services Controller API Guide.

**REST Northbound API**

The REST API is a programmatic interface to the ESC that uses a Representational State Transfer (REST) architecture. The API accepts and returns HTTP or HTTPS messages that contain JavaScript Object Notation (JSON) or Extensible Markup Language (XML) documents. You can use any programming language to generate the messages and the JSON or XML documents that contain the API methods or managed object (MO) descriptions.

The API model includes these programmatic entities:

- **Classes**—Templates that define the properties and states of objects in the management information tree.
- **Methods**—Actions that the API performs on one or more objects.
- **Types**—Object properties that map values to the object state (for example, equipmentPresence).

A typical request comes into the ESC and is placed in the transactor queue in first in, first out (FIFO) order. The transactor gets the request from the queue, interprets the request, and performs an authorization check. After the request is confirmed, the transactor updates the MIT. This complete operation is done in a single transaction.

For detailed reference information about API classes, properties, and data types, see the Cisco Elastic Services Controller API Guide.

---

**Example of REST APIs**

To create a tenant using REST:

```
POST /v0/tenants/123 HTTP/1.1
Host: client.host.com
Content-Type: application/xml
Accept: application/xml
Client-Transaction-Id: 123456
Callback:/createtenantcallback

<?xml version="1.0" encoding="UTF-8"?>
<tenant xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <name>Elie</name>
  <enabled>true</enabled>
  <description>A description...</description>
</tenant>
```
REST response after a tenant is successfully created:

HTTP/1.1 201 OK
Content-Type: application/xml; charset=UTF-8
Content-Length: 200
Date: Sun, 1 Jan 2011 9:00:00 GMT
ESC-Transaction-Id: 123456
ESC-Status-Code: 200
ESC-Status-Message: Success

<?xml version="1.0" encoding="UTF-8"?>
<tenant>
  <external_tenant_id>23423490854004</external_tenant_id>
  <internal_tenant_id>434344896854965</internal_tenant_id>
  <name>Elie</name>
  <enabled>true</enabled>
  <description>A description...</description>
</tenant>

For more details on response callback, request parameters, see the Cisco Elastic Services Controller API Guide.

Note: Further in this document, examples for scenarios will be provided either using REST or NETCONF/YANG, but not both.

Cisco Elastic Services Controller Portal

The ESC portal is a simplified Web-based tool for an ESC administrator to create, read, update, or delete (CRUD) operations related to VNF lifecycle management. As an administrator you can create and view the real-time activities of ESC such as deploying, undeploying, healing and scaling.

The ESC portal is enabled by default while creating an ESC VM either in an OpenStack or VMWare environment. For more information on enabling or disabling the ESC portal, see Cisco Elastic Services Controller Install and Upgrade Guide.

To start, stop and check the status of the ESC UI, do the following:

1. Connect to ESC using SSH:
   
   ssh user-name@ip-address-of-esc-vm

2. Switch to the root user:
   
   sudo su

3. To check the ESC UI process status:
   
   status esc_ui

4. To start ESC UI, run start esc_ui

5. To stop ESC UI, run stop esc_ui

6. To restart ESC UI, run restart esc_ui

ESC UI limitations are listed under Limitations section in the Elastic Services Controller Release Notes.

Logging into the Web Portal

The ESC portal provides default admin access. admin can create new users, and assign privileges to them. User profiles can also be updated in the UI.
1. Using your web browser, enter the IP address of ESC and port number:
   - Login via HTTP: http://esc-mgmt-ip:9000
   - Login via HTTPS: https://esc-mgmt-ip:9001

2. Enter the username and password, and login to the portal.

For more details on port, and browser requirements see the Cisco Elastic Services Controller Install and Upgrade Guide

**Getting Started with ESC Portal**

Table below lists the details users can view in the portal:

<table>
<thead>
<tr>
<th>Task</th>
<th>Navigate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To view Dashboard</td>
<td>Go to Dashboard</td>
<td>View the summary of all the managed ESC resources, infrastructure, notifications, and the system health.</td>
</tr>
</tbody>
</table>

*Note* These tasks can also be performed using the NB APIs. See the Elastic Services Controller NB APIs, on page 9 for more details.
<table>
<thead>
<tr>
<th>Task</th>
<th>Navigate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To view existing deployments</td>
<td><strong>OpenStack</strong></td>
<td>Shows all the deployed VMs, and VM groups and their status.</td>
</tr>
<tr>
<td></td>
<td>Go to Deployments</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• To view further details, click <strong>Actions</strong> and then <strong>View VNFs</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Select the KPI Data tab to view the monitoring details.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Select the Rules tab to view the admin rules and actions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Select the Scaling tab to view scaling details.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>VMware</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Go to Deployments</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• To view further details, click <strong>Actions</strong> and then <strong>View VNFs</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Select the KPI Data tab to view the monitoring details.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Select the Rules tab to view the admin rules and actions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Select the Scaling tab to view scaling details.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• To start, stop, reboot and enable or disable monitoring for a specific VM, click <strong>Actions</strong> and then select the corresponding action.</td>
<td></td>
</tr>
<tr>
<td>To view tenants</td>
<td>Go to <strong>Resources &gt; Tenants</strong></td>
<td>Provides a list of tenants, name, description and tenant ID.</td>
</tr>
<tr>
<td></td>
<td><strong>Important</strong></td>
<td>Tenant administrators will be used for all deployments in VMware. ESC does not support multi-tenancy in the VMware environment.</td>
</tr>
<tr>
<td>Task</td>
<td>Navigate</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>To view networks, sub-networks and interfaces</td>
<td>Go to <strong>Resources &gt; Networks</strong></td>
<td>Details of Network, Sub-network and Interfaces are available on different tabs. You can find details such as name, network ID, tenant ID and so on for each of them.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> Sub-network and Interfaces tabs are available only on Openstack environment.</td>
</tr>
<tr>
<td>To view VNF Images</td>
<td>Go to <strong>Resources &gt; Images</strong></td>
<td>Provides list of images for the selected resources details.</td>
</tr>
<tr>
<td>To view VNF deployment flavors</td>
<td>Go to <strong>Resources &gt; Flavors</strong></td>
<td>Provides list of flavors for the selected resources.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> Flavors is available only on Openstack based ESC VMs.</td>
</tr>
<tr>
<td>To view the switch details (VMware only)</td>
<td>Go to <strong>Resources &gt; Switches</strong></td>
<td>Provides a list of switches, name, description, UUID and hosts.</td>
</tr>
<tr>
<td>To view logs</td>
<td>Go to <strong>System &gt; Log</strong></td>
<td>You will find real-time logs for ESC events throughout VNF lifecycle. In the Setting page, you can filter error, trace, warning, thread activities, rest calls, database, transitions, OpenStack driver and timer messages. Displays real-time logs for ESC events, such as messages from the external systems, messages from ESC to external systems, and some key events such as spin up, spin down, failures and services.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To view settings, Click the gear icon at the top right corner.</td>
</tr>
<tr>
<td>To view incoming requests to ESC</td>
<td>Go to <strong>System &gt; Incoming Requests</strong></td>
<td>Lists all the incoming requests to ESC such as Transaction ID and request details.</td>
</tr>
<tr>
<td>To view configurations</td>
<td>Go to <strong>System &gt; Configuration</strong></td>
<td>Lists all the configuration parameters used for ESC, and configuration details of VMs, Day-0 configuration and monitoring rules. Shows dynamic information on the CPU usage, and memory utilization.</td>
</tr>
<tr>
<td>Task</td>
<td>Navigate</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>To view boot parameters (OpenStack only)</td>
<td>Go to <strong>System &gt; Boot Parameters</strong></td>
<td>Lists all the boot parameters used to boot ESC.</td>
</tr>
<tr>
<td>To view host details (OpenStack only)</td>
<td>Go to <strong>System &gt; Host details</strong></td>
<td>Lists the host details such as Operating System (OS), version of the OS, System up time, RAM, Storage and other details.</td>
</tr>
<tr>
<td>To view the health of ESC (OpenStack only)</td>
<td>Go to <strong>System &gt; Health</strong></td>
<td>Show the health of ESC, Conf'd status, Operational status and other details.</td>
</tr>
<tr>
<td>To view the infrastructure details (OpenStack only)</td>
<td>Go to <strong>Infrastructure &gt; Instances</strong></td>
<td>All VMs running on the virtualization infrastructure.</td>
</tr>
<tr>
<td>To view the Hypervisors (OpenStack only)</td>
<td>Go to <strong>Infrastructure &gt; Hypervisors</strong></td>
<td>All hypervisors running on the virtualization infrastructure.</td>
</tr>
</tbody>
</table>
| To deploy a VNF | - Go to **Deployments**  
- Click **New Deployment**  
- Select **Deploy from a file**  
- Click **Drop File Here** and locate the file or, Drag and drop your file to the **Drop File Here** area  
| **Important** To deploy a VNF in the VMWare environment using a form, see the **Deploying Using a Form**, on page 42. | Deploys a VNF.  
The drag and drop feature allows you to grab an existing deployment datamodel and to re-use it by dragging the file to the drop off area.  
**Note** Only xml files are accepted.  
The drag and drop feature executes a REST call as of now and does not execute NETCONF calls. |
| To un-deploy | - Go to **Deployments**  
- Click **Actions**  
- Click **Un-deploy** | Un-deploys VNF(s). |
| To view VDC | Choose **Resources > Datacenters** | View list of all Virtual Datacenters. |
Cisco Elastic Services Controller Portal Dashboard

The Cisco Elastic Services Controller dashboard provides a tabular representation of all the managed ESC resources, deployments, infrastructure, incoming requests, notifications, and visual indicators of system health. The following dashboard elements help you to track, diagnose, monitor data over time and the system health.

- Resources
- Deployments
- Infrastructure
- Incoming Requests
- Notifications
- System

The dashboard is best used in a monitoring desk context, where the system displaying the dashboard is dedicated for that purpose and might be distinct from the systems running the protocol servers. The dashboard system should point its browser to the system running the protocol servers.

If you notice unusual spikes or drops in activity, there could be communication failures or power outages on the network that you need to investigate.

Resources

The Resources dashboard displays information related to ESC portal managed (in-band) resources. You can view the active VMs count, VMs deploying count, Error VMs count, count status of Tenants, Images, Flavors, and Networks.

Deployments

The Deployments dashboard displays a high-level summary of VNFs that is currently being deployed. You can view the name and status of the VNF, and the number of VMs that are deployed in the VNF.

Infrastructure

The Infrastructure dashboard displays information about the virtual infrastructure, the status and IP address of the VIM.

Incoming Requests

The Incoming Requests dashboard displays the requested action, callback URL from where the request action was performed, the type of incoming request and the time when the request was sent. ESC integrates with other applications through Northbound APIs (both REST and NETCONF) to list incoming request details.

Notifications

The Notifications dashboard displays only notifications received on the Portal from ESC.
Systems

The System dashboard element displays the tabular and graphical representation of the platform's performance data. It also displays components health of the selected resources.
Virtual Network Functions Life Cycle Management

Cisco Elastic Services Controller (ESC) provides a single point of control to manage all aspects of VNF lifecycle for generic virtual network functions (VNFs) in a dynamic environment. It provides advanced VNF life cycle management capabilities through an open, standards-based platform that conforms to the ETSI VNF management and orchestration (MANO) reference architecture.

In ESC Release 2.0, you can orchestrate VNFs within a virtual infrastructure domain—either in an OpenStack or ESXi 6.0 environment. A VNF deployment is initiated as a service request. The service request comprises of templates that consist of XML payloads.

Note

Starting from Cisco ESC Release 2.0, you can deploy VNFs either on OpenStack or VMware environment. Hybrid VNF deployment is not supported.

ESC manages the complete life cycle of a VNF. Triggered by a northbound request, it instantiates a virtual machine to facilitate the requirements of a VNF. The requester can specify all the characteristics (for example, vCPU, memory, disk, monitoring KPIs, and more) typically associated with spinning up and managing a virtual machine in an XML template.

The figure explains the lifecycle management of ESC:
• Onboarding—In ESC, you can onboard any new VNF type as long as it meets the prerequisites for supporting it in an OpenStack and VMware environment. For example in Openstack environment, Cisco ESC supports QOW2 image format and config drive support for the VNF bootstrap mechanism. You can define the XML template for the new VNF type to onboard the VNF with ESC.

• Deploying—When a VNF is deployed, Cisco ESC applies "day-zero" configuration for a new service. A typical configuration includes credentials, licensing, connectivity information (IP address, gateway), and other static parameters to make the new virtual resource available to the system. It also activates licenses for the new VNFs.

• Monitoring—ESC integrates with the host hypervisor, whether KVM/OpenStack or VMware, to monitor the health of virtual machines. It tracks performance metrics such as CPU use, memory consumption, and other core parameters. The requester can specify all of the characteristics (for example, vCPU, memory, disk, monitoring KPIs, and more) typically associated with spinning up and managing a virtual machine in an XML template. It also provides an elaborate framework to monitor service performance-related metrics and other key parameters that you define.

• Healing—ESC heals the VNFs when there is a failure. The failure scenarios are configured in the KPI section of the datamodel. ESC uses KPI to monitor the VM and the events are triggered based on the KPI conditions. The actions to be taken for every event that is triggered is configured in the rules section during the deployment.

• Updating—Starting from ESC Release 1.1 and onwards, ESC allows deployment updates during deployment. You can either perform all the updates (that is, add or delete a vm_group, add or delete a ephemeral network in a vm_group, and add or delete an interface in a vm_group) in a single deployment or individually.

Note

Deployment updates is supported only on the OpenStack Environment.

• Undeploy—ESC allows you to undeploy an already deployed VNF. This operation is either done using the northbound APIs or through the ESC portal.

The following section explains how to deploy VNFs in OpenStack and VMware environments:

• Deploying VNFs in the OpenStack Environment, page 20
• Deploying VNFs in the VMware Environment, page 21

Deploying VNFs in the OpenStack Environment

In ESC, VNF deployment is initiated as a service request either originating from the ESC portal or the northbound interfaces. The service request comprises of templates that consist of XML payloads. These resources must either be available in OpenStack or can be created in ESC using the ESC portal or the northbound interfaces. For more information on managing resources in ESC, see Managing VM Resources in ESC, on page 23. The deployment datamodel refers to the resources to deploy VNFs in the OpenStack environment. Based on how the resources are setup, you can deploy VNFs in one of the following ways:
### Scenarios

<table>
<thead>
<tr>
<th>Deploying VNFs (Creating Images and Flavors through ESC)</th>
<th>Description</th>
<th>Datamodel</th>
<th>Images, Flavors, and Volumes</th>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>The process of VNF deployment is as follows:</td>
<td>• deployment datamodel</td>
<td>Images and Flavors are created through ESC using NETCONF/REST APIs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 VNF Deployment-</td>
<td>• image datamodel</td>
<td>• You can add or delete image definitions through ESC.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The deployment datamodel refers to the images and flavors created and then deploys VNFs.</td>
<td>• flavor datamodel</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deploying VNFs (Using out-of-band images, flavors, and volumes)</th>
<th>Description</th>
<th>Datamodel</th>
<th>Images, Flavors and Volumes</th>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>The process of VNF deployment is as follows:</td>
<td>• deployment datamodel</td>
<td>Images, Flavors and Volumes are not created through ESC.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 VNF Deployment-</td>
<td>• Images, Flavors, and Volumes in OpenStack</td>
<td>• The images, flavors, and volumes can be used in multiple VNF deployments.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The deployment datamodel refers to the out-of-band images, flavors, and volumes in OpenStack and then deploys VNFs.</td>
<td></td>
<td>• You cannot add or delete images through ESC.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Note**

For more information on Deploying VNFs in OpenStack environments, see Deploying Virtual Network Functions in the OpenStack Environment, on page 37.

---

### Deploying VNFs in the VMware Environment

In ESC, VNF deployment is initiated as a service request either originating from the ESC portal or the Northbound interface. The service request comprises of templates that consist of XML payloads such as Tenants, Network, Image, Flavor, and so on. These resources must be available in VMware environment. For more information on managing VM resources in ESC, see Managing VM Resources in ESC, on page 23. The deployment datamodel refers to the resources to deploy VNFs in the VMware environment.

In ESC Release 2.0, when you deploy VNFs in VMware environment, you can either use the out-of-band images that are already available in VMware or create an image in the ESC portal or using REST APIs. For more information on creating images in the ESC portal, see Managing Images, on page 31. The deployment datamodel refers to these images to deploy VNFs.
### Scenarios

<table>
<thead>
<tr>
<th><strong>Deploying VNFs (Creating Images through ESC)</strong></th>
<th><strong>Description</strong></th>
<th><strong>datamodel templates</strong></th>
<th><strong>Images</strong></th>
<th><strong>Advantages</strong></th>
</tr>
</thead>
</table>
| **Important** Images are also referred to as Templates in the VMWare environment. | The process of VNF deployment is as follows: 1. VNF Deployment: The *deployment datamodel* refers to the images created and then deploys VNFs. | • *deployment datamodel*  
• *image datamodel* | Images are created through ESC using REST APIs. | • The images can be used in multiple VNF deployments.  
• You can add or delete image definitions through ESC. |

| **Deploying VNFs (Using out-of-band images)** | **1** VNF Deployment: The *deployment datamodel* refers to the out-of-band images in VMware and then deploys VNFs. | • *deployment datamodel*  
• Image in VMware | Images cannot be created or deleted through ESC. | • The images can be used in multiple VNF deployments.  
• You can view images through ESC portal.  
• During out-of-band deployment, you can choose images. |

For more information on Deploying VNFs in VMWare environments, see Deploying Virtual Network Functions in the VMWare Environment, on page 39.
Managing VM Resources in ESC

Cisco Elastic Services Controller (ESC) resources comprises of Images, Flavors, Tenants, Volumes, Networks, and Subnets. These resources make that ESC request to provision a Virtual Network Function. These resources makeup the basic building blocks of a VNF service request, for example, Image is a bootable file system that can be used to launch VM instances. To manage these resources, you need to create the corresponding resources in ESC. These resource definitions exist or are created in OpenStack or VMware based on the provisioned infrastructure.

Depending upon the type of VNF Deployment, you must ensure that the necessary resource definitions are available either on OpenStack or VMware environments. When you deploy VNFs in the OpenStack environment you can either create these resource definitions in ESC or you have the option to use out-of-band image and flavor definitions that are already available in OpenStack.

In ESC Release 2.1, when you deploy VNFs in VMware environment, you can either use the out-of-band images that are already available in VMware or create an image in the ESC portal or using REST APIs. For more information on creating images in the ESC portal, see Managing Images, on page 31. The deployment datamodel refers to these images to deploy VNFs.

In ESC Release 2.1, the procedure to create the resource definitions varies in OpenStack and VMware environments.

The following table lists the different environments and the list of resource definitions that must be made available before VNF deployment:

<table>
<thead>
<tr>
<th>Resource Definitions</th>
<th>OpenStack</th>
<th>VMware</th>
</tr>
</thead>
</table>
| Tenants              | Creating and deleting tenant definitions is done in one of the following ways:  
|                      | • NETCONF API  
|                      | • REST API  
<p>|                      | <strong>Important</strong> ESC does not support out-of-band tenants to deploy VNFs. | <strong>Important</strong> Tenant administrators will be used for all deployments. ESC does not support multi-tenancy in the VMware environment. |</p>
<table>
<thead>
<tr>
<th>Resource Definitions</th>
<th>OpenStack</th>
<th>VMware</th>
</tr>
</thead>
</table>
| Networks             | Creating and deleting network definitions is done in one of the following ways:  
  • NETCONF API  
  • REST API | Creating and deleting distributed port group definitions is done in one of the following ways:  
  • NETCONF API  
  • REST API  
  • ESC Portal | Note  
  • In ESC 2.1, the resource creation is supported only in the default data center in the VMWare environment, and not supported in the Multi-VDC environment. For more information see Multi-Virtual Data Center Support (VMware Only), on page 40.  
  • ESC does not support out-of-band networks to deploy VNFs. |
| Subnets              | Creating and deleting subnet definitions is done in one of the following ways:  
  • NETCONF API  
  • REST API | Creating and deleting subnet definition is done in one of the following ways:  
  • NETCONF API  
  • REST API | Important  
  ESC does not support out-of-band subnets to deploy VNFs. |
| Flavors              | You can either use out-of-band flavor definitions that are already available in OpenStack or create flavor definitions in one of the following ways:  
  • NETCONF API  
  • REST API | Not applicable |
You can either use out-of-band image definitions that are already available in VMware or create image definitions in one of the following ways:

- NETCONF API
- REST API

In ESC 2.1, the resource creation is supported only in the default data center in the VMWare environment, and not supported in the Multi-VDC environment. For more information, see Multi-Virtual Data Center Support (VMware Only), on page 40.

<table>
<thead>
<tr>
<th>Resource Definitions</th>
<th>OpenStack</th>
<th>VMware</th>
</tr>
</thead>
</table>
| Images               | You can either use out-of-band image definitions that are already available in OpenStack or create image definitions in one of the following ways:  
  • NETCONF API  
  • REST API | You can either use out-of-band image definitions that are already available in VMware or create image definitions in one of the following ways:  
  • NETCONF API  
  • REST API  
  • ESC Portal |

| Volumes | You can use out-of-band volumes that are already available in OpenStack. For more information, see Managing Volumes (OpenStack Only), on page 33. | Not applicable |

The following chapter explains the procedure to manage the VNF resource definitions for OpenStack and VMware in ESC.

- Managing Tenants, page 25
- Managing Networks, page 26
- Managing Subnets, page 29
- Managing Flavors, page 30
- Managing Images, page 31
- Managing Volumes (OpenStack Only), page 33

## Managing Tenants

In ESC 2.0 and later, a tenant identifies a tenant organization or group that is associated with a set of administrators. When you create tenant definitions, the data stored on both regional and local clusters is segmented by tenant. A tenant cannot access the data of another tenant. You can use NETCONF/REST interface to create a tenant definition through ESC.
ESC Release 2.1 does not support creating and deleting tenant definitions in VMware environments.

Adding Tenants Using Northbound APIs

The following example explains how to create a tenant definition using NETCONF:

```xml
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <source>
      <running/>
    </source>
    <config>
      <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
        <tenants>
          <tenant>
            <name>mytenant</name>
          </tenant>
        </tenants>
      </esc_datamodel>
    </config>
  </edit-config>
</rpc>
```

For more information about creating and deleting tenant definitions using NETCONF or REST APIs, see Cisco Elastic Services Controller API Guide.

Managing Networks

In ESC, you can configure rich network topologies by creating and configuring networks and subnets, and then instructing either OpenStack or VMware services to attach virtual machines to ports on these networks.

OpenStack Network

In particular, OpenStack network supports each tenant to have multiple private networks, and allows tenants to choose their own IP addressing scheme, even if those IP addresses overlap with those used by other tenants. This enables very advanced cloud networking use cases, such as building multi-tiered web applications and allowing applications to be migrated to the cloud without changing IP addresses.

ESC supports the following networking functions:

- **Tenant Network**—A tenant network is created for a single network and all its instances. It is isolated from the other tenants.

- **Provider Network**—A provider network is created by the administrator. The attributes are mapped to the physical underlying network or a segment.

  The following attributes define a provider network:

  - network_type
  - physical_network
  - segmentation_id
• **External Network**—An external network typically provides Internet access for your instances. By default, this network only allows Internet access from instances using Network Address Translation (NAT). You can enable Internet access to individual instances using a floating IP address and suitable security group rules. The admin tenant owns this network because it provides external network access for multiple tenants.

ESC also supports Ephemeral networks which are short-lived tenant networks purposely created during unified deployment and exists only till the lifetime of that deployment. For more details, see Unified Deployment Request.

### VMware Network

In the VMware environment, you configure a distributed port on a vSphere distributed switch that connects to the VMkernel or to a virtual machine's network adapter. It specifies port configuration options for each member port on a vSphere distributed switch. Distributed port groups define how a connection is made to a network. You can use REST interface to create distributed port groups.

The following example shows how to create a distributed port group (VMWare only) using REST API:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <name>network-portgroup-01</name>
  <switch_name>vdSwitch-01</switch_name>
  <vlan_id>0</vlan_id>
  <number_of_ports>8</number_of_ports>
</network>
```

### Adding Networks in ESC Portal (VMware Only)

To add a network from the ESC portal, do the following:

**Procedure**

1. **Step 1** Choose Resources > Networks.
2. **Step 2** Click Add Network.
3. **Step 3** From the Switch drop-down list, select a switch.
4. **Step 4** In the Network Name field, enter the network name.
5. **Step 5** In the VLAN field, enter the number of VLANs.
6. **Step 6** In the Number of Ports field, enter the number of ports.
7. **Step 7** Click Create Network.

### Deleting Networks in ESC Portal

To delete a network from the ESC portal, do the following:
Procedure

**Step 1** Choose Resources > Networks.

**Step 2** Select a network from the list and click Actions.

**Step 3** Click Delete.

## Adding Networks Using Northbound APIs

The following example shows how to create a tenant network definition using NETCONF:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns="http://www.cisco.com/esc/esc">
<tenants>
<tenant>
<name>quicktest4</name>
<networks>
<network>
<name>proto-tenant-network34</name>
<shared>false</shared>
<admin_state>true</admin_state>
</network>
</networks>
</tenant>
</tenants>
</esc_datamodel>
```

The following example shows how to create a subnet for tenant network definition using NETCONF:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns="http://www.cisco.com/esc/esc">
<tenants>
<tenant>
<name>quicktest4</name>
<networks>
<network>
<name>proto-tenant-network27</name>
<subnet>
<name>proto-tenant-subnet4</name>
<ipversion>ipv4</ipversion>
<dhcp>true</dhcp>
<address>10.60.2.0</address>
<netmask>255.255.255.0</netmask>
<gateway>10.60.2.1</gateway>
</subnet>
</network>
</networks>
</tenant>
</tenants>
</esc_datamodel>
```

The following example shows how to create a simple provider network definition using NETCONF:

```xml
<?xml version="1.0"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns="http://www.cisco.com/esc/esc">
<networks>
<network>
<name>leke-net-12</name>
</network>
</networks>
</esc_datamodel>
```
The following example shows how to create a subnet for a provider network definition using NETCONF:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns="http://www.cisco.com/esc/esc">
<networks>
<network>
<name>leke-net-12</name>
<subnet>
<name>leke-net-12-subnet</name>
<ipversion>ipv4</ipversion>
<dhcp>false</dhcp>
<address>10.20.0.0</address>
<gateway>10.20.0.1</gateway>
<netmask>255.255.255.0</netmask>
</subnet>
</network>
</networks>
</esc_datamodel>
```

The following example shows how to create an external network definition using NETCONF:

```xml
<network>
<name>hezh-yesc-net-1</name>
<shared>false</shared>
<admin_state>true</admin_state>
<router_external/>
<subnet>
<name>hezh-yesc-subnet-1</name>
<ipversion>ipv4</ipversion>
<dhcp>true</dhcp>
<address>10.25.90.0</address>
<gateway>10.25.90.1</gateway>
<netmask>255.255.255.0</netmask>
</subnet>
</network>
```

For more information about creating and deleting network using NETCONF or REST APIs, see Cisco Elastic Services Controller API Guide.

### Managing Subnets

In ESC, a subnet is assigned to a virtual network. It specifies the IP address, the IP version for a network and so on. You can use NETCONF/REST interface to create subnet definitions. Subnet is supported in OpenStack environment and not supported in VMware environment.

### Adding Subnet Definitions Using Northbound APIs

The following example shows how to create a subnet definition using NETCONF:

```xml
<rpc message-id="1" xmlns:urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
<target>
<running/>
```
For more information about creating subnets using NETCONF or REST APIs, see Cisco Elastic Services Controller API Guide.

Managing Flavors

A flavor defines sizes for RAM, disk, and number of cores.

When you deploy VNFs in OpenStack environment, you either have an option to use out-of-band flavors that are already available in OpenStack or create flavors in ESC. These flavors can be created using NETCONF or REST interface and can be used to deploy multiple VNFs without registering. For more information on deployment attributes see, Cisco Elastic Services Controller Deployment Attributes.

Note

ESC Release 2.0 and later does not support creating or deleting flavor definitions in VMware environments.

Adding Flavors Using Northbound APIs

NETCONF request to create a flavor:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
<flavors>
  <flavor>
    <name>nashrest-flavor-indep</name>
    <vcpus>1</vcpus>
    <memory_mb>512</memory_mb>
    <root_disk_mb>0</root_disk_mb>
    <ephemeral_disk_mb>0</ephemeral_disk_mb>
    <swap_disk_mb>0</swap_disk_mb>
  </flavor>
</flavors>
</esc_datamodel>
```

Note

For more information about creating subnets using NETCONF or REST APIs, see Cisco Elastic Services Controller API Guide.
NETCONF notification upon successful creation of a flavor:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-07-13T13:33:51.805+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_message>Flavor creation completed successfully.</status_message>
    <flavor>nashrest-flavor-indep</flavor>
    </escEvent>
</notification>
```

For more information about creating and deleting flavors using NETCONF or REST APIs, see Cisco Elastic Services Controller API Guide.

Managing Images

In ESC, an image is a bootable file system that can be used to launch VM instances.

When you deploy VNFs in OpenStack environment, you either have an option to use out-of-band images that are already available in OpenStack or create images in ESC. These images can be created using NETCONF/REST interface and can be used to deploy multiple VNFs without registering.

When you deploy VNFs in VMware environment, you can either use the out-of-band images that are already available in VMware or create an image in the ESC portal, or using REST APIs, or using NETCONF API.

For more information on deployment attributes see, Cisco Elastic Services Controller Deployment Attributes.

Adding Images in ESC Portal (VMware Only)

The ESC portal allows you to create an image by filling the appropriate fields in the form.

Creating Image from a Form

To create images from a form, do the following:

**Procedure**

**Step 1** Choose Resources > Images.

**Step 2** Click Add VNF Image.

**Step 3** Select the Create Image from a Form.

**Step 4** In the Image Name field, enter the image name.

**Step 5** In the Image Path field, enter the image path.

**Step 6** Click Create Image.
Deleting Images in ESC Portal

To delete images from the ESC Portal, do the following:

**Procedure**

**Step 1** Choose **Resources > Images**.
**Step 2** Select an image from the list and click **Delete**.

Adding Images Using Northbound APIs

**Note**

In ESC Release 2.0 and later, when you deploy VNFs in VMware environment, you can either use the out-of-band images that are already available in VMware or create an image in the ESC portal or using REST APIs, or using NETCONF API.

**NETCONF request to create an image:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
    xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
    xmlns:ns0="http://www.cisco.com/esc/esc"
    xmlns="http://www.cisco.com/esc/esc">
    <images>
        <image>
            <name>nashrest-cirrosimage-indep</name>
            <src>http://10.85.74.227:/share/images/esc_automated_test_images/cirros-0.3.3-x86_64-disk.img</src>
            <disk_format>qcow2</disk_format>
            <container_format>bare</container_format>
            <serial_console>true</serial_console>
            <disk_bus>virtio</disk_bus>
        </image>
    </images>
</esc_datamodel>
```

**NETCONF notification upon successful creation of an image:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2015-07-13T13:46:50.339+00:00</eventTime>
    <escEvent xmlns="http://www.cisco.com/esc/esc">
        <status>SUCCESS</status>
        <status_message>Image creation completed successfully.</status_message>
        <image>nashrest-cirrosimage-indep</image>
    </escEvent>
</notification>
```
NETCONF request to delete an image:

For more information about creating images using NETCONF or REST APIs, see Cisco Elastic Services Controller API Guide.

Managing Volumes (OpenStack Only)

Starting from ESC Release 2.1, when you deploy VNFs in the OpenStack environment, you have an option to attach out-of-band volumes that are already available in OpenStack to VMs created in OpenStack from ESC.

The following snippet shows how to specify an out-of-band volume in deployment datamodel:

```
<volumes>
  <volume>
    <name>pre-existing volume</name>
    <volid>1</volid>
    <bus>ide</bus>
    <type>lvm</type>
  </volume>
</volumes>
```

Where,

- **Name**— Specifies the display name of the pre-existing volume.
- **Volid**— Specifies the order in which volumes are attached. These are consecutive numbers starting from 0 or 1 for every VM group.
- **Bus**— Specifies the bus type of the volumes to be attached; a value of ide maps to hdb; a value of virtio maps to vdb; a value of scsi maps to sdb.
- **Type**— (Optional) If <type> is specified, then ESC matches the volume with the type provided.
Deploying Virtual Network Functions

This chapter describes several deployment scenarios for Elastic Services Controller (ESC), the procedures to deploy VNFs (OpenStack and VMware environments), and the operations that you can perform during deployment.

Important

Starting from ESC Release 2.0, you can assign a static IP address to connect the network to the VNF. The deployment datamodel introduces a new ip_address attribute to specify the static IP address. See the Cisco Services Controller Deployment Attributes for more details.

The following table lists the VNF features that are supported on OpenStack or VMware:

<table>
<thead>
<tr>
<th>Feature</th>
<th>OpenStack</th>
<th>VMware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affinity and Anti-affinity</td>
<td>Creating and deleting affinity and anti-affinity rule definitions is done in one of the following ways:</td>
<td>Creating and deleting affinity rule definition in one of the following ways:</td>
</tr>
<tr>
<td>Rule</td>
<td>• NETCONF API</td>
<td>• NETCONF API</td>
</tr>
<tr>
<td></td>
<td>• REST API</td>
<td>• REST API</td>
</tr>
<tr>
<td></td>
<td>Note: ESC 2.1 supports only Affinity rules in VMware environment.</td>
<td>Note: ESC 2.1 supports only Affinity rules in VMware environment.</td>
</tr>
<tr>
<td>Single Root I/O Virtualization</td>
<td>Configuration of Single Root I/O Virtualization is done in one of the following ways:</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>• NETCONF API</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>• REST API</td>
<td>—</td>
</tr>
<tr>
<td>Individual and Composite VNFs</td>
<td>Configuration of Individual and Composite VNFs is done in one of the following ways:</td>
<td>Configuration of Individual and Composite VNFs is done in one of the following ways:</td>
</tr>
<tr>
<td></td>
<td>• NETCONF API</td>
<td>• NETCONF API</td>
</tr>
<tr>
<td></td>
<td>• REST API</td>
<td>• REST API</td>
</tr>
<tr>
<td>Feature</td>
<td>OpenStack</td>
<td>VMware</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Undeploy Virtual Network Functions</td>
<td>Undeploying is done in one of the following ways:</td>
<td>Undeploying VNFs is done in one of the following ways:</td>
</tr>
<tr>
<td></td>
<td>• NETCONF API</td>
<td>• NETCONF API</td>
</tr>
<tr>
<td></td>
<td>• REST API</td>
<td>• REST API</td>
</tr>
<tr>
<td></td>
<td>• ESC Portal</td>
<td>• ESC Portal</td>
</tr>
<tr>
<td>Day Zero Configuration</td>
<td>Day Zero configuration is done in one of the following ways:</td>
<td>Day Zero configuration is done in one of the following ways:</td>
</tr>
<tr>
<td></td>
<td>• NETCONF API</td>
<td>• NETCONF API</td>
</tr>
<tr>
<td></td>
<td>• REST API</td>
<td>• REST API</td>
</tr>
<tr>
<td></td>
<td>• ESC Portal</td>
<td>• ESC Portal</td>
</tr>
<tr>
<td>VNF Operations</td>
<td>VNF Operations is done in one of the following ways:</td>
<td>VNF Operations is done in one of the following ways:</td>
</tr>
<tr>
<td></td>
<td>• REST API</td>
<td>• REST API</td>
</tr>
<tr>
<td></td>
<td>For more information, see the Virtual Network Function Operations, on page 75.</td>
<td>For more information, see the Cisco Elastic Services Controller Portal, on page 12.</td>
</tr>
<tr>
<td>Multi Cluster</td>
<td>Not applicable</td>
<td>Multi Cluster configuration is done in one of the following ways:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• REST API</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ESC Portal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, see the Deploying Virtual Network Functions in ESC Portal (VMware Only), on page 41.</td>
</tr>
<tr>
<td>Multiple Virtual Datacenter</td>
<td>Not applicable</td>
<td>Multiple Virtual Datacenter selection is done in one of the following ways:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• REST API</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ESC Portal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, see the Multi-Virtual Data Center Support (VMware Only), on page 40.</td>
</tr>
</tbody>
</table>
### Hardware Acceleration Support (OpenStack Only)

<table>
<thead>
<tr>
<th>Feature</th>
<th>OpenStack</th>
<th>VMware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Acceleration</td>
<td>Hardware Acceleration is supported in one of the following ways:</td>
<td>Not applicable</td>
</tr>
<tr>
<td></td>
<td>• NETCONF API</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• REST API</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see the Hardware Acceleration Support (OpenStack Only), on page 53.</td>
<td></td>
</tr>
</tbody>
</table>

After you have decided on the deployment scenario and the environment you want to implement, see the following sections:

- Deploying Virtual Network Functions in the OpenStack Environment, page 37
- Deploying Virtual Network Functions in the VMware Environment, page 39
- Deploying Virtual Network Functions in ESC Portal (VMware Only), page 41
- Day-0 Configuration, page 43
- Unified Deployment Request, page 45
- Applying Affinity and Anti-affinity Rules, page 46
- Advanced Interface Configurations, page 50
- Managing Individual and Composite VNFs, page 52
- Hardware Acceleration Support (OpenStack Only), page 53
- Undeploying Virtual Network Functions, page 53
- Updating Deployments, page 54

## Deploying Virtual Network Functions in the OpenStack Environment

This section describes several deployment scenarios for Elastic Services Controller (ESC) and the procedure to deploy VNFs. The following table lists the different deployment scenarios:
Deploying Virtual Network Functions

In ESC, VNF deployment is initiated as a service request either originating from the ESC portal or the northbound interfaces. The service request comprises of XML payloads. These resources must either be available in OpenStack using the northbound interfaces. ESC supports the following deployment scenarios:

- Deploying VNFs (Creating images, and flavors through ESC)
- Deploying VNFs (Using out-of-band images, flavors, and volumes)

Before you deploy VNFs, you must ensure that images, flavors, and volumes are available in OpenStack or you must create these resources. For more details on creating images, flavors, and volumes see Managing VM Resources in ESC, on page 23.

During deployment, ESC looks for the deployment details in the deployment datamodel. For more information on the deployment datamodel, see Cisco Elastic Services Controller Deployment Attributes. If ESC is unable to find the deployment details for a particular service, it uses the existing flavor and image under the `vm_group` to continue the deployment. If ESC is unable to find the image, and flavor details, the deployment fails.
From ESC Release 2.0 and later, you can also specify the subnet that is used for a network. The deployment datamodel introduces a new `subnet` attribute to specify the subnet. See the Cisco Elastic Services Controller Deployment Attributes for more details.

For deployments in OpenStack, the UUID or name can be used to refer to the image and flavor. The name has to be unique on VIM. If there are multiple images with the same name, the deployment cannot identify the right image and the deployment fails.

### Deploying Virtual Network Functions in the VMware Environment

This section describes the deployment scenario for Elastic Services Controller (ESC) and the procedure to deploy VNFs in the VMware environment. In ESC Release 2.1, you can deploy VNFs using out-of-band image definitions. The following table lists the deployment scenario:

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Description</th>
<th>datamodel templates</th>
<th>Images</th>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploying VNFs (Creating Images through ESC)</td>
<td>The process of VNF deployment is as follows:</td>
<td></td>
<td>Images are created through ESC using REST APIs.</td>
<td>• The images can be used in multiple VNF deployments.</td>
</tr>
<tr>
<td>Important</td>
<td>Images are also referred to as Templates in the VMware environment</td>
<td></td>
<td>• deployment datamodel</td>
<td>• You can add or delete image definitions through ESC.</td>
</tr>
<tr>
<td></td>
<td>• VNF Deployment - The <code>deployment datamodel</code> refers to the images created and then deploys VNFs.</td>
<td></td>
<td>• image datamodel</td>
<td></td>
</tr>
<tr>
<td>Deploying VNFs (Using out-of-band images)</td>
<td>1 VNF Deployment - The <code>deployment datamodel</code> refers to the out-of-band images in VMware and then deploys VNFs.</td>
<td></td>
<td>Images cannot be created or deleted through ESC.</td>
<td>• The images can be used in multiple VNF deployments.</td>
</tr>
<tr>
<td></td>
<td>1 VNF Deployment - The <code>deployment datamodel</code> refers to the out-of-band images in VMware and then deploys VNFs.</td>
<td></td>
<td>• Image in VMware</td>
<td>• You can view images through ESC portal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• During out-of-band deployment, you can choose images.</td>
<td></td>
</tr>
</tbody>
</table>
Multi-Virtual Data Center Support (VMware Only)

A Virtual Data Center (VDC) is an environment that combines virtual resources, operational details, rules, and policies to manage specific group requirements. A group can manage multiple VDCs, images, templates, and policies. This group can allocate quotas and assign resource limits for individual groups at the VDC level.

When you deploy a VNF, you need to specify the virtual datacenter name on which the VNF needs to be provisioned. You need to specify the datacenter name as below:

```xml
<deployments>
  <deployment>
    <datacenter>OTT-03</datacenter>
    ...
  </deployment>
  ...
<deployments>
```

**Note** If the data center name is not specified during deployment, ESC deploys the VM in a default data center that is provided in the configuration parameters.

Cisco Elastic Services Controller Portal allows you to choose the VDC on which the VM is provisioned. When you are creating a service request, you can choose the VDC on which this VM is provisioned. For more information on deploying VNF on a VDC, see Deploying Virtual Network Functions in ESC Portal (VMware Only), on page 41.

To view the list of VDCs that are available and on the ESC portal, choose Resources > Datacenter.

**Before you Begin**

Before you deploy VNFs on multiple VDCs, ensure that the following conditions are met:

- Verify that a standard external network spanning both VDCs is available for the ESC to ping the deployed VMs.
- Verify that at least one management interface on the VMs is connected to the external network.
- Verify that the VDC is present in the vCenter.

**Limitations**

The following is the list of VDC limitations:

- Supported only through REST interface.
- Supports only deployment and not resource management. ESC assumes all required resources to be created in VDC are out of band and present in the VDC.
- ESC does not support admin level control on VDC scoping. Any present VDC on a vCenter can be accessed by ESC.
Passing OVF Properties to a VM

As a part of deploying a VNF in VMware environment, you can pass the name value pair as OVF property to the VM. To pass these configurations while deploying a VNF, you must include additional arguments in the *deployment datamodel* template.

A sample configuration is as follows:

```xml
<esc_datamodel ...>
  ...<config_data>
    ...
    <configuration>
      <dst>ovfProperty:mgmt-ipv4-addr</dst>
      <data>$NICID_1_IP_ADDRESS/24</data>
    </configuration>
    ...
    <configuration>
      <dst>ovfProperty:com.cisco.csr1000v:hostname</dst>
      <data>$HOSTNAME</data>
      <variable>
        <name>HOSTNAME</name>
        <val>csrhost1</val>
        <val>csrhost2</val>
      </variable>
    </configuration>
  </config_data>
...</esc_datamodel>
```


deploying virtual network functions in ESC Portal (VMware Only)

The ESC portal allows you to deploy a single VNF or multiple VNFs together. An existing deployment datamodel is either uploaded through the portal, or a new deployment datamodel created. A new deployment datamodel is created by filling all the appropriate fields in the ESC portal. ESC also allows you to export a deployment datamodel from the portal. The following section explains multiple ways to deploy VNFs using the ESC portal.

Deploy Using a File (deployment datamodel)

An existing deployment datamodel is used to deploy VNFs. The deployment datamodel is pre-configured with the number of VNFs and other specifications. It is either uploaded by locating the deployment datamodel or you can drag and drop the existing deployment datamodel. The drag and drop feature allows you to grab an existing deployment datamodel and to re-use it by dragging the file to the drop off area.

>Note>
Only XML files are accepted.
Deploying Using a Form

To create a new deployment template, do the following:

Procedure

Step 1 Choose Deployments.
Step 2 Click New Deployment.
Step 3 Select Deploy from a Form.
Step 4 Enter a Deployment name.
Step 5 From the Datacenter drop-down list, choose a datacenter on which you want to deploy the VNF. For more information on virtual datacenter, see Multi-Virtual Data Center Support (VMware Only), on page 40.
Step 6 In the General tab, enter the appropriate values for the fields.
   a) In the Placement field, select the Cluster or Host radio button.
      • Cluster—Choose the name of a cluster to deploy a VNF in the same cluster.
      • Host—Choose a host to deploy a VNF in the same host.
      • Datastore—Choose a datastore for the selected cluster.
      • Image Choose an image.
Step 7 Click Enable Smart Licensing to enable smart licensing.
Step 8 Click Enable Intragroup Rules to enable intragroup rules.
   a) From the Type drop-down list, choose Affinity to enable affinity rules.
      For more information on intragroup affinity rules, Applying Affinity and Anti-affinity Rules, on page 46.
Step 9  (Optional): Click the Add VNF Intergroup Rule tab to select VNFs for which you the affinity rules to be applicable. For more information on intergroup affinity rules, see Applying Affinity and Anti-affinity Rules, on page 46.

Step 10  To specify the parameters that ESC will utilize to heal the VNFs when there is a failure, click the Recovery tab. For more information on recovery or healing, see Healing Virtual Network Functions, on page 67.

Step 11  To specify the number of interfaces and properties for each interface, click the Interfaces tab. The order of the interfaces specified here does not correspond to the order of the interfaces in the VM.
   a)  Click Add Interface to add interfaces.

Step 12  To specify the number of instances of a particular type of VM that needs to be instantiated and to elastic scale in and scale out, click the Scaling tab.
   a)  Click Add Static IP Pool to add a static IP pool.

Step 13  To specify the monitoring rules that will be used to configure the monitor module within ESC, click the Monitoring tab. For more information on monitoring, see Monitoring Virtual Network Functions, on page 57.

Step 14  In the Config Data tab, enter the appropriate values for the fields.

Step 15  In the OVF Settings tab, enter the appropriate values for the fields.
   a)  Click Add OVF Property to add a list of OVF properties.

---

Day-0 Configuration

The initial or day-0 configuration of a VNF is based on the VM type. A VNF administrator configures the initial template for each VM type at the time of VNF deployment. The same configuration template is applied to all deployed and new VMs of that VM type. The template is processed at the time of individual VM deployment. The day-0 configuration continues to persist, so that all initial deployment, healing and scaling of VMs have the same day-0 template.

Some of the day-0 configuration tasks include bringing up the interface, managing the network, support for static or dynamic IP (DHCP, IPAM), SSH keys, and NetConf enabled configuration support on VNF.

Day-0 in the Configuration datamodel

Day 0 configuration is defined in the datamodel under the config_data tag. Each user data and the configuration drive file is defined under the configuration tag. The contents are in the form of a template. ESC processes the template through the Apache Velocity Template Engine before passing to the VM.

The config_data tag is defined for each vm_group. The same configuration template is applied to all VMs in the vm_group. The template file is retrieved and stored at deployment initialization. Template processing is applied at time of VM deployment. The content of the config file can be retrieved from the file or data

```xml
<file> url </file>
<data> inline config content </data>
```

The url specifies a file on the ESC VM file system or file hosted on report http server.

A destination name is assigned to the config by <dst>. User Data is a treated as a special case with <dst>=-user-data</dst>. 
A sample config datamodel,

```
<config_data>
<configuration>
<file>file://cisco/userdata_file.txt</file>
<dst>--user-data</dst>
<variable>
<name>CUSTOM_VARIABLE FOR_USERDATA</name>
<val>SOME_VALUE_XXX</val>
<variable>
</configuration>
<configuration>
<file>file://cisco/config.sh</file>
<dst>config.sh</dst>
<variable>
<name>CUSTOM_VARIABLE FOR_CONFIG</name>
<val>SOME_VALUE_XXX</val>
<variable>
</configuration>
</config_data>
```

Custom variable can be specified in the variables tag within the configuration. Zero or more variables can be included in each configuration. Each variable can have multiple values. Multiple values are only useful when creating more than one VM per vm_group. Also, when performing scale-in and scale-out, additional VMs can be added and removed from the VM group.

The contents of <file> are a template that is processed by the Velocity Template Engine. ESC populates a set of variables for each interface before processing the configuration template:

<table>
<thead>
<tr>
<th>NICID_n_IP_ALLOCATION_TYPE</th>
<th>string containing FIXED</th>
<th>DHCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>NICID_n_NETWORK_ID</td>
<td>string containing neutron network uuid</td>
<td></td>
</tr>
<tr>
<td>NICID_n_IP_ADDRESS</td>
<td>ipv4 or ipv6 address</td>
<td></td>
</tr>
<tr>
<td>NICID_n_MAC_ADDRESS</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>NICID_n_GATEWAY</td>
<td>ipv4 or ipv6 gateway address</td>
<td></td>
</tr>
<tr>
<td>NICID_n_CIDR_ADDRESS</td>
<td>ipv4 or ipv6 cidr prefix address</td>
<td></td>
</tr>
<tr>
<td>NICID_n_CIDR_PREFIX</td>
<td>integer with prefix-length</td>
<td></td>
</tr>
<tr>
<td>NICID_n_NETMASK</td>
<td>If an ipv4 CIDR address and prefix are present, ESC will automatically calculate and populate the netmask variable. This is not substituted in the case of an IPv6 address and should not be used.</td>
<td></td>
</tr>
<tr>
<td>NICID_n_ANYCAST_ADDRESS</td>
<td>string with ipv4 or ipv6</td>
<td></td>
</tr>
<tr>
<td>NICID_n_IPV4_OCTETS</td>
<td>string with last 2 octets of ip address, such as 16.66, specific to CloudVPN</td>
<td></td>
</tr>
</tbody>
</table>

Where n is the interface number from the datamodel, for example, 0, 1, 2, 3

Example
```
NICID_0_IP_ALLOCATION_TYPE: FIXED
NICID_0_NETWORK_ID: 9f8d9a97-d873-4a1c-8e95-1a123686f038
```
 Unified Deployment Request

ESC creates OpenStack resources such as tenants, networks, and sub-networks before deploying a VNF. During a unified deployment request, you send a unified request to create or delete the OpenStack resources, and deploy a VNF. You can create multiple networks and sub-networks, but can create only a single VNF and a single tenant using unified deployment request.

Update the deployment datamodel and the files with the necessary information such as the service and deployment ID, tenant, network and sub-network ids and so on. You can either use NETCONF or REST APIs. For example, send POST REST and DELETE REST calls.

See the Elastic Services Controller Registration and Deployment Attributes for a list of registration and deployment attributes.

- To create and update deployment datamodel with a single deployment request, send POST REST call to:
  http://[ESC_IP]:8080/v0/deployments/[internal_dep_id]

- To delete a single deployment request, send DELETE REST call to:
  http://[ESC_IP]:8080/v0/deployments/[internal_dep_id]

The VNF will be undeployed, and the subnet, network and tenant will be deleted in the specified order.

**Note**

- If a tenant is not created as part of unified deployment request, there is nothing to roll back. An error occurs and a manual un-deploy must be triggered to cleanup.

- If both create and delete unified deployment requests fail causing two failed events, there is an attempt to roll back twice. But, there is nothing to rollback.
Limitations:

- A unified deployment request can reuse an existing tenant only. It cannot reuse any existing networks, sub-networks or deployments. If there is a duplicate network, sub-network, or deployment, then the unified deployment request is rejected.

- During an undeploy request, any network and subnetwork created as part of the unified deployment request will be deleted along with the VNF. However, tenant created through unified deployment request will not be deleted.

Applying Affinity and Anti-affinity Rules

Affinity and anti-affinity rules create relationship between virtual machines (VMs) and hosts. The rule can be applied to VMs, or a VM and a host. The rule either keeps the VMs and hosts together (affinity) or separated (anti-affinity).

Important

ESC 2.1 supports only affinity rules in the VMware environment.

Policies are applied during individual VM deployment. Policies such as affinity and anti-affinity streamline the deployment process. During a composite VNF deployment, rules defined in these policies enable easy deployment of individual VMs reducing the deployment time and cost.

Affinity and anti-affinity rules are created and applied on VMs at the time of deployment. VM receives the placement policies when the deploy workflow is initialized.

During a composite VNF deployment, if a couple of VMs need to communicate with each other constantly, they can be grouped together (affinity rule) and placed on the same host.

If two VMs are over-loading a network, they can be separated (anti-affinity rule) and placed on different hosts to balance the network.

Grouping or separating VMs and hosts at the time of deployment helps ESC to manage load across the VMs and hosts in the network. Recovery and scale-up of these VMs do not impact the affinity and anti-affinity rules.

The anti-affinity rule can also be applied between VMs within the same group and on a different host. These VMs perform similar functions and support each other. When one host is down, VM on the other host continues to run preventing any loss of service.

Affinity and Anti-Affinity Rules in the OpenStack Environment

The following sections describe affinity and anti-affinity policies with examples.

Intra Group Affinity Policy

The VNFs within the same VM group can either be deployed on the same host, or into the same availability zone.

Example for Intra Group Affinity Policy:

```
<vm_group>
  <name>affinity-test-gp</name>
  <placement>
```

Cisco Elastic Services Controller 2.1 User Guide
Starting from ESC Release 2.0, the type `zone-host` is used to deploy VNFs in the same host, or into the same availability zone.

**Zone or Host Based Placement**

Starting from ESC Release 2.0, The VNFs are within the same VM group and deployed on the same host or the same available zone. The `host` tag is used to deploy VMs on the same host and the `zone` tag is used to deploy VMs in the same available zone. Before deploying, you need to make sure that the host exists in OpenStack. ESC validates the specified host in OpenStack. The `zone-host` tag specifies the type of placement. Hence, if a host or zone not specified during deployment, the deployment fails.

You cannot specify both the host and zone tag to deploy VM on the same host or the same available zone.

**Example for host placement:**

```xml
<vm_group>
    <name>zone-host-test-gp1</name>
    <placement>
        <type>zone_host</type>
        <enforcement>strict</enforcement>
        <host>my-ucs-4</host>
    </placement>
</vm_group>
```

**Example for zone placement:**

```xml
<vm_group>
    <name>zone-host-test-gp2</name>
    <placement>
        <type>zone_host</type>
        <enforcement>strict</enforcement>
        <zone>dt-zone</zone>
    </placement>
</vm_group>
```

**Intra Group Anti-Affinity Policy**

The VNFs within the same VM group are explicitly deployed on different hosts. For example, back-up VNFs.

**Example for Intra Group Anti-Affinity Policy:**

```xml
<vm_group>
    <name>anti-affinity-test-gp</name>
    <placement>
        <type>anti_affinity</type>
        <enforcement>strict</enforcement>
    </placement>
</vm_group>
```

**Inter Group Affinity Policy**

The VNFs in the same deployment but different VM groups can be explicitly deployed in the same host. For example VNF bundles. Multiple VM groups can follow this policy by adding the `vm_group_ref` tag and providing the VM group name as the value.
You can only use one `vm_group_ref` tag, type tag and enforcement tag under the placement tag. The host or zone cannot be specified.

**Example for Inter Group Affinity Policy:**

```xml
<services>
  <service_definition>
    <name>test-strict-affinity-2groups</name>
    <version>1.4</version>
    <policies>
      <placement>
        <target_vm_group_ref>affinity-test-gp1</target_vm_group_ref>
        <type>affinity</type>
        <vm_group_ref>affinity-test-gp2</vm_group_ref>
        <enforcement(strict)>
      </placement>
    </policies>
  </service_definition>
</services>
```

**Inter Group Anti-Affinity Policy**

The VNFs in the same deployment but different VM Groups can be explicitly deployed in different hosts. For example, back-up VNFs or High-availability VNFs. Multiple VM groups can follow this policy by adding the `vm_group_ref` tag and providing the VM group name as the value.

You can only use one `<target_vm_group_ref>` tag, type tag and enforcement tag under the placement tag. The host or zone cannot be specified.

You can use multiple `<vm_group_ref>` tags, however, the anti-affinity policy only applies between each `<vm_group_ref>` and their `<target_vm_group_ref>`, which means that 2 or more `<vm_group_ref>` can be deployed on the same host, as long as each of them are deployed on a different host from their `<target_vm_group_ref>` that is acceptable.

**Example for Inter Group Anti-Affinity Policy:**

```xml
<service_definition>
  <name>test-strict-affinity-2groups</name>
  <version>1.4</version>
  <policies>
    <placement>
      <target_vm_group_ref>affinity-test-gp1</target_vm_group_ref>
      <type>anti_affinity</type>
      <vm_group_ref>affinity-test-gp2</vm_group_ref>
      <enforcement>strict</enforcement>
    </placement>
  </policies>
</service_definition>
```

**Affinity Rules in the VMWare Environment**

Affinity rules are created as host-affinity and during deployment ESC deploys the first VM as an anchor VM for the affinity. All other VMs that follow the same affinity rule will be deployed to the same host as the anchor VM. The anchor VM deployment helps to optimize the resource usage.
Intra Group Affinity Policy

The VNFs within the same VM group can either be deployed on the same host, or into the same availability zone.

Example for Intra Group Affinity Policy:

```
<vm_group>
    <name>affinity-test-gp</name>
    <placement>
        <type>affinity</type>
        <enforcement>strict</enforcement>
    </placement>
...
```

Starting from ESC Release 2.0, the type `zone-host` is used to deploy VNFs in the same host, or into the same availability zone.

Cluster Placement

All VMs in a VM group can be deployed to a cluster. For example, all VMs in a vm group CSR-gp1 can be deployed to cluster ott-cluster2.

**Note**
The Vmware cluster must be created by the administrator.

Example for cluster placement:

```
<name>CSR-gp1</name>
    <placement>
        <type>zone_host</type>
        <enforcement>strict</enforcement>
        <zone>ott-cluster2</zone>
    </placement>
```

Host Placement

All VMS in a VM group can be deployed to a host. For example, all VMs in the vm group CSR-gp1 will be deployed to host 10.2.0.2.

```
<name>CSR-gp1</name>
    <placement>
        <type>zone_host</type>
        <enforcement>strict</enforcement>
        <host>10.2.0.2</host>
    </placement>
```

Inter Group Affinity Policy

The VMs in different VM groups can be deployed to the same host. For example, all VMs in the VM group ASA-gp1 can be deployed to the same host as the VMs in the VM group CSR-gp1.

**Note**
To ensure that the inter-group affinity rules are applied within a single cluster, verify that all VM groups in a deployment is specified to the same cluster (`<zone>` in esc_data_model).

Example for Inter Group Affinity Policy :

```
<deployment>
    <deployment>
```

Cisco Elastic Services Controller 2.1 User Guide
Limitations

Following are the limitations when affinity rules are applied in VMware environment:

- All Affinity rules defined in the Vmware will be implemented in a cluster.
- DPM and HA must be turned off.
- VM deployment and recovery are managed by ESC.
- DRS must be set to manual mode if it is turned on.
- DPM and Vmotion should be unchecked.
- Supported value for <enforcement> tag should be 'strict'.
- <zone_host> must be used for any vm group being deployed as part of an Intragroup affinity or Intergroup affinity placement.

Advanced Interface Configurations

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

Configuring Single Root I/O Virtualization in ESC

Prerequisites

Before you configure Single Root I/O Virtualization (SR-IOV) in ESC, we highly recommend that you configure the hardware and OpenStack with the correct parameters.

Configuring SR-IOV in ESC

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

The ESC deployment datamodel allows you to configure SR-IOV by using the interface type as direct. The following example explains how to configure SR-IOV:

```xml
<interfaces>
  <interface>
    <nicid>0</nicid>
    <network>esc-net</network>
    <type>direct</type>
  </interface>
</interfaces>
```
Configuring Allowed Address Pair

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

The address pair is configured in the following ways:

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

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

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

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

Configuring Security Group Rules

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

To configure security group rules, do the following:

Before You Begin

- Make sure you have created a tenant through ESC.
- Make sure you have security groups created.
- Make sure you have the list of UUIDs for the security groups.
Managing Individual and Composite VNFs

An individual service consists of a single VNF. A coupled service or a composite VNF consists of several VMs of different types. The ESC interface receives VM interdependency information from the northbound system, and uses this information during VM and VNF creation, and life cycle management. Interdependency could include bootup sequence, VM specific workflow in the group of VMs in a single VNF, VNF monitoring and scalability and so on.

Create, read, update and delete operations are allowed on the VMs. To add more VM instances to a deployed VNF using static IP, you must provide additional IP address into the static IP pool. If you are using an existing static IP deployment, the minimum number of VMs is altered.

If the new minimum value, which is the number of VMs is greater than the active VMs, a new VM is added to the service. If the value is greater than the max value, the update is rejected.
Hardware Acceleration Support (OpenStack Only)

Starting from ESC Release 2.1 and later, you can configure hardware acceleration features in OpenStack using the flavor datamodel. The following hardware acceleration features can be configured:

- **vCPU Pinning**—enables the binding and unbinding of a process to a vCPU (Virtual Central Processing Unit) or a range of CPUs, so that the process executes only on the designated CPU or CPUs rather than any CPU.

- **Openstack Kilo performance optimization for Large Pages and NUMA**—enables improvement of system performance for large pages and NUMA i.e., System's ability to accept higher load and modifying the system to handle a higher load.

- **Openstack Kilo support for PCIe Passthrough interface**—enables assigning a PCI device to an instance in OpenStack.

The following example explains how to configure hardware acceleration features using flavor datamodel:

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

Undeploying Virtual Network Functions

You can undeploy an already deployed VNF. Use the REST or NetConf / Yang APIs to un-deploy the VNF.

**Important**

Sample undeploy request

```
DELETE /v0/deployments/567 HTTP/1.1
Host: client.host.com
Content-Type: application/xml
Accept: application/xml
Client-Transaction-Id: 123456
Callback:/undeployservicecallback

For more details, see Cisco Elastic Services Controller API Guide.
```
Updating Deployments

Starting from Cisco ESC Release 1.1, you can add or delete a vm_group, add or delete a ephemeral network in a vm_group, and add or delete an interface in a vm_group during deployment.

**Note**

On OpenStack environment you can perform all the updates such as add or delete a vm_group, add or delete a ephemeral network in a vm_group, and add or delete an interface in a vm_group) in a single deployment.

**Adding a VM Group**

You can add or delete a vm_group from a running deployment using existing images and flavors.

**NETCONF request to add a vm_group:**

```xml
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"> <tenants><tenant>
  <name>Admin</name>
  <deployments>
    <deployment>
      <deployment_name>NwDepModel_nosvc</deployment_name>
      <vm_group>
        <image></image>
        <Flavor></Flavor>
        .......... 
      </vm_group>
      <vm_group>
        <image></image>
        <Flavor></Flavor>
        .......... 
      </vm_group>
      <vm_group>
        <image></image>
        <Flavor></Flavor>
        .......... 
      </vm_group>
    </deployment>
  </deployments>
</tenant></tenants>
</esc_datamodel>
```

**NETCONF notification upon successful addition of VM Group:**

UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
VM_DEPLOYED
VM_ALIVE
SERVICE_UPDATED
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)

**Deleting a VM Group**

**NETCONF request to delete a vm_group:**

```xml
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"> <tenants><tenant>
  <name>Admin</name>
  <deployments>
    <deployment>
      <deployment_name>NwDepModel_NoSvc</deployment_name>
      <vm_group>
        <image></image>
        <Flavor></Flavor>
        .......... 
      </vm_group>
    </deployment>
  </deployments>
</tenant></tenants>
</esc_datamodel>
```
Adding an Ephemeral Network in a VM Group

You can add an ephemeral network in a vm_group using existing images and flavors.

NETCONF request to add an ephemeral in a vm_group:

```xml
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"> <tenants><tenant>
    <name>Admin</name>
    <deployments>
        <deployment>
            <deployment_name>NwDepModel_nosvc</deployment_name>
            <networks>
                .......
            </networks>
        </deployment>
        .......
    </deployments>
</tenant></tenants>
</esc_datamodel>
```

NETCONF notification upon successful addition of an ephemeral network in a vm_group:

UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
CREATE_NETWORK
CREATE_SUBNET
SERVICE_UPDATED
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)

Deleting an Ephemeral Network in a VM Group

NETCONF request to delete an ephemeral network in a vm_group:

```xml
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"> <tenants><tenant>
    <name>Admin</name>
    <deployments>
        <deployment>
            <deployment_name>NwDepModel</deployment_name>
            <networks>
                <network nc:operation="delete">
                    .......
                </network>
            </networks>
        </deployment>
        .......
    </deployments>
</tenant></tenants>
</esc_datamodel>
```

NETCONF notification upon successful deletion of vm_group:

UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
VM_UNDEPLOYED
SERVICE_UPDATED
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
Deploying Virtual Network Functions

Updating Deployments

NETCONF notification upon successful deletion of an ephemeral network in a vm_group:

UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
DELETE_SUBNET
DELETE_NETWORK
SERVICE UPDATED
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)

Adding an Interface in a VM Group

You can add an interface in a vm_group from a running deployment using existing images and flavors.

NETCONF request to add an interface in a vm_group:

<interfaces>
  <interface>
    <nicid>0</nicid>
    <network>esc-net</network>
  </interface>
  <interface>
    <nicid>1</nicid>
    <network>utroycho-net</network>
  </interface>
  <interface>
    <nicid>2</nicid>
    <network>utroycho-net-1</network>
  </interface>
</interfaces>

Deleting an Interface in a VM Group

NETCONF request to delete an interface in a vm_group:

<interfaces>
  <interface>
    <nicid>0</nicid>
    <network>esc-net</network>
  </interface>
  <interface>
    <nicid>1</nicid>
    <network>utroycho-net</network>
  </interface>
  <interface nc:operation="delete">
    <nicid>2</nicid>
    <network>utroycho-net-1</network>
  </interface>
</interfaces>
CHAPTER 7

Monitoring Virtual Network Functions

- Monitoring the VNFs, page 57
- Monitoring Methods, page 59
- Monitoring a VM, page 61
- Monitoring Operations, page 62

Monitoring the VNFs

After deploying VNFs, they are monitored periodically to check their health and workload. ESC allows you to define the metrics to be monitored and the actions that need to be executed when the conditions are met. These metrics and actions are defined in the deployment datamodel. Several monitoring methods are used to monitor the VNFs. You can monitor the following:

- VM aliveness
- VM variables for Disk usage, Memory, CPU, Network throughput
- ICMP message on the VM monitoring interface.

Pre-requisites for Monitoring

The following pre-requisites must be met for the VMs to be monitored by ESC:

- Monitoring is enabled for VMs that are successfully deployed. The deployed VMs must be alive.
- KPI must be configured in the data model with the monitoring parameters.

KPIs, Rules and Dynamic Mapping Files in the Datamodel

The Elastic Service Controller allows user to define metrics to be monitored, and actions to be executed when certain conditions are met. These metrics and actions are defined at the time of deployment.

The ESC metrics and actions datamodel is divided into 2 sections:

1. KPI—Defines the type of monitoring, events, polling interval and other parameters. This includes the event_name, threshold and metric values. The event_name is user defined. The metric_values specify threshold conditions and other details. An event is triggered when the threshold condition is reached.
Rule—Defines the actions when the KPI monitoring events are triggered. The action element defines the actions to be performed when an event corresponding to the event_name is triggered.

The monitoring and actions service (MONA) engine performs monitoring activities, and executes actions. A dynamic mapping file supports mapping of KPI and rules to the MONA data model.

Dynamic mapping file structure:

```
<dynamic_mappings>
  <actions>
    <!-- service booted action for backward compatibility with previous script -->
    <action>
      <name>TRUE_servicebooted.sh</name>
      <type>ESC_POST_EVENT</type>
      <metadata>
        : : : : : :
        : : : : : :
      </metadata>
    </action>
    <metrics>
      <metric>
        <name>MEMORY</name>
        <type>MONITOR_COMPUTE_THRESHOLD</type>
        <metadata>
          <properties>
            <property>
              : : : : : :
              <type>snmp_get_threshold</type>
            </property>
          </properties>
        </metadata>
      </metric>
    </metrics>
  </actions>
</dynamic_mappings>
```

Each of the sections in the dynamic mapping file allows administrator to add and remove actions and metrics to the set of supported ESC KPIs and rules. Following are some of the parameters that can be passed to the scripts at the time of execution. Parameter value is populated at runtime only if the parameter is a supported one, and its value is empty within the dynamic-mappings.xml file. Otherwise, the value defined within the script is passed as is. Table below shows the parameters passed during runtime.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>esc_url</td>
<td>The URL of the Elastic Services Controller.</td>
</tr>
<tr>
<td>vm_external_id</td>
<td>The external id of the managed VM.</td>
</tr>
<tr>
<td>vm_name</td>
<td>The name of the managed VM.</td>
</tr>
<tr>
<td>vm_mac_address</td>
<td>The mac address of the managed VM.</td>
</tr>
<tr>
<td>vm_external_host_id</td>
<td>The VM external host Identifier.</td>
</tr>
<tr>
<td>vm_external_host_name</td>
<td>The VM external hot name.</td>
</tr>
<tr>
<td>vm_group_name</td>
<td>The VM group name.</td>
</tr>
<tr>
<td>ip_address</td>
<td>The VM IP Address.</td>
</tr>
<tr>
<td>agent_address</td>
<td>The VM IP Address.</td>
</tr>
<tr>
<td>event_name</td>
<td>The ESC event name.</td>
</tr>
</tbody>
</table>
A sample dynamic mapping file is as follows:

```xml
<action>
  <name>TRUE demo_script_service_booted</name>
  <type>SCRIPT</type>
  <metaData>
    <type>script</type>
    <properties>
      <property>
        <name>script_filename</name>
        <value>/cisco/esc-scripts/pipe.sh</value>
      </property>
      <property>
        <name>esc_url</name>
        <value></value>
      </property>
      <property>
        <name>vm_external_id</name>
        <value></value>
      </property>
      <property>
        <name>vm_name</name>
        <value></value>
      </property>
      <property>
        <name>event_name</name>
        <value>VM_ALIVE</value>
      </property>
      <property>
        <name>esc_event</name>
        <value>SERVICE_BOOTED</value>
      </property>
      <property>
        <name>ip_address</name>
        <value></value>
      </property>
    </properties>
  </metaData>
</action>
```

The name is set to TRUE demo_script_service_booted. The prefix TRUE is maintained to be compliant with the existing ESC datamodel. The action name here denotes a script that the user wants to run at service boot up time. /cisco/esc-scripts/pipe.sh is the script to be executed. Parameters will be added to the empty value field at the time of execution.

**Monitoring Methods**

ESC uses several monitoring methods to monitor the VNFs. You must configure the KPI data model for the monitoring methods.

**ICMP Ping Monitoring**

ICMP Ping monitoring, ICMP pings assess the liveliness or reachability of a VNF.

If the VM is dead then the healing of the VM is triggered. At every defined interval, ESC polls the metric value and sends alarms whenever needed. The number of polls, metric value, and other configuration are set in the KPI datamodel.
SNMP Monitoring

In SNMP Monitoring, load of the VM such as memory usage, CPU in a given period is monitored. The SNMP Get operation is used to assess the liveliness or reachability of a VNF. In this monitoring method, only the success or failure is monitored.

SNMP Threshold Monitoring

In SNMP threshold monitoring, you can set the upper and lower threshold levels in the kpi section of the data model. Actions are performed based on the upper and lower threshold levels.

Custom Monitoring

In script based monitoring, scripts are executed to assess a condition on a VM. Custom scripts are pre-installed on ESC VM. These scripts are executed at defined intervals to assess the liveliness of the VNFs, and also support special operations. The execution of these scripts are controlled by the data model, kpi, metric_occurrences and so on. The script must return 0 for success, and 1 for failure.

For example, to add custom action scripts, include the details in the dynamic mapping file and update your KPI model to reference it.

You can update the dynamic mapping script with

```xml
<action>
    <name>TRUE Sample_Demo_Script</name>
    <type>SCRIPT</type>

......................
</action>
```

Update your KPI model. For example,

```xml
<kpi>
    <event_name>VM_SAMPLE_DEMO_EVENT</event_name>
    <metric_value>100</metric_value>
    <metric_cond>LT</metric_cond>
    <metric_type>UINT32</metric_type>
    <metric_occurrences_true>1</metric_occurrences_true>
    <metric_occurrences_false>1</metric_occurrences_false>
    <metric_collector>
        <type>SUBSCRIBER_SESSION</type>
        <nicid>0</nicid>
        <poll_frequency>15</poll_frequency>
        <polling_unit>seconds</polling_unit>
        <continuous_alarm>false</continuous_alarm>
</metric_collector>
</kpi>
```

The rule section of the same deployment can have the following custom script.

```xml
<rule>
    <event_name>VM_UNDERLOADED_EMPTY_SAMPLE</event_name>
    <action>TRUE Sample_Demo_Script</action>
</rule>
```

Upon execution of the action identified by TRUE Sample_Demo_Script, the custom script defined in the property script_filename is run, and the event VM_UNDERLOADED_EMPTY_SAMPLE is triggered.

```xml
<action>
    <name>TRUE Sample_Demo_Script</name>
    <type>SCRIPT</type>
    <metaData>
        <type>script</type>
        <properties>
            <property>
                <name>script_filename</name>
                <value>/cisco/scripts/sample/sampleScript.sh</value>
            </property>
            <property>
                <name>esc_event</name>
                <value></value>
            </property>
        </properties>
</metaData>
```
Monitoring a VM

ESC monitors the VM to detect any erroneous condition. ESC uses one of its monitoring methods to detect actions on a VM, and passes this information to the rules service for processing. The monitoring request comes from the northbound client along with VNF deployment requests.

There are two sections in the datamodel xml file which define the events and rules: KPI and Rule.

Based on the monitors and actions, rules are triggered.

In the example above, an event is sent to check whether the VM is alive. The VM is pinged at regular intervals, and based on the result VM_ALIVE event is sent to the rules engine along with the details of the VM.

The rules engine receives events from the monitoring engine. The rules engine can handle simple to complex events. Based on the event received an action is triggered.

If the VM is not alive, based on the event the actions defined in the <rule> section are triggered. This can be found in the dep.xml datamodel.

The rules section describes the actions to be executed once a monitoring event has been detected. The dynamic mapping file drives the rules based on keywords.

In the above example, the following actions are performed based on the given condition:

- **ALWAYS log**: Whether the event is pingable or not, the details are logged.
- **TRUE servicebooted.sh**: The action identified by this keyword in the dynamic mapping file will be triggered when the VM moves from a non-pingable to a pingable state. The serviceboot script informs ESC that the VM is "alive" allowing it to transition the VMs state.
- **FALSE recover autohealing**: The action identified by this keyword will be triggered and the VM will be recovered without the administrator's intervention.

Monitoring log files for troubleshooting are available at `/var/log/mona`.  

---

Cisco Elastic Services Controller 2.1 User Guide 61
Monitoring Operations

You can set and unset monitoring of VMs using RESTful interface.

A payload is required to monitoring VMs:

```
POST ESCManager/v0/[internal_tenant_id]/deployments/vm/[vm_name]
```

Example,

```xml
<?xml version='1.0' encoding='UTF-8'?>
<vm_operation xmlns='urn:ietf:params:xml:ns:netconf:base:1.0'>
  <operation>enable_maintenance</operation>
  <force>false</force>
</vm_operation>
```

You must mention enable maintenance to set VM monitoring, and disable maintenance to unset VM monitoring in the operation field.
Scaling Virtual Network Functions

- Scaling Overview, page 63
- Scale In and Scale Out of VMs, page 63
- Scaling Notifications and Events, page 65

Scaling Overview

ESC is capable of elastically scaling the service. It can be configured to do both scale in and scale out automatically. The scaling is achieved using KPI, rules and actions. These are configured during deployment. The KPI define the event name and threshold. The rules define action to trigger scale out and scale in.

Scale In and Scale Out of VMs

Scaling workflow begins after successful deployment of a VNF. VMs are configured to monitor attributes such as CPU load, memory usage, and so on, which form the KPI data in the data model. If for any attributes, KPI reaches its threshold, based on the action defined, scale in and scale out is performed.

- During scale out, if the number of VMs is less than maximum active, a new VM deployment is triggered.
- During scale in, if the number of VMs is greater than the minimum active, the VM will be undeployed.

Note

If the VM is deployed and did not receive the VM alive event, then recovery will be triggered. Any error during undeployment will be notified to the northbound user.

In the scaling section of the datamodel, the minimum and maximum values are configured. The min_active defines the number of VMs deployed. The max_active defines the number of maximum VMs that can be deployed. For example, if a VNF is deployed with minimum 2 VMs and a maximum of 100 VMs, the below xml will define scaling under each VM group.
Starting from ESC Release 2.0, if the primary VM was configured using a static IP address, the scaled up
VMs must be assigned a static IP address. During deployment, a list of static IP addresses must be specified.
The following example explains how create a static IP pool:

```xml
<scaling>
  <min_vms>1</min_vms>
  <max_vms>2</max_vms>
  <elastic>true</elastic>
  <static_ip_address_pool>
    <network>1234-5678-9123</network>
    <gateway>10.86.22.1</gateway>
    <netmask>255.255.255.0</netmask>
    <ip_address>10.86.22.227</ip_address>
    <ip_address>10.86.22.228</ip_address>
  </static_ip_address_pool>
</scaling>
```

The following example explains the method of detecting the CPU load in the KPI data section.

```xml
<kpi>
  <event_name>VM_OVERLOADED</event_name>
  <metric_value>70</metric_value>
  <metric_cond>GT</metric_cond>
  <metric_type>UINT32</metric_type>
  <metric_occurrences_true>2</metric_occurrences_true>
  <metric_occurrences_false>4</metric_occurrences_false>
  <metric_collector>
    <type>CPU_LOAD_1</type>
    <nicid>0</nicid>
    <poll_frequency>3</poll_frequency>
    <polling_unit>seconds</polling_unit>
    <continuous_alarm>false</continuous_alarm>
  </metric_collector>
</kpi>

<kpi>
  <event_name>VM_UNDERLOADED</event_name>
  <metric_value>40</metric_value>
  <metric_cond>LT</metric_cond>
  <metric_type>UINT32</metric_type>
  <metric_occurrences_true>2</metric_occurrences_true>
  <metric_occurrences_false>4</metric_occurrences_false>
  <metric_collector>
    <type>CPU_LOAD_1</type>
    <nicid>0</nicid>
    <poll_frequency>3</poll_frequency>
    <polling_unit>seconds</polling_unit>
    <continuous_alarm>false</continuous_alarm>
  </metric_collector>
</kpi>
```

KPI rules are as follows:

```xml
<rule>
  <event_name>VM_OVERLOADED</event_name>
  <action>ALWAYS log</action>
  <action>TRUE servicescaleup.sh</action>
</rule>

<rule>
  <event_name>VM_UNDERLOADED</event_name>
  <action>ALWAYS log</action>
  <action>TRUE servicescaledown.sh</action>
</rule>
```
Scaling Notifications and Events

The scaling notifications are sent to the northbound users. The notification includes status message and other details to identify the service that is undergoing scaling. Below is the list of notifications:

- VM_SCALE_OUT_INIT
- VM_SCALE_OUT_DEPLOYED
- VM_SCALE_OUT_COMPLETE
- VM_SCALE_IN_INIT
- VM_SCALE_IN_COMPLETE

The following table lists the scaling scenarios and the notifications that are generated:

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Notifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale Out</td>
<td>ESC deploys VMs and sets KPI/Monitors and all VM Alives received. The following NETCONF notification is triggered.</td>
</tr>
<tr>
<td></td>
<td>When ESC receives a VM_OVERLOADED event, the following NetConf notification is triggered:</td>
</tr>
<tr>
<td></td>
<td>ESC checks if the max limit is reached, if not, it deploys a new VM.</td>
</tr>
<tr>
<td></td>
<td>Once the deployment is complete, the following Netconf Notification is sent,</td>
</tr>
<tr>
<td>Scale In</td>
<td>ESC deploys VMs and sets KPI/Monitors and all VM Alives received.</td>
</tr>
<tr>
<td></td>
<td>Netconf Notification Sent</td>
</tr>
<tr>
<td></td>
<td>When ESC receives a VM_UNDERLOADED event, the following NetConf notification is triggered</td>
</tr>
<tr>
<td></td>
<td>ESC checks if number of VM is more than minimum active limit, if so, it undeploys one of the VM after undeployment is complete, Netconf Notification Sent.</td>
</tr>
</tbody>
</table>

For all the error scenarios, the notification will be sent with FAILURE status. Also status message should have the corresponding failure details.
Healing Virtual Network Functions

- Healing Overview, page 67
- Healing a VM, page 68
- Notifications and Events, page 69

Healing Overview

As part of life cycle management, ESC heals the VNFs when there is a failure. The failure scenarios are configured in the KPI section of the datamodel. ESC uses KPI to monitor the VM and the events are triggered based on the KPI conditions. The actions to be taken for every event that is triggered are configured in the rules section during the deployment.

Starting from ESC Release 2.0 and onwards, you can configure the recovery policy in the deployment datamodel. The recovery policy for a VM can be set to one of the following:

- **Reboot and Redeploy (default)**—When a VM down event is received, the healing workflow firstly attempts to reboot the VM, if it fails to reboot, then it attempts to redeploy the VM.
- **Reboot**—When a VM down event is received, the healing workflow only attempts to reboot the VM.
- **Redeploy**—When a VM down event is received, the healing workflow only attempts to redeploy the VM.

The *action_on_recovery* parameter in the datamodel specifies the type of recovery policy. The following example explains how to set the recovery policy to REBOOT_THEN_REDEPLOY.

```xml
<vm_group>
    ......
    <recovery_policy>
        <action_on_recovery>
            REBOOT_THEN_REDEPLOY
        </action_on_recovery>
    </recovery_policy>
</vm_group>
```

For more information about setting the recovery policy in the datamodel, see Elastic Services Controller Deployment Attributes. For more information about setting the recovery policy in the ESC portal (VMware only), see the Deploying Virtual Network Functions in ESC Portal (VMware Only), on page 41.
Healing a VM

Each VM group is configured to enable the healing. Healing is performed at various stages and in various ways depending on the state of the VM.

The VMs are deployed and are being monitored. After ESC receives a VM Alive event, if it receives a VM Down event, the healing workflow attempts to reboot the VM. If the reboot does not recover the VM, the VM is undeployed and redeployed.

If ESC does not receive a VM Alive after deployment, it undeploys and redeploy the VM when timeout happens. All the deploy and undeploy procedures will depend on the recovery policy configuration. For example, if the user configured either one of the recovery policy such as Reboot Only, Redeploy Only, or Reboot and Redeploy then ESC will follow the same configured policy.

ESC provides YANG based data model with comprehensive details of all the parameters and description that is needed to define the healing. ESC uses two sections in the data model xml file which define the events and rules:

• <kpi> section defines the type of monitoring, events, polling interval and other parameters.

• <rule> section defines the actions when the KPI monitoring events are triggered.

For more information on KPI, rules, and data model, see Monitoring the VNFs, on page 57.

The configuration involves the following steps:

1 Define kpi

2 Define rules

The following example shows how to configure the KPI in the data model:

```
<kpi>
  <event_name>VM_ALIVE</event_name>
  <metric_value>1</metric_value>
  <metric_cond>GT</metric_cond>
  <metric_type>UINT32</metric_type>
  <metric_collector>
    <type>ICMP Ping</type>
    <nicid>0</nicid>
    <poll_frequency>3</poll_frequency>
    <polling_unit>seconds</polling_unit>
    <continuous_alarm>false</continuous_alarm>
  </metric_collector>
</kpi>
```

The following example shows how to configure the rules for every event:

```
<rules>
  <admin_rules>
    <rule>
      <event_name>VM_ALIVE</event_name>
      <action>ALWAYS log</action>
      <action>FALSE recover autohealing</action>
      <action>TRUE servicebooted.sh</action>
    </rule>
  </admin_rules>
</rules>
```

In the above examples, we define a KPI to monitor the ICMP Ping on the nicid 0. It defines the attributes metric condition and polling. Based on the KPI, the VM_ALIVE event is triggered with appropriate values. The action in the corresponding rule defines what the next steps are:
• FALSE—Triggers recovery of the VM.
• TRUE—Triggers the defined action.

If recovery is triggered on the VM, ESC reboots the VM as the first step to recover the VM. If it fails, the VM is un-deployed and a new VM with same day-0 configuration is deployed. ESC tries to reuse the same network configuration like MAC and IP Address as the previous VM.

Notifications and Events

The following notifications are generated by the ESC during healing:

• VM_RECOVERY_INIT
• VM_RECOVERY_DEPLOYED
• VM_RECOVERY_UNDEPLOYED
• VM_RECOVERY_COMPLETE

These notifications are generated based on the workflow. Each notification will have details about the deployment for which the notification is triggered. All recovery starts with VM_RECOVERY_INIT and ends with VM_RECOVERY_COMPLETE.

The following table lists the different scenarios and the notifications that are generated for every event:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Notifications</th>
</tr>
</thead>
</table>
| ESC-NORTHBOUND Recovery Call Flow After VM Alive - Reboot | When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alive received. The following NETCONF notification is triggered:  

```
<type>SERVICE_ALIVE</type>
<status>SUCCESS</status>
```

After ESC receives VM down event, the following NETCONF notification is triggered:

```
<type>VM_RECOVERY_INIT</type>
<status>SUCCESS</status>
```

ESC performs hard reboot on the VM, and the VM alive event is received within the boot time.

```
<type>VM_RECOVERY_COMPLETE</type>
<status>SUCCESS</status>
```
## Scenario

### ESC-NORTHBOUND Recovery Call Flow After VM Alive - Undeploy/Redeploy

When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alive received. The following NETCONF notification is triggered:

```
<type>SERVICE_ALIVE</type>
<status>SUCCESS</status>
```

After ESC receives VM down event, the following NETCONF notification is triggered:

```
<type>VM_RECOVERY_INIT</type>
<status>SUCCESS</status>
```

ESC fails to recover the VM by **Reboot** and proceeds with recovery by **Undeploy** and then **Redeploy**.

It unsets monitoring and un-deploys the VM.

The following NETCONF notification is triggered:

```
<type>VM_RECOVERY_UNDEPLOYED</type>
<status>SUCCESS</status>
```

ESC deploys VM and sets KPI to monitor VM Alive event and triggers the following NETCONF notifications:

```
<type>VM_RECOVERY_DEPLOYED</type>
<status>SUCCESS</status>
```

ESC receives a VM Alive event and triggers the following NETCONF notifications:

```
<type>VM_RECOVERY_COMPLETE</type>
<status>SUCCESS</status>
```
When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alive received. The following NETCONF notification is triggered:

```xml
<type>SERVICE_ALIVE</type>
<status>SUCCESS</status>
```

After ESC receives VM down event, the following NETCONF notification is triggered:

```xml
<type>VM_RECOVERY_INIT</type>
<status>SUCCESS</status>
```

ESC fails to recover the VM by Undeploy and then ReDeploy until it receives a VM Alive event. It keeps attempting the recovery for a specified boot time until the maximum attempts of recovery is reached.

It un-sets monitoring and un-deploys the VM.

The following NETCONF notification is triggered:

```xml
<type>VM_RECOVERY_UNDEPLOYED</type>
<status>SUCCESS</status>
```

ESC deploys VM and sets KPI to monitor VM Alive event.

The following NETCONF notifications is triggered:

```xml
<type>VM_RECOVERY_DEPLOYED</type>
<status>SUCCESS</status>
```

ESC receives a VM Alive event and triggers the following NETCONF notifications:

```xml
<type>VM_RECOVERY_COMPLETE</type>
<status>SUCCESS</status>
```
### Notifications

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Notifications</th>
</tr>
</thead>
</table>
| ESC-NORTHBOUND Recovery Call Flow Before VM Alive - Undeploy/Redeploy | When Northbound places a deploy request to ESC, ESC deploys VMs and sets KPI to monitor on all VM Alive received. ESC does not receive a VM Alive event after the deployment. Recovery is performed by **Undeploying** and **Redeploying** the VM. The following NETCONF notification is triggered:  
```
<type>VM_RECOVERY_INIT</type>  
<status>SUCCESS</status>
```
ESC un-sets the monitoring and un-deploys the VM. The following NETCONF notification is triggered:  
```
<type>VM_RECOVERY_UNDEPLOYED</type>  
<status>SUCCESS</status>
```
ESC deploys VM and sets KPI to monitor VM Alive event and triggers the following NETCONF notifications:  
```
<type>VM_RECOVERY_DEPLOYED</type>  
<status>SUCCESS</status>
```
ESC receives a VM Alive event and triggers the following NETCONF notifications:  
```
<type>VM_RECOVERY_COMPLETE</type>  
<status>SUCCESS</status>
```
| Error Path For ESC-NORTHBOUND Recovery Call Flow After VM Alive - Reboot | When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alive received. The following NETCONF notification is triggered:  
```
<type>SERVICE_ALIVE</type>  
<status>SUCCESS</status>
```
After ESC receives VM down event, the following NETCONF notification is triggered:  
```
<type>VM_RECOVERY_INIT</type>  
<status>SUCCESS</status>
```
ESC receives an error while attempting to recover through **Reboot**. The following NETCONF notification is triggered:  
```
<type>VM_RECOVERY_COMPLETE</type>  
<status>FAILURE</status>
```
### Notifications and Events

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Notifications</th>
</tr>
</thead>
</table>
| **Error Path For ESC-NORTHBOUND Recovery Call Flow After VM Alive - Undeploy/ReDeploy** | When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alives received. The following NETCONF notification is triggered:  
  `<type>SERVICE_ALIVE</type>`  
  `<status>SUCCESS</status>`  
  After ESC receives VM down event, the following NETCONF notification is triggered:  
  `<type>VM_RECOVERY_INIT</type>`  
  `<status>SUCCESS</status>`  
  ESC fails to recover the VM by Reboot and proceeds with recovery by Undeploy and then Redeploy.  
  It un-sets monitoring and un-deploys the VM.  
  The following NETCONF notification is triggered:  
  `<type>VM_RECOVERY_UNDEPLOYED</type>`  
  `<status>SUCCESS</status>`  
  If ESC receives an error or if the maximum attempts for recovery is reached.  
  The following NETCONF notifications is triggered:  
  `<type>VM_RECOVERY_COMPLETE</type>`  
  `<status>FAILURE</status>`                                                                 |
| **Error Path For ESC-NORTHBOUND Recovery Call Flow Before VM Alive - Undeploy/Redeploy** | When Northbound places a deploy request to ESC, ESC deploys VMs and set KPI to monitor on all VM Alives received. The following NETCONF notification is triggered:  
  `<type>SERVICE_ALIVE</type>`  
  `<status>SUCCESS</status>`  
  After ESC receives VM down event, the following NETCONF notification is triggered:  
  `<type>VM_RECOVERY_INIT</type>`  
  `<status>SUCCESS</status>`  
  ESC un-sets monitoring and un-deploys the VM. Recovery is performed by Undeploy and then Redeploy.  
  The following NETCONF notification is triggered:  
  `<type>VM_RECOVERY_UNDEPLOYED</type>`  
  `<status>SUCCESS</status>`  
  If ESC receives an error or if the maximum attempts for recovery is reached.  
  The following NETCONF notifications is triggered:  
  `<type>VM_RECOVERY_COMPLETE</type>`  
  `<status>FAILURE</status>`  
  `<type>SERVICE_ALIVE</type>`  
  `<status>FAILURE</status>` |

---

Healing Virtual Network Functions
Virtual Network Function Operations

VNF Operations

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

A payload is required for VNF operations:
POST ESCManager/v0/{internal_tenant_id}/deployments/service/{internal_deployment_id}

Example,
```xml
<?xml version='1.0' encoding='UTF-8'?>
<service_operation xmlns='urn:ietf:params:xml:ns:netconf:base:1.0'>
  <operation>stop</operation>
</service_operation>
```

You must mention start, stop or reboot in the operation field.

- Start VNF: Starts all VMs, enables monitoring, and reassigns thresholds according the KPI details. The VMs start running and move to VM_ALIVE_STATE. The service will be in service_active state. Only undeploy can interrupt the start VNF workflow.

- Stop VNF: Once the service is stopped, monitoring is disabled and all the VM services are stopped. The VMs are no longer available. The service will be in service_stopped_state. VM will be in shutoff_state. You cannot perform any recovery, scale up, scale down. You can only undeploy the VNFs.

- Reboot VNF: Disables monitoring, reboots all VMs, that is stop and then start in OpenStack, enables monitoring, and reassigns thresholds according to KPI details. The VM is in VM_ALIVE_STATE and the service is in service_alive_state. Only undeploy can interrupt the reboot operation.

You cannot start monitoring a VNF which is already running. After a reboot, logging back into the VM must indicate the reboot, update and monitoring details. It must also indicate recovery.

VM Operations

Similar to VNF operations, you can start, stop and reboot individual VMs.

A payload is required for VM operations:
POST ESCManager/v0/{internal_tenant_id}/deployments/vm/{vm_name}
Example,

```xml
<?xml version='1.0' encoding='UTF-8'?>
<vm_operation xmlns='urn:ietf:params:xml:ns:netconf:base:1.0'>
  <operation>stop</operation>
  <force>true/false</force>
</vm_operation>
```

You must mention start, stop or reboot in the operation field.
Error Handling

Error Conditions for ESC Operations

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

Notification or Logging details for Error Conditions

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

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

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

### ESC System Logs

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

The log message format is as follows:

```
timestamp threadname loglevel [filename:methodname():line#] [transactionId] [classification] [tags] message
```

The following classifications are supported:
• **SM**—stands for StateMachine. This classification indicates logs in the StateMachine category.

• **REST_EVENT**—indicates REST EVENTS in logs. This is used by ESC UI.

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 log analysis tools.

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