Elastic Services Controller Interfaces

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 Services Orchestrator (NSO), and available within Cisco Solutions such as Cisco Managed Services Accelerator (MSX).
- 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 MSX, 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. 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

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.

ESC also supports a REST API that conforms to the ETSI NFV MANO standards. Note that when using the ETSI API, the other ESC interfaces should be used for reading information only to preserve the integrity of the data models. The details of this API is outside the scope of this document. See the Cisco Elastic Services Controller ETSI NFV MANO User Guide for more information.

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.

Note You can just type *esc_nc_cli --user <username> --password <password> command <file name>* instead of the complete path for any CRUD operations using the netconf CLI. For more information on CLI, see the *Cisco Elastic Services Controller Install and Upgrade Guide*.

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.

You can create a single NETCONF request to perform multiple actions. For more details, see Netconf Enhancement Request. The following is a NETCONF request to delete two tenants simultaneously:

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

Examples of NETCONF/YANG API are as follows:

NETCONF request to create a Tenant,

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <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:

```
<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 />
    <event>
      <type>CREATE_TENANT</type>
    </event>
  </escEvent>
</notification>
```

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

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

NETCONF response,

```
<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>
    </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.

The NETCONF API configuration and RPC calls are validated. If the request is not valid, it is rejected. The NETCONF API does not send any error code to NB, unlike REST (for example, REST sends 404 not found error).

A sample error message (rejected request) is as follows

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-path xmlns:esc="http://www.cisco.com/esc/esc"

xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">/nc:rpc/esc:filterLog</error-path>
        <error-message xml:lang="en">Exception from action callback: Error when handling
RPC
            calls: You can only query up to 30 logs.</error-message>
        <error-info>
            <bad-element>filterLog</bad-element>
        </error-info>
```

```
    </rpc-error>
</rpc-reply>
```

The no_gateway attribute allows ESC to create a subnet with the gateway disabled.

In the example below, the no_gateway attribute is set to true to create a subnet without gateway.

```
<networks>
    <network>
        <name>mgmt-net</name>
        <subnet>
            <name>mgmt-net-subnet</name>
            <ipversion>ipv4</ipversion>
            <dhcp>false</dhcp>
            <address>10.0.0.0</address>
            <no_gateway>true</no_gateway>
            <!-- DISABLE GATEWAY -->
            <gateway>10.0.0.1</gateway>
            <netmask>255.255.255.0</netmask>
        </subnet>
    </network>
</networks>
```

ESC shows OpenStack and VMware vCenter username in its Operational Data section.

The following configuration details are displayed in the Operational Data for,

OpenStack

- active_vim—displays the value as OpenStack
- os_auth_url—displays the OpenStack authentication URL
- admin_role—displays if the OpenStack user is an admin
- os_tenant_name—displays the tenant
- os_username—displays the Openstack user
- member_role—displays if the OpenStack user is a member

VMware vCenter

- active_vim—displays the value as VMware
- vcenter_ip—displays the vCentre IP address
- vcenter_port—displays if the vCentre port
- vcenter_username—displays the vCentre user

NETCONF Request to Configure Multiple Resources

A user can create a single NETCONF request to configure multiple resources.

Note A single request to configure multiple resources is supported using NETCONF only.

A single NETCONF request associates multiple resources based on the dependencies between the resources. For example, a subnet is dependent on a network, and a deployment is dependent on the image and flavor.

There are 2 types of dependencies in ESC.

1. Referential Dependency
2. Hierarchical Dependency

Referential Dependency

In referential dependency, one configuration has a reference to another configuration.

In the example below, deployment has referential dependency on image (test-mix-cirros) and flavor (test-mix-small). The image and flavor must be created before the deployment configuration.

```
  <images>
    <image>
        <name>test-mix-cirros</name>
...
</image>
</images>
<flavors>
    <flavor>
        <name>test-mix-small</name>
...
</flavor>
</flavors>
<tenants>
    <tenant>
        <name>test-mix-tenant</name>
        <deployments>
            <deployment>
                <name>dep</name>
                <vm_group>
                    <name>Group1</name>
                    <image>test-mix-cirros</image>
                    <flavor>test-mix-small</flavor>
...
</vm_group>
            </deployment>
        </deployments>
    </tenant>
</tenants>
```

Hierarchical Dependency

In hierarchical dependency, one configuration is within another configuration.

In the example below, the subnet (test-mix-shared-subnet1) is within the network (test-mix-shared-net1). The subnet has a hierarchical dependency on the network.

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
<networks>
    <network>
        <name>test-mix-shared-net1</name>
        <shared>true</shared>
        <admin_state>true</admin_state>
        <subnet>
            <name>test-mix-shared-subnet1</name>
            <ipversion>ipv4</ipversion>
            <dhcp>true</dhcp>
            <address>10.193.90.0</address>
```

```
                <netmask>255.255.255.0</netmask>
                <gateway>10.193.90.1</gateway>
            </subnet>
        </network>
    </networks>
    </esc_datamodel>
```

A hierarchical dependency is a subset of referential dependency. These configuration dependencies of the resources allow NETCONF to perform multiple configurations using a single request.

REST Northbound API

The REST API is a programmatic interface to 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 (MIT).
- Methods—Actions that the API performs on one or more objects.
- Types—Object properties that map values to the object state (for example, equipmentPresence).

The ESC REST API contains headers, and other parameters. The header parameter contains a callback field with a URI. The client callback expects this value. A callback will not be performed if the URI field is not present.

REST API Documentation

You can access the REST API documentation directly from the ESC VM:

```
http://[ESC VM IP]:8080/ESCAPI
```

For detailed information, you can also see the Cisco Elastic Services Controller API Guide.

The REST API documentation provides details about all the various operations supported through the REST interface.

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>tenant1</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>234243490854004</external_tenant_id>
   <internal_tenant_id>434344896854965</internal_tenant_id>
   <name>tenant1</name>
   <enabled>true</enabled>
   <description>A description...</description>
</tenant>
```

You cannot deploy VNFs with the same tenant name and deployment name using the REST API.

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

ETSI NFV MANO Northbound API

The ETSI NFV MANO API (ETSI API) is another programmatic interface to ESC that uses the REST architecture. The ETSI MANO adheres to the standards defined by the European Telecommunications Standards Institute (ETSI), specifically around Management and Orchestration (MANO).

For more information see, the ETSI NFV MANO Northbound API Overview in the *Cisco Elastic Services Controller ETSI NFV MANO Guide*.

ETSI API Documentation

You can access the ETSI API documentation directly from the ESC VM:

```
http://[ESC VM IP]:8250/API
```

The ETSI API documentation provides details about all the various operations supported through the ESTI MANO interface.You can also see the Cisco ETSI API Guide for more information.

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 on OpenStack, VMware vCenter or KVM. For more information on enabling or disabling the ESC portal, see ESC Portal Dashboard.

To start, stop and restart the ESC Portal, do the following:

- To start the ESC portal, run *sudo escadm portal start*

- To stop the portal, run *sudo escadm portal stop*
- To restart the portal, run *sudo escadm portal restart*

Note The recommended browser screen size is 1920 pixels by 1080 pixels.