Managing ESC Resources

Managing VIM Connectors

A VIM connector contains details such as URL and authentication credentials, which enables ESC to connect and communicate with the VIM. ESC connects to more than one VIM if the VIM connectors are configured. You can configure the VIM connector and its credentials in two ways:

- At the time of installation using the bootvm.py parameters—Only a single VIM connector can be configured using bootvm.py, which becomes the default VIM connector.
- Using the VIM Connector APIs—The VIM connector API allows you to add multiple VIM connectors. You can configure a default VIM connector (if it is not already configured using the bootvm.py parameters), and additional VIM connectors.

The default VIM connector connects ESC to the default VIM. Each VIM in a multi VIM deployment is configured with a VIM connector. These VIMs are non-default VIMs. ESC creates and manages resources on a default VIM. Only deployments are supported on a non-default VIM.

For a single VIM deployment, a single configured VIM connector becomes the default VIM connector. For a multiple VIM deployment, you need to add multiple connectors, and specify one connector as default using the default VIM connector API. For more information, see Deploying VNFs on Multiple OpenStack VIMs.

Note ESC accepts the northbound configuration request to create, update, or delete a resource, or a deployment only if the following conditions are met:

- ESC has the target VIM/VIMs and corresponding VIM user configured.
- ESC is able to reach the target VIM/VIMs.
- ESC is able to authenticate the VIM user.

Configuring the VIM Connector

You can configure the VIM Connector during or after installation.

Configuring the VIM Connector During Installation

To configure the VIM Connector during installation, the following parameter must be provided to bootvm.py:

Environment variables	bootvm.py arguments
OS_TENANT_NAME	--os_tenant_name
OS_USERNAME	--os_username
OS_PASSWORD	--os_password
OS_AUTH_URL	--os_auth_url

Configuring the VIM Connector After Installation

To configure the VIM Connector after installation, the following parameter must be provided to bootvm.py:

```
--no_vim_credentials
```

When the no_vim_credentials parameter is provided, the following bootvm.py arguments are ignored:

- os_tenant_name
- os_username
- os_password
- os_auth_url

For details on Installation, see the Cisco Elastic Services Controller Install and Upgrade Guide. You can configure the same using the VIM Connector APIs post installation, for more details, see Managing VIM Connector Using the VIM Connector APIs, on page 3.

Default VIM Connector

The default VIM connector API allows you to specify a default VIM connector when multiple connectors are available in a deployment.

For a Single VIM deployment, ESC supports a single VIM connector. This single VIM connector becomes the default VIM connector. ESC supports multiple VIM connectors for multi VIM deployments. You can configure the default VIM connector using the new locator attribute. If you are using the ESC Release 2.x datamodel for deployments and creating resources, then configure the default VIM connector explicitly in ESC.

The locator attribute is introduced in the data model for deploying VMs on non-default VIMs. For more details, see Deploying VNFs on Multiple OpenStack VIMs.

While deploying, if the VIM connectors are available, but the default connector is not yet configured, then it is mandatory that you specify the locator attribute else the request is rejected.

The data model prior to ESC Release 3.0 cannot be used if the default VIM connector is not configured. While upgrading from ESC Release 2.x to ESC Release 3.0 and later, the existing VIM connector is provisioned as the default VIM connector.

Note You cannot change or delete the default VIM connector to a different one once configured.

You must specify the default connector at the top level (or beginning) of the data model. The data model is as follows:

```
<esc_system_config>
    <vim_connectors>
        <default_vim_connector>vim1</default_vim_connector>
        <vim_connector>
            <id>vim1</id>
...
</vim_connector>
        <vim_connector>
            <id>vim2</id>
...
</vim_connector>
    </vim_connectors>
</esc_system_config>
```

To add the default VIM connector using the REST API,

```
<?xml version="1.0"?>
<default_vim_connector xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <defaultVimConnectorId>tb3_v3</defaultVimConnectorId>
</default_vim_connector>
```

To add a VIM connector at the time of installation, see Configuring the VIM Connector During Installation in Configuring the VIM Connector, on page 2. The VIM connectors allow multiple VIMs to connect to ESC. For more details on multi VIM deployment, see Deploying VNFs on Multiple OpenStack VIMs.

Deleting VIM Connector

ESC creates SystemAdminTenant automatically when the default VIM connector is created and configured. The SystemAdminTenant cannot be deleted. The VIM is connected and the VIM user is authenticated to the system admin tenant. Hence, the default VIM cannot be deleted or updated. However, the VIM user and its properties can be deleted or updated. You can update and delete the non-default VIM connectors if there are no resources created on the VIM from ESC. If there are resources created on the VIM through ESC, then you must first delete the resources, and then the VIM user to delete the VIM connector.

Managing VIM Connector Using the VIM Connector APIs

If ESC was deployed without passing VIM credentials, you can set the VIM credentials through ESC using the VIM connector and VIM User APIs (REST or Netconf API). Even if the default VIM connector is configured during installation, the additional VIM connectors can be configured using the VIM connector APIs.

Managing using Netconf API

- Passing VIM credential using Netconf:

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <!--represents a vim-->
    <vim_connector>
      <!--unique id for each vim-->
      <id>my-ucs-30</id>
      <!--vim type [OPENSTACK|VMWARE_VSPHERE|LIBVIRT|AWS|CSP]-->
      <type>OPENSTACK</type>
      <properties>
        <property>
          <name>os_auth_url</name>
          <value>http://{os_ip:port}/v3</value>
        </property>
        <!-- The project name for openstack authentication and authorization -->
        <property>
          <name>os_project_name</name>
          <value>vimProject</value>
        </property>
        <!-- The project domain name is only needed for openstack v3 identity api -->
        <property>
          <name>os_project_domain_name</name>
          <value>default</value>
        </property>
        <property>
          <name>os_identity_api_version</name>
          <value>3</value>
        </property>
      </properties>
      <users>
        <user>
          <id>admin</id>
          <credentials>
            <properties>
              <property>
                <name>os_password</name>
                <value>********</value>
              </property>
              <!-- The user domain name is only needed for openstack v3 identity api
-->
              <property>
                <name>os_user_domain_name</name>
                <value>default</value>
              </property>
            </properties>
          </credentials>
        </user>
      </users>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

- Updating VIM Connector using Netconf:

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

```
      <type>OPENSTACK</type>
      <properties>
        <property>
          <name>os_auth_url</name>
          <value>{auth_url}</value>
        </property>
        <property>
          <name>os_project_name</name>
          <value>vimProject</value>
        </property>
        <!-- The project domain name is only needed for openstack v3 identity api -->
        <property>
          <name>os_project_domain_name</name>
          <value>default</value>
        </property>
        <property>
          <name>os_identity_api_version</name>
          <value>3</value>
        </property>
      </properties>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

- Updating VIM user using Netconf:

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <vim_connector>
      <id>example_vim</id>
        <users>
          <user nc:operation="replace">
            <id>my_user</id>
            <credentials>
             <properties>
              <property>
                <name>os_password</name>
                <value>********</value>
              </property>
              <!-- The user domain name is only needed for openstack v3 identity api
-->
              <property>
                <name>os_user_domain_name</name>
                <value>default</value>
              </property>
             </properties>
           </credentials>
          </user>
        </users>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

- Deleting VIM connector using Netconf:

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

- Deleting VIM User using Netconf:

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

- Deleting VIM Connector using command:

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

- Deleting VIM user using command:

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

Managing using REST API

- Adding VIM using REST:

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

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

- Adding VIM user using REST:

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

<?xml version="1.0"?>
<user xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
  <id>my_user</id>
  <credentials>
    <properties>
      <property>
        <name>os_password</name>
        <value>********</value>
      </property>
      <!-- The user domain name is only needed for openstack v3 identity api -->
      <property>
        <name>os_user_domain_name</name>
        <value>default</value>
      </property>
    </properties>
  </credentials>
</user>
```

- Updating VIM using REST:

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

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

- Updating VIM user using REST:

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

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

```
        <name>os_user_domain_name</name>
        <value>default</value>
      </property>
    </properties>
  </credentials>
</user>
```

- Deleting VIM using REST:

```
DELETE /ESCManager/v0/vims/{vim_id}
```

- Deleting VIM user using REST:

```
DELETE /ESCManager/v0/vims/{vim_id}/vim_users/{vim_user_id}
```

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

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

For more information on the APIs, see Cisco Elastic Services Controller API Guides.

Important Notes:

- You can add more than one VIM connector, but all the VIM connectors must have the same VIM type. Multiple VIM connectors can be added for OpenStack VIM only. However, only one VIM user can be configured per VIM connector.
- os_project_name and os_project_domain_name properties specify the OpenStack project details for authentication and authorization under the VIM connector properties. If the os_tenant_name property exists under the Vim User, it will be ignored.
- The VIM connector properties os_auth_url and os_project_name and VIM User property os_password are mandatory properties for the OpenStack VIM. If these properties are not provided, then the request to create the VIM connector is rejected.
- VIM username and password can be updated anytime. VIM endpoint cannot be updated while resources created through ESC exist.
- The name of a VIM property or VIM user credentials property are not case sensitive, e.g. OS_AUTH_URL and os_auth_url is the same to ESC.

You can encrypt the VIM connector credentials by replacing the existing <value> field with <encrypted_value>.

For example,

```
<credentials>
  <properties>
    <property>
      <name>os_password</name>
```

```
      <encrypted_value>********</encrypted_value>
    </property>
    <property>
      <name>os_user_domain_name</name>
      <value>default</value>
    </property>
  </properties>
</credentials>
```

This stores the os_value password as an aes-cfb-128-encrypted-string in the CFB using the keys contained in /opt/cisco/esc/esc_database/esc_production_confd.conf.

Note The existing value must be replaced with encrypted value only within the credentials specified.

For more information, see Encrypting Configuration Data.

VIM Connector Status API

The table below shows the VIM connector status and a status message for each VIM connector. The status shows ESC connection and authentication status of the VIM.

VIM Reachability	User Authentication	Status (by ESC)	Status Message
NOT REACHABLE	-	CONNECTION_FAILED	Unable to establish VIM connection
REACHABLE	VIM user is not configured	NO_CREDENTIALS	No VIM user credentials found
REACHABLE	Authentication failed	AUTHENTICATION_FAILED	VIM authentication failed
REACHABLE	Authentication successful	CONNECTION_SUCCESSFUL	Successfully connected to VIM

Status using the REST API

HTTP Operation: GET

Path: `ESCManager/v0/vims, ESCManager/v0/vims/<specific_vim_id>`

Sample REST Response is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vim_connector xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <properties>
        <property>
            <name>os_auth_url</name>
            <value>http://10.85.103.37:5000/v2.0/</value>
        </property>
    </properties>
    <id>default_openstack_vim</id>
    <status>CONNECTION_SUCCESSFUL</status>
    <status_message>Successfully connected to VIM</status_message>
    <type>OPENSTACK</type>
</vim_connector>
```

Status using the NETCONF API

The opdata shows the status. The VIM connector status is within the vim connector container.

Sample opdata is as follows:

```
<system_config>
    <active_vim>OPENSTACK</active_vim>
    <openstack_config>
        <os_auth_url>http://10.85.103.37:5000/v2.0/</os_auth_url>
        <admin_role>admin</admin_role>
        <os_tenant_name>admin</os_tenant_name>
        <os_username>admin</os_username>
        <member_role>_member_</member_role>
    </openstack_config>
    <vim_connectors>
        <vim_connector>
            <id>my-ucs-XY</id>
            <status>CONNECTION_FAILED</status>
            <status_message>Unable to establish VIM connection</status_message>
        </vim_connector>
        <vim_connector>
            <id>Openstack-Liberty</id>
            <status>NO_CREDENTIALS</status>
            <status_message>No VIM user credentials found</status_message>
        </vim_connector>
    </vim_connectors>
</system_config>
```

VIM Connector Operation Status

The VIM_CONNECTION_STATE notification notifies the status of each VIM connector and user added to ESC through REST and NETCONF. For more details about the VIM connectors, see Managing VIM Connectors, on page 1.

The notification shows:

- Event Type: VIM_CONNECTION_STATE
- Status: Success or Failure
- Status message
- vim_connector_id

Notifications are sent for monitoring the VIM connector, adding or deleting the VIM user, and updating the VIM connector. The success and failure notification examples are as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2017-06-27T14:50:40.823+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>FAILURE</status>
    <status_code>0</status_code>
    <status_message>VIM Connection State Down</status_message>
    <vim_connector_id>my-ucs-25-bad-user</vim_connector_id>
    <event>
      <type>VIM_CONNECTION_STATE</type>
    </event>
  </escEvent>
</notification>


<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
```

```
  <eventTime>2017-06-27T14:51:55.862+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>0</status_code>
    <status_message>VIM Connection State Up</status_message>
    <vim_connector_id>my-ucs-25-bad-user</vim_connector_id>
    <event>
      <type>VIM_CONNECTION_STATE</type>
    </event>
  </escEvent>
</notification>
```

VIM Connector Configurations for OpenStack

You can configure the VIM connector for OpenStack specific operations.

Note To configure a VIM connector, see Configuring the VIM Connector, on page 2.

Creating Non-admin Roles for ESC Users in OpenStack

By default, OpenStack assigns an admin role to the ESC user. Some policies may restrict using the default admin role for certain ESC operations. Starting from ESC Release 3.1, you can create non-admin roles with limited permissions for ESC users in OpenStack.

To create a non-admin role,

1. Create a non-admin role in OpenStack.
2. Assign the non-admin role to the ESC user.

   You must assign ESC user roles in OpenStack Horizon (Identity) or using the OpenStack command line interface. For more details see, OpenStack Documentation.

   The role name can be customized in OpenStack. By default, all non-admin roles in OpenStack have the same level of permissions.
3. Grant the required permissions to the non-admin role.

   You must modify the `policy.json` file to provide the necessary permissions.

Note You must grant permissions to the *create_port: fixed_ips* and *create_port: mac_address* parameters in the policy.json file for ESC user role to be operational.

The table below lists the ESC operations that can be performed by the non-admin role after receiving the necessary permissions.

Table 1: Non-admin role permissions for ESC operations

ESC VIM Operation	Description	Permission	Note
Create Project	To create an OpenStack project	`/etc/keystone/policy.json "identity:create_project" "identity:create_grant"`	For ESC managed OpenStack project, adding the user to the project with a role requires *identity:create_grant*.
Delete Project	To delete an OpenStack project	`/etc/keystone/policy.json "identity:delete_project"`	
Query Image	To get a list of all images	Not required	The owner (a user in the target project) can query. You can retrieve public or shared images.
Create Image	To create a public image	`/etc/glance/policy.json "publicize_image"`	By default an admin can create a public image. Publicizing an image is protected by the policy.
	To create a private image	Not required	You can use the following to create a private image `<image> <name>mk-test-image</name> ... <disk_bus>virtio</disk_bus> <visibility>private</visibility> </image>`
Delete Image	To delete an image	Not required	The owner can delete the image.
Query Flavor	To query a pre-existing flavor	Not required	The owner can query a flavor. You can query public flavors as well.
Create Flavor	To create a new flavor	`/etc/nova/policy.json "os_compute_api:os-flavor-manage"`	Managing a flavor is typically only available to administrators of a cloud.
Delete Flavor	To delete a flavor	`/etc/nova/policy.json "os_compute_api:os-flavor-manage"`	
Query Network	To get a list of networks	`/etc/neutron/policy.json "get_network"`	Owner can get the list of networks including shared networks.

ESC VIM Operation	Description	Permission	Note
Create Network	To create a normal network	Not required	
	To create network with special cases	/etc/neutron/policy.json "create_network:provider:physical_network" "create_network:provider:network_type" "create_network:provider:segmentation_id" "create_network:shared"	You need these rules when you are creating network with physical_network (e.g., SR-IOV), or network_type (e.g., SR-IOV), or segmentation_id (e.g., 3008), or set the network for sharing. <network> <name>provider-network</name> <!--<shared>false</shared> //default is true--> <admin_state>true</admin_state> <provider_physical_network>VAR_PHYSICAL_NET</provider_physical_network> <provider_network_type>vlan</provider_network_type> <provider_segmentation_id>2330</provider_segmentation_id> ... </network>
Delete Network	To delete a network	Not required	The owner can delete the network.
Query Subnet	To get a list of subnets	/etc/neutron/policy.json "get_subnet"	The network owner can get a list of the subnets. You can get a list of subnets from a shared network as well. <network> <name>esc-created-network</name> <!--network must be created by ESC--> <admin_state>false</admin_state> <subnet> <name>makulandyescextnet1-subnet1</name> <ipversion>ipv4</ipversion> <dhcp>true</dhcp> <address>10.6.0.0</address> <netmask>255.255.0.0</netmask> </subnet> </network>
Create Subnet	To create a subnet	Not required	The network owner can create a subnet.
Delete Subnet	To delete a subnet	Not required	The network owner can delete a subnet.

ESC VIM Operation	Description	Permission	Note
Query Port	Get a pre-existing port	Not required	The owner can get a list of ports.
Create Port	To create a network interface with DHCP	Not required	
	Create a network interface with a mac address	/etc/neutron/policy.json "create_port:mac_address"	<interfaces> <interface> <nicid>0</nicid> <mac_address>fa:16:3e:73:19:b5</mac_address> <network>esc-net</network> </interface> </interfaces>VM recovery also requires this privilege.
	To create a network interface with a fixed IP or shared ips	/etc/neutron/policy.json "create_port:fixed_ips"	<subnet> <name>IP-pool-subnet</name> <ipversion>ipv4</ipversion> <dhcp>false</dhcp> <address>10.65.5.0</address> <netmask>255.255.255.0</netmask> <gateway>10.65.5.1</gateway> </subnet><shared_ip> <nicid>0</nicid> <static>false</static> </shared_ip> VM recovery also requires this privilege.
Update Port	Update port device owner	Not required	The owner can update the port.
	Update port to allow address pairs	/etc/neutron/policy.json "update_port:allowed_address_pairs"	<interface> <nicid>0</nicid> <network>VAR_MANAGEMENT_NETWORK_ID</network> <allowed_address_pairs> <network> <name>VAR_MANAGEMENT_NETWORK_ID</name> </network> <address> <ip_address>123.45.0.0</ip_address> <netmask>255.255.0.0</netmask> </address> <address> <ip_address>123.45.6.0</ip_address> <ip_prefix>24</ip_prefix> </address> </allowed_address_pairs> </interface>
Delete Port	To delete a port	Not required	The owner can delete the port.

ESC VIM Operation	Description	Permission	Note
Query Volume	To get a list of volumes	Not required	The owner can get the list of volumes.
Create Volume	To create a volume	Not required	
Delete Volume	To delete a volume	Not required	The owner can delete the volume.
Query VM	To get all the VMs in a project	Not required	The owner can get the list of all the VMs in a project.
Create VM	To create a VM	Not required	
	To create a VM in a host targeted deployment	/etc/nova/policy.json "os_compute_api:servers:create:forced_host"	<placement> <type>zone_host</type> <enforcement>strict</enforcement> <host>anyHOST</host> </placement>
	To create VMs in a zone targeted deployment	Not required	
	To create VMs in the same Host Affinity/Anti-affinity	Not required	
	To create VMs in a servergroup Affinity/Anti-affinity	Not required	This support is for intragroup anti-affinity only.
Delete VM	To delete a VM	Not required	The owner can delete the VM.

For more details on managing resources on OpenStack, see Managing Resources on OpenStack .

Overwriting OpenStack Endpoints

By default, ESC uses endpoints catalog return option provided by OpenStack after a successful authentication. ESC uses these endpoints to communicate with different APIs in OpenStack. Sometimes the endpoints are not configured correctly, for example, the OpenStack instance is configured to use KeyStone V3 for authentication, but the endpoint returned from OpenStack is for KeyStone V2. You can overcome this by overwriting the OpenStack endpoints.

You can overwrite (configure) the OpenStack endpoints while configuring the VIM connector. This can be done at the time of installation using the bootvm.py parameters, and using the VIM connector APIs.

The following OpenStack endpoints can be configured using the VIM connector configuration:

- OS_IDENTITY_OVERWRITE_ENDPOINT
- OS_COMPUTE_OVERWRITE_ENDPOINT
- OS_NETWORK_OVERWRITE_ENDPOINT
- OS_IMAGE_OVERWRITE_ENDPOINT
- OS_VOLUME_OVERWRITE_ENDPOINT

To overwrite OpenStack endpoints at the time of installation, a user can create an esc configuration parameters file, and pass the file as an argument to bootvm.py while deploying an ESC VM.

Below is an example of the param.conf file:

```
openstack.os_identity_overwrite_endpoint=http://www.xxxxxxxxxx.com
```

For more information on configuring the VIM connector at the time of Installation, see Configuring the VIM Connector.

To overwrite (configure) the OpenStack endpoints for a non-default VIM connector using the VIM connector APIs (both REST and NETCONF), add the overwriting endpoints as the VIM connector properties either while creating a new VIM connector or updating an existing one.

Each VIM connector can have its own overwriting endpoints. There is no default overwriting endpoint.

In the example below, *os_identity_overwrite_endpoint* and *os_network_overwrite_endpoint* properties are added to overwrite the endpoints.

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <!--represents a vim-->
    <vim_connector>
      <id>default_openstack_vim</id>
      <type>OPENSTACK</type>
      <properties>
        <property>
          <name>os_auth_url</name>
          <value>http://10.85.103.153:35357/v3</value>
        </property>
        <property>
          <name>os_project_domain_name</name>
          <value>default</value>
        </property>
        <property>
          <name>os_project_name</name>
          <value>admin</value>
        </property>
        <property>
          <name>os_identity_overwrite_endpoint</name>
```

```
          <value>http://some_server:some_port/</value>
        </property>
        <property>
          <name>os_network_overwrite_endpoint</name>
          <value>http://some_other_server:some_other_port/</value>
        </property>
      </properties>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

VIM Connector Configurations for AWS

You can set the VIM credentials for an AWS deployment using the VIM connector and VIM User API.

Note AWS deployment does not support default VIM connector.

The VIM connector **aws_default_region** value provides authentication, and updates the VIM status. The default region cannot be changed after authentication.

Configuring the VIM Connector

To configure the VIM connector for AWS deployment, provide the AWS_ACCESS_ID, AWS_SECRET_KEY from your AWS credentials.

```
[admin@localhost ~]# esc_nc_cli edit-config
aws-vim-connector-example.xml
```

Note To edit the existing VIM connector configuration, use the same command after making the necessary changes.

The AWS VIM connector example is as follows:

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
   <vim_connectors>
      <vim_connector>
         <id>AWS_EAST_2</id>
         <type>AWS_EC2</type>
         <properties>
            <property>
               <name>aws_default_region</name>
               <value>us-east-2</value>
            </property>
         </properties>
         <users>
            <user>
               <id>AWS_ACCESS_ID</id>
               <credentials>
                  <properties>
                     <property>
                        <name>aws_secret_key</name>
                        <encrypted_value>AWS_SERCRET_KEY</encrypted_value>
                     </property>
                  </properties>
               </credentials>
            </user>
```

```
        </users>
      </vim_connector>
   </vim_connectors>
</esc_system_config>
```

Deleting VIM Connector

To delete the existing VIM connector, you must first delete the deployment, the VIM user, and then the VIM connector.

```
[admin@localhost ~]# esc_nc_cli delete-vimuser
AWS_EAST_2 AWS_ACCESS_ID

[admin@localhost ~]# esc_nc_cli delete-vimconnector
AWS_EAST_2
```

Note You can configure multiple VIM connectors, but for the same VIM type.

The VIM connectors for AWS deployment must be configured using the VIM connector API.

ESC supports one VIM user per VIM connector.

The VIM connector and its properties cannot be updated after deployment.

For information on deploying VNFs on AWS, see Deploying VNFs on a Single or Multiple AWS Regions.

VIM Connector Configuration for VMware vCloud Director (vCD)

You must configure a VIM connector to connect to the vCD organization. The organization and the organization user must be preconfigured in the VMware vCD. For the deployment datamodel, see the Deploying Virtual Network Functions on VMware vCloud Director (vCD).

The VIM connector details are as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
   <vim_connectors>
      <vim_connector>
         <id>vcd_vim</id>
         <type>VMWARE_VCD</type>
         <properties>
            <property>
               <name>authUrl</name>
                <!-- vCD is the vCD server IP or host name -->
               <value>https://vCD</value>
            </property>
         </properties>
         <users>
            <user>
              <!-- the user id here represents {org username}@{org name} -->
               <id>user@organization</id>
               <credentials>
                  <properties>
                     <property>
                        <name>password</name>
                          <!—the organization user’s password-->
```

```
                        <value>put user's password here</value>
                      </property>
                    </properties>
                  </credentials>
                </user>
              </users>
            </vim_connector>
          </vim_connectors>
        </esc_system_config>
```

Authenticating External Configuration Files

Prior to Cisco ESC Release 4.0, ESC supports several external configuration files and scripts as part of day 0 configuration, monitoring, deployment and LCS actions. ESC supports getting these files from a remote server with or without authentication as part of the deployment.

Starting from ESC Release 4.0, the file locator attribute is defined at the deployment level, that is, directly under the deployment container. This allows multiple VM groups and their day 0 configuration and LCS actions to reference the same file locator wherever needed within the deployment.

Sample deployment data model is as follows:

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>sample-tenant</name>
      <deployments>
        <deployment>
          <name>sample-deployment</name>
          <file_locators>
            <file_locator>
              <name>post_deploy_alive_script</name>
              <remote_file>
                <file_server_id>http-my-ucs-42</file_server_id>
                <remote_path>/share/qatest/vnfupgrade/lcspostdeployalive.sh</remote_path>
                <local_target>vnfupgrade/lcspostdepalive.sh</local_target>
                <persistence>FETCH_ALWAYS</persistence>
                <properties/>
              </remote_file>
            </file_locator>
            <file_locator>
              <name>asa-day0-config</name>
              <remote_file>
                <file_server_id>http-my-ucs-42</file_server_id>
                <remote_path>/share/qatest/day0/asa_config.sh</remote_path>
                <local_target>day0.1/asa_config.sh</local_target>
                <persistence>FETCH_ALWAYS</persistence>
              </remote_file>
            </file_locator>
            <file_locator>
              <name>scriptlocator</name>
              <remote_file>
                <file_server_id>dev_test_server</file_server_id>
                <remote_path>/share/users/gomoore/actionScript.sh</remote_path>
                <local_target>action/actionScript.sh</local_target>
                <persistence>FETCH_MISSING</persistence>
                <properties/>
              </remote_file>
            </file_locator>
          </file_locators>
          <policies>
```

```
    <policy>
      <name>VNFUPGRADE_POST_DEPLOY_ALIVE</name>
      <conditions>
        <condition>
          <name>LCS::POST_DEPLOY_ALIVE</name>
        </condition>
      </conditions>
      <actions>
        <action>
          <name>post_deploy_alive_action</name>
          <type>SCRIPT</type>
          <properties>
            <property>
              <name>file_locator_name</name>
              <value>post_deploy_alive_script</value>
            </property>
          </properties>
        </action>
      </actions>
    </policy>
  </policies>
<vm_group>
    <name>ASA-group</name>
    <image>ASAImage</image>
    <flavor>m1.large</flavor>
    <recovery_policy>
      <max_retries>1</max_retries>
    </recovery_policy>
    <scaling>
      <min_active>1</min_active>
      <max_active>1</max_active>
      <elastic>true</elastic>
    </scaling>
    <placement>
      <type>affinity</type>
      <enforcement>strict</enforcement>
    </placement>
    <bootup_time>120</bootup_time>
    <recovery_wait_time>60</recovery_wait_time>
    <interfaces>
      <interface>
        <nicid>0</nicid>
        <network>esc-net</network>
      </interface>
    </interfaces>
    <kpi_data>
      <kpi>
        <event_name>VM_ALIVE</event_name>
        <metric_value>1</metric_value>
        <metric_cond>GT</metric_cond>
        <metric_type>UINT32</metric_type>
        <metric_occurrences_true>1</metric_occurrences_true>
        <metric_occurrences_false>5</metric_occurrences_false>
        <metric_collector>
          <nicid>0</nicid>
          <type>ICMPPing</type>
          <poll_frequency>5</poll_frequency>
          <polling_unit>seconds</polling_unit>
          <continuous_alarm>false</continuous_alarm>
        </metric_collector>
      </kpi>
    </kpi_data>
    <rules>
<admin_rules>
```

```
                <rule>
                  <event_name>VM_ALIVE</event_name>
                  <action>ALWAYS log</action>
                  <action>TRUE servicebooted.sh</action>
                  <action>FALSE recover autohealing</action>
                </rule>
              </admin_rules>
            </rules>
            <config_data>
              <configuration>
                <dst>ASA.static.txt</dst>
                <file_locator_name>asa-day0-config</file_locator_name>
              </configuration>
            </config_data>
            <policies>
              <policy>
                <name>SVU1</name>
                <conditions>
                <condition><name>LCS::DEPLOY_UPDATE::PRE_VM_VOLUME_DETACH</name></condition>

                </conditions>
                <actions>
                  <action>
                    <name>LOG</name><type>pre_defined</type>
                  </action>
                  <action>
                    <name>pre_vol_detach</name>
                    <type>SCRIPT</type>
                    <properties>
                      <property>
                        <name>file_locator_name</name>
                        <value>scriptlocator</value>
                      </property>
                      <property>
                        <name>exit_val</name>
                        <value>0</value>
                      </property>
                    </properties>
                  </action>
                </actions>
              </policy>
            </policies>
          </vm_group>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```

You must configure a remote server (file server) separately using the APIs before performing any deployment. Both REST and NETCONF APIs are supported

- A remote server with URL, authentication details including username, and password. You can either use REST or NETCONF to configure.

Note The username and password are optional. The password is encrypted within ESC.

You must configure the remote file server before deployment. You can update the credentials anytime during the deployment.

- File locator is added to the deployment data model. It contains a reference to the file server, and the relative path to the file to be downloaded.

To get files remotely with authentication, you must

1. Add a remote server.
2. Refer the remote server in the file locator. The file locator is part of config data in day 0 and LCS action blocks.
3. The day 0 and lifecycle stage (LCS) scripts will then be retrieved based on the file locator as part of the deployment.

The file server parameters include:

- id—used as the key and identifier for a file server.
- base_url—the address of the server. (e.g. http://www.cisco.com or https://192.168.10.23)
- file_server_user—the username to use when authenticating to the server.
- file_server_password—string containing the password for authenticating to the server. Initially the user provides a cleartext string, which is encrypted internally.
- properties—name-value pair for extensibility in the future.

The file locator parameters include:

- name—used as the key and identifier for a file locator.
- local_file or remote_file—choice of file location. Local file is used to specify a file existing on the ESC VM file system already. The remote_file is used to specify a file to fetch from a remote server.
  - file_server_id—id of the File Server object to fetch the file from.
  - remote_path—path of the file from the base_url defined in the file server object.
  - local_target—optional local relative directory to save the file.
  - properties—name-value pairs of of information that may be required.
  - persistence—options for file storage. Values include CACHE, FETCH_ALWAYS and FETCH_MISSING (default).
- checksum—optional BSD style checksum value to use to validate the transferred file's validity.

The file server values such as server connectivity, file existence, checksum and so on will be verified for validity.

The encrypted_data values in the file_server_password and properties encrypted_data fields are encrypted using AES/128bits in CFB mode for transmission.The data remains encrypted until it is required for accessing the server. For more information on encrypted values, see Encrypting Configuration Data.

Example of file servers,

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <file_servers>
    <file_server>
      <id>server-1</id> <!-- unique name for server -->
      <base_url>https://www.some.server.com</base_url>
```

```
      <file_server_user>user1</file_server_user>
     <file_server_password>sample_password</file_server_password> <!-- encrypted value -->

      <!-- properties list containing additional items in the future -->
      <properties>
        <property>
          <name>server_timeout</name>
          <value>60</value> <!-- timeout value in seconds, can be over-ridden in a
file_locator -->
        </property>
      </properties>
    </file_server>
    <file_server>
      <id>server-2</id>
      <base_url>https://www.some.other.server.com</base_url>
          <properties>
        <property>
          <name>option1</name>
          <encrypted_value>$8$EADFAQE</encrypted_value>
        </property>
    </file_server>
  </file_servers>
</esc_datamodel>
```

Example for day 0 configuration

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants><tenant>
      <name>sample-tenant</name>
      <deployments><deployment>
          <name>sample-deployment</name>
          <vm_group>
            <name>sample-vm-group</name>
            <config_data>
              <!-- exisiting configuration example - remains valid -->
              <configuration>
                <file>file:///cisco/config.sh</file>
                <dst>config.sh</dst>
              </configuration>
              <!-- new configuration including use of file locators -->
              <configuration>
                <dst>something</dst>
                <file_locators>
                  <file_locator>
                    <name>configlocator-1</name> <!-- unique name -->
                    <remote_file>
                      <file_server_id>server-1</file_server_id>
                      <remote_path>/share/users/configureScript.sh</remote_path>
                      <!-- optional user specified local silo directory -->
                      <local_target>day0/configureScript.sh</local_target>
                      <!-- persistence is an optional parameter -->
                      <persistence>FETCH_ALWAYS</persistence>
                      <!-- properties in the file_locator are only used for
                           fetching the file not for running scripts -->
                      <properties>
                        <property>
                          <!-- the property name "configuration_file" with value "true"
indictates this is the
                               script to be used just as using the <file> member case of
the configuration -->
                          <name>configuration_file</name>
                          <value>true</value>
                        </property>
                        <property>
                          <name>server_timeout</name>
```

```
                          <value>120</value> <!-- timeout value in seconds, overrides the
file_server property -->
                        </property>
                        </properties>
                      </remote_file>
                      <!-- checksum is an optional parameter.
                           The following algorithms are supported: SHA-1, SHA-224, SHA-256,
SHA-384, SHA-512 -->
                      <checksum>SHA256 (configureScript.sh) =
dd526bb2c0711238ec2649c4b91598fb9a6cf1d2cb8559c337c5f3dd5ea1769e</checksum>
                    </file_locator>
                    <file_locator>
                      <name>configlocator-2</name>
                      <remote_file>
                        <file_server_id>server-2</file_server_id>
                        <remote_path>/secure/requiredData.txt</remote_path>
                        <local_target>day0/requiredData.txt</local_target>
                        <persistence>FETCH_ALWAYS</persistence>
                        <properties/>
                      </remote_file>
                    </file_locator>
                  </file_locators>
                </configuration>
              </config_data>
            </vm_group>
          </deployment></deployments>
    </tenant></tenants>
</esc_datamodel>
```

For more details on day 0 configuration and LCS actions, see day 0 configuration, and Redeployment Policy sections.

Encrypting Configuration Data

You can encrypt configuration data with secret keys and private information. In ESC, the day 0 configuration, day 0 configuration variables, VIM connector and VIM user, and LCS actions contain secret keys.

ConfD provides encrypted string types. Using the built-in string types, the encrypted values are stored in ConfD. The keys used to encrypt the values are stored in confd.conf.

Encrypting data is optional. You can use the encrypt_data value to store data if necessary.

In the example below, the day 0 configuration data has encrypted values. The encrypted_data uses the built in string type *tailf:aes-cfb-128-encrypted-string*.

```
choice input_method {
  case file {
    leaf file {
      type ietf-inet-types:uri;
    }
  }
  case data {
    leaf data {
      type types:escbigdata;
    }
  }
  case encrypted_data {
    leaf encrypted_data {
      type tailf:aes-cfb-128-encrypted-string;
    }
  }
}
```

Generating Advanced Encryption Standard (AES) Key

The AES key is 16 bytes in length, and contains a 32 character hexadecimal string.

You must configure the AES key in confd.conf for the encryption to work.

```
/opt/cisco/esc/esc-confd/esc_production_confd.conf
```

```
<encryptedStrings>
    <AESCFB128>
        <key>0123456789abcdef0123456789abcdef</key>
        <initVector>0123456789abcdef0123456789abcdef</initVector>
    </AESCFB128>
</encryptedStrings>
```

A default AES key is available in confD:

```
0123456789abcdef0123456789abcdef
```

The confD key is hard-coded. The escadm.py generates a random AES key and replaces the default confD AES key before confD starts.