



## OpenStack のリソースの管理

---

- [OpenStack のリソースの管理 \(1 ページ\)](#)
- [テナントの管理 \(1 ページ\)](#)
- [ネットワークの管理 \(10 ページ\)](#)
- [サブネットの管理 \(13 ページ\)](#)
- [フレイバの管理 \(14 ページ\)](#)
- [イメージの管理 \(15 ページ\)](#)
- [ボリュームの管理 \(16 ページ\)](#)

## OpenStack のリソースの管理

### テナントの管理

テナントは、一連の管理者に関連付けられているテナント組織またはグループを識別します。テナント定義を作成すると、リージョンとローカルの両方のクラスタに保存されるデータが、テナント別にセグメント化されます。テナントが別のテナントのデータにアクセスすることはできません。NETCONF/REST インターフェイスまたは ESC ポータルを使用し、ESC を介してテナント定義を作成できます。



---

(注) テナントは VMware vCenter ではサポートされていません。

---

ESC では、次の 3 種類のテナントを作成できます。

1. VIM 上のテナント (ESC がテナントを作成) : ESC ではデフォルトの VIM での展開用にテナントを作成して使用できます。ESC ではこのテナントを削除できます。
2. VIM 上の既存の (アウトオブバンド) テナント : ESC ではこのテナントを作成せず、デフォルトの VIM での展開にのみテナントを使用します。たとえば、**admin** テナントは、ESC 自体が展開されている既存のテナントです。ESC では、名前または UUID で識別される既存のテナントへのフレイバー、イメージ、ボリュームなどのリソースの展開がサポー

トされます。ESCでは、デフォルトのVIMに対してのみ既存のテナントが管理されます。ESCでは既存のテナントを削除できません。

3. ESC内のテナント：ESCでは、ESC内にテナントが作成されます。このテナントは、いずれのVIMからも独立しています。このテナントは、複数のVIMにVMを展開するためのルートテナントとして機能します。

テナント名は一意である必要があります。



- (注) ESCでは、テナント、ネットワーク、サブネットワーク、イメージ、フレーバーなどのリソースをデフォルトのVIMでのみ作成して管理できます。(デフォルトのVIM以外の)デフォルトではないVIMでは、展開のみがサポートされます。

データモデルのテナントは、次の属性で管理します。

- managed\_resource 属性
- vim\_mapping 属性

次の表に、データモデルのテナントと属性のマッピングの詳細が示されています。

テナントタイプ	managed_resource	vim_mapping	説明
VIMのテナント (ESCによって作成)	true	true	<p>managed_resource 属性が true に設定されている場合、ESCではVIM上にテナントが作成されます。デフォルトでは、managed_resource は true です。vim_mapping 属性は true です。</p> <pre> &lt;tenants&gt;   &lt;tenant&gt;     &lt;name&gt;new-tenant&lt;/name&gt;  &lt;managed_resource&gt;true&lt;/managed_resource&gt;   &lt;/tenant&gt; &lt;/tenants&gt;</pre>

テナントタイプ	managed_resource	vim_mapping	説明
VIM 上の既存のテナント	false	true	<p>既存のテナントの場合、managed_resource 属性は false に設定されます。vim_mapping 属性は true です。</p> <pre> &lt;tenants&gt;   &lt;tenant&gt;     &lt;name&gt;pre-existing&lt;/name&gt;  &lt;managed_resource&gt;false&lt;/managed_resource&gt;   &lt;/tenant&gt; &lt;/tenants&gt; </pre> <p>テナント UUID を使用したサンプルデータモデル</p> <pre> &lt;tenants&gt;   &lt;tenant&gt;  &lt;name&gt;76eedcae-6067-44a7-b733-fc99a2e50bdf&lt;/name&gt;  &lt;managed_resource&gt;false&lt;/managed_resource&gt;   &lt;/tenant&gt; &lt;/tenants&gt; </pre>
ESC 内のテナント	-	false	<p>ESC 内にテナントを作成するには、vim_mapping 属性を false に設定します。</p> <pre> &lt;tenants&gt;   &lt;tenant&gt;     &lt;name&gt;esc-tenant-A&lt;/name&gt;     &lt;vim_mapping&gt;false&lt;/vim_mapping&gt;   &lt;/tenant&gt; &lt;/tenants&gt; </pre>
-	false	false	テナントは作成されません。要求はESCによって拒否されます。

同じタイプの複数の VIM (OpenStack VIM) に VM を展開するには、vim\_mapping 属性を false に設定してテナントを作成する必要があります。このテナントは、個別に作成することも、展開の一部として作成することもできます。これにより、ESC 内にテナントが作成され、マルチ VIM 展開のルートテナントとして機能します。マルチ VIM 展開の場合は、各 VM グループ内で VIM ロケータ属性を指定する必要があります。詳細については、[単一の VMware vCenter VIM での VNF の展開](#)を参照してください。

### テナントクォータ

ESC で作成されたテナントのクォータと呼ばれる動作制限を設定できます。クォータは、展開データモデルを使用して展開中に設定できます。



(注) テナントクォータは、既存のテナントおよび ESC 内のテナントではサポートされません。

テナントは、コンピューティング (Nova) およびネットワーク (Neutron) の次のクォータ設定をサポートしています。

コンピューティングの設定 :

- metadata\_items
- floating\_ips
- コア
- jected\_file\_path\_bytes
- jected\_files
- jected\_file\_content\_bytes
- インスタンス
- key\_pairs
- ram
- security\_groups
- security\_group\_rules

コンピューティングの設定 :

- floatingip
- security\_group\_rule
- security\_group
- network
- サブネット
- port
- ルータ

次の展開データモデルは、テナントのクォータ設定を示しています。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>tenant-quota-example</name>
      <extensions>
        <extension>
          <name>quota</name>
          <properties>
            <property>
              <name>cores</name>
```

```
    <value>26</value>
  </property>
</property>
<property>
  <name>metadata_items</name>
  <value>260</value>
</property>
<property>
  <name>floating_ips</name>
  <value>26</value>
</property>
<property>
  <name>injected_file_content_bytes</name>
  <value>26000</value>
</property>
<property>
  <name>injected_file_path_bytes</name>
  <value>246</value>
</property>
<property>
  <name>injected_files</name>
  <value>26</value>
</property>
<property>
  <name>instances</name>
  <value>26</value>
</property>
<property>
  <name>key_pairs</name>
  <value>26</value>
</property>
<property>
  <name>ram</name>
  <value>26</value>
</property>
<property>
  <name>security_groups</name>
  <value>26</value>
</property>
<property>
  <name>security_group_rules</name>
  <value>26</value>
</property>
<property>
  <name>floatingip</name>
  <value>26</value>
</property>
<property>
  <name>security_group_rule</name>
  <value>26</value>
</property>
<property>
  <name>security_group</name>
  <value>26</value>
</property>
<property>
  <name>network</name>
  <value>26</value>
</property>
<property>
  <name>subnet</name>
  <value>26</value>
</property>
<property>
  <name>port</name>
```

```

      <value>26</value>
    </property>
  </property>
  <name>router</name>
  <value>26</value>
</property>
</properties>
</extension>
</extensions>
</tenant>
</tenants>
</esc_datamodel>

```



- (注) 展開データモデルのプロパティ名は、前述のコンピューティングおよびネットワークの設定名と一致している必要があります。テナント作成要求は拒否されます。

### ノースバウンド API を使用したテナントの追加

次に、NETCONF を使用してテナント定義を作成する例を示します。

```

<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>

```



- (注) NETCONF API を使用したテナント定義の作成と削除の詳細については、[Cisco Elastic Services Controller API ガイド \[英語\]](#) を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、[REST ノースバウンド API](#) を参照してください。ESC ポータルを使用したネットワークの追加と削除の詳細については、[ESC ポータルを使用したリソースの管理](#) を参照してください。

### テナントのクォータの更新

ESC で作成されたテナントのクォータを更新できます。クォータの更新は、`managed_resource` 属性と `vim_mapping` 属性が `true` に設定されているテナントでのみ許可されます。ただし、`name`、`vim_mapping`、`managed_resource`、`description` などの設定は更新できません。

次の展開データモデルは、テナントのクォータの1つまたは複数のプロパティを更新するプロセスを示しています。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>ten-test-1</name>
      <managed_resource>true</managed_resource>
      <vim_mapping>true</vim_mapping>
      <extensions>
        <extension>
          <name>quota</name>
          <properties>
            <property>
              <name>cores</name>
              <value>15</value>
            </property>
            <property>
              <name>ram</name>
              <value>10000</value>
            </property>
          </properties>
        </extension>
      </extensions>
    </tenant>
  </tenants>
</esc_datamodel>
```

次のデータモデルは、テナントのクォータのコアプロパティを変更する方法を示しています。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>ten-test-1</name>
      <managed_resource>true</managed_resource>
      <vim_mapping>true</vim_mapping>
      <extensions>
        <extension>
          <name>quota</name>
          <properties>
            <property>
              <name>cores</name>
              <value>20</value>
            </property>
            <property>
              <name>ram</name>
              <value>10000</value>
            </property>
          </properties>
        </extension>
      </extensions>
    </tenant>
  </tenants>
</esc_datamodel>
```

次のデータモデルは、存在しないプロパティをテナントのクォータに追加する方法を示しています。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>ten-test-1</name>
      <managed_resource>true</managed_resource>
      <vim_mapping>true</vim_mapping>
      <extensions>
        <extension>
          <name>quota</name>
          <properties>
```

```

      <property>
        <name>cores</name>
        <value>15</value>
      </property>
      <property>
        <name>ram</name>
        <value>10000</value>
      </property>
      <property>
        <name>network</name>
        <value>10</value>
      </property>
    </properties>
  </extension>
</extensions>
</tenant>
</tenants>
</esc_datamodel>

```

次の例は、データモデルからプロパティを削除する方法を示しています。

```

<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>ten-test-1</name>
      <managed_resource>true</managed_resource>
      <vim_mapping>true</vim_mapping>
      <extensions>
        <extension>
          <name>quota</name>
          <properties>
            <property nc:operation="delete">
              <name>cores</name>
              <value>15</value>
            </property>
            <property>
              <name>ram</name>
              <value>10000</value>
            </property>
          </properties>
        </extension>
      </extensions>
    </tenant>
  </tenants>
</esc_datamodel>

```



- 
- (注) プロパティはデータモデルからのみ削除されます。クォータ値は、OpenStack 内のそのテナントに対しては同じままです。
- 

### REST API を使用したテナントクォータの更新

REST API を使用して、新しいテナントを作成したり、ESC の既存テナントのクォータを変更したりできます。

方式タイプ :

PUT

URL : `/ESCManager/v0/tenants/[tenant_internal_id]`



HTTP 要求ヘッダー :

**internal\_tenant\_id** : 更新するテナント ID

**callback** : 残りのコールバック通知を受信するアドレスとポート

**Content-Type** : application/xml

クォータを指定してテナントを作成する際の REST API の例。

```
<tenant xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <name>tenant_internal_id</name>
  <managed_resource>true</managed_resource>
  <extensions>
    <extension>
      <name>quota</name>
      <properties>
        <property>
          <name>port</name>
          <value>17</value>
        </property>
        <property>
          <name>ram</name>
          <value>17021</value>
        </property>
        <property>
          <name>cores</name>
          <value>22</value>
        </property>
      </properties>
    </extension>
  </extensions>
</tenant>
```

クォータを変更または追加してテナントを作成する際の REST API の例。

```
<tenant xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <name>tenant_internal_id</name>
  <managed_resource>true</managed_resource>
  <extensions>
    <extension>
      <name>quota</name>
      <properties>
        <property>
          <name>port</name>
          <value>20</value>
        </property>
        <property>
          <name>ram</name>
          <value>15000</value>
        </property>
        <property>
          <name>network</name>
          <value>5</value>
        </property>
      </properties>
    </extension>
  </extensions>
</tenant>
```

# ネットワークの管理

ESCでは、ネットワークとサブネットワークを作成して設定し、OpenStack または VMware vCenter のいずれかのサービスにそれらのネットワークのポートに仮想マシンを接続するように指示することで、豊富なネットワークトポロジを設定できます。

## OpenStack ネットワーク

特に、OpenStack ネットワークでは、各テナントが複数のプライベートネットワークを持つことができ、他のテナントで使用されている IP アドレスと重複する場合でも、各テナントは独自の IP アドレス方式を選択できます。これにより、多層 Web アプリケーションを構築する、IP アドレスを変更せずにアプリケーションをクラウドに移行するなど、非常に高度なクラウドネットワークの使用例を実現できます。

ESC は次のネットワーク機能をサポートしています。

- テナントネットワーク：テナントネットワークは、単一のネットワークとそのすべてのインスタンスに対して作成されます。また、他のテナントから分離されます。
- プロバイダーネットワーク：プロバイダーネットワークは管理者によって作成されます。属性は、基盤となる物理ネットワークまたはセグメントにマッピングされます。

次の属性で、プロバイダーネットワークを定義します。

- network\_type
  - physical\_network
  - segmentation\_id
- 外部ネットワーク：通常、外部ネットワークはインスタンスにインターネットアクセスを提供します。デフォルトでは、外部ネットワークは、ネットワークアドレス変換 (NAT) を使用してインスタンスからのインターネットアクセスのみ許可します。フローティング IP アドレスと適切なセキュリティグループルールを使用して、個々のインスタンスへのインターネットアクセスを有効にできます。admin テナントは、複数のテナントに外部ネットワークアクセスを提供するため、このネットワークを所有します。

ESC はエフェメラルネットワークもサポートします。エフェメラルネットワークは、統合型の展開中に意図的に作成され、その展開の存続期間中のみ存在します。詳細については、「[統合型の展開要求](#)」を参照してください。

## ノースバウンド API を使用したネットワークの追加

次に、NETCONF を使用してテナントネットワーク定義を作成する例を示します。

```
<?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>
    </tenant>
  </tenants>
</esc_datamodel>

```

次に、NETCONF を使用してテナントネットワーク定義のサブネットを作成する例を示します。

```

<?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>
    </tenant>
  </tenants>
  <networks>
    <network>
      <name>proto-tenant-network27</name>
      <subnet>
        <name>proto-tenant-subnet4</name>
        <ipversion>ipv4</ipversion>
        <dhcp>true</dhcp>
        <address>172.16.0.0</address>
        <netmask>255.255.255.0</netmask>
        <gateway>172.16.0.1</gateway>
      </subnet>
    </network>
  </networks>
</esc_datamodel>

```

次に、NETCONF を使用して単純なプロバイダーネットワーク定義を作成する例を示します。

```

<?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>test-net-12</name>
      <shared>true</shared>
      <admin_state>true</admin_state>
      <provider_physical_network>vm_physnet</provider_physical_network>
      <provider_network_type>vlan</provider_network_type>
      <provider_segmentation_id>200</provider_segmentation_id>
    </network>
  </networks>
</esc_datamodel>

```

次に、NETCONF を使用してプロバイダーネットワーク定義のサブネットを作成する例を示します。

```
<?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>test-net-12</name>
      <subnet>
        <name>test-net-12-subnet</name>
        <ipversion>ipv4</ipversion>
        <dhcp>false</dhcp>
        <address>172.16.0.0</address>
        <gateway>172.16.0.1</gateway>
        <netmask>255.255.255.0</netmask>
      </subnet>
    </network>
  </networks>
</esc_datamodel>
```

次に、Cisco VIM でプロバイダー ネットワーク タイプの vxlan-evpn を作成する例を示します。

```
<?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>ProviderNetworkAttributes-vxlan-evpn</name>
      <shared>true</shared>
      <provider_network_type>vxlan-evpn</provider_network_type>
      <provider_segmentation_id>3010</provider_segmentation_id>
    </network>
  </networks>
</esc_datamodel>
```

次に、NETCONF を使用して外部ネットワーク定義を作成する例を示します。

```
<network>
  <name>xyz-yesc-net-1</name>
  <shared>false</shared>
  <admin_state>true</admin_state>
  <router_external></router_external>
  <subnet>
    <name>xyz-yesc-subnet-1</name>
    <ipversion>ipv4</ipversion>
    <dhcp>true</dhcp>
    <address>172.16.0.0</address>
    <netmask>255.255.255.0</netmask>
    <gateway>172.16.0.1</gateway>
  </subnet>
</network>
```



- (注) NETCONF API を使用したネットワークの作成と削除の詳細については、[Cisco Elastic Services Controller API ガイド \[英語\]](#) を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、[REST ノースバウンド API](#) を参照してください。ESC ポータルを使用したネットワークの追加と削除の詳細については、[ESC ポータルを使用したリソースの管理](#) を参照してください。

## サブネットの管理

ESCでは、サブネットは仮想ネットワークに割り当てられます。IPアドレス、ネットワークのIPバージョンなどを指定します。NETCONF/REST インターフェイスを使用してサブネット定義を作成できます。



(注) サブネットは OpenStack でのみサポートされます。

### ノースバウンド API を使用したサブネット定義の追加

次に、NETCONF を使用してサブネット定義を作成する例を示します。

```
<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/>
    </target>
    <config
      <esc_datamodel xmlns="http://www.cisco.com/esc/esc"
        xmlns:ns0="http://www.cisco.com/esc/esc"
        xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
        xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
        xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0">
      <networks>
        <network>
          <name>mgmt-net</name>
          <subnet>
            <name>mgmt-net-subnet</name>
            <ipversion>ipv4</ipversion>
            <dhcp>false</dhcp>
            <address>172.16.0.0</address>
            <gateway>172.16.0.1</gateway>
            <netmask>255.255.255.0</netmask>
          </subnet>
        </network>
      </networks>
    </esc_datamodel>
  </edit-config> </edit-config>
</rpc>
```

`no_gateway` 属性を使用すると、ESC はゲートウェイを無効にした状態でサブネットを作成できます。

次に、`no_gateway` 属性を `true` に設定して、ゲートウェイなしでサブネットを作成する例を示します。

```
<networks>
  <network>
    <name>mgmt-net</name>
    <subnet>
      <name>mgmt-net-subnet</name>
      <ipversion>ipv4</ipversion>
      <dhcp>false</dhcp>
      <address>172.16.0.0</address>
      <no_gateway>true</no_gateway><!-- DISABLE GATEWAY -->
```

```

        <gateway>172.16.0.1</gateway>
        <netmask>255.255.255.0</netmask>
    </subnet>
</network>
</networks>

```



- (注) NETCONF API を使用したサブネットの作成の詳細については、[Cisco Elastic Services Controller API ガイド \[英語\]](#) を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、[REST ノースバウンド API](#) を参照してください。ESC ポータルを使用したネットワークの追加と削除の詳細については、[ESC ポータルを使用したリソースの管理](#) を参照してください。

## フレーバの管理

フレーバは、RAM とディスクのサイズ、およびコアの数を定義します。

OpenStack に VNF を展開する場合は、OpenStack ですでに使用可能なアウトオブバンドフレーバを使用するか、ESC でフレーバを作成するかを選択できます。これらのフレーバは、NETCONF または REST インターフェイス、または ESC ポータルを使用して作成でき、複数の展開に使用できます。展開属性の詳細については、「[Cisco Elastic Services Controller Deployment Attributes](#)」を参照してください。



- (注) ESC リリース 2.0 以降では、VMware vCenter でのフレーバ定義の作成または削除はサポートされていません。

### ノースバウンド API を使用したフレーバの追加

NETCONF のフレーバ作成要求 :

```

<?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">
  <flavors>
    <flavor>
      <name>test-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>

```

フレーバの作成に成功した場合の NETCONF 通知 :

```

<?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>test-flavor-indep</flavor>
  <vm_source>
</vm_source>
  <vm_target>
</vm_target>
  <event>
    <type>CREATE_FLAVOR</type>
  </event>
</escEvent>
</notification>

```



- (注) NETCONF API を使用したフレーバの作成と削除の詳細については、[Cisco Elastic Services Controller API ガイド \[英語\]](#) を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、[REST ノースバウンド API](#) を参照してください。ESC ポータルを使用したフレーバの追加と削除の詳細については、[ESC ポータルを使用したリソースの管理](#) を参照してください。

## イメージの管理

ESC では、イメージは VM インスタンスの起動に使用できるブート可能なファイルシステムです。

OpenStack に VNF を展開する場合は、OpenStack ですでに使用可能なアウトオブバンドイメージを使用するか、ESC でイメージを作成するかを選択できます。これらのイメージは、NETCONF または REST インターフェイスを使用して作成でき、複数の展開に使用できます。

イメージは、OpenStack でパブリックまたはプライベートに設定できます。デフォルトでは、イメージはパブリックです。visibility 属性は、イメージをパブリックまたはプライベートとしてマークするために使用されます。パブリックイメージは管理者だけが作成できますが、プライベートイメージには管理者のログイン情報は必要ありません。

サンプル XML は次のとおりです。

```

<images>
  <image>
    <name>mk-test-image</name>
    <src>file:///opt/cisco/esc/esc-confd/esc-cli/dummy.xml</src>
    <disk_format>qcow2</disk_format>
    <container_format>bare</container_format>
    <serial_console>>true</serial_console>
    <disk_bus>virtio</disk_bus>
    <visibility>private</visibility>
  </image>
</images>

```

アウトオブバンドイメージおよび ESC によって作成されたイメージはどちらも、パブリックまたはプライベートにできます。

## ノースバウンド API を使用したイメージの追加

イメージを作成するための NETCONF 要求 :

```
<?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>example-cirrosimage-indep</name>

      <src>http://172.16.0.0:/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>
```

イメージが正常に作成された時の NETCONF 通知 :

```
<?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>example-cirrosimage-indep</image>
    <vm_source>
    </vm_source>
    <vm_target>
    </vm_target>
    <event>
      <type>CREATE_IMAGE</type>
    </event>
  </escEvent>
</notification>
```



- (注) NETCONF API を使用したイメージの追加の詳細については、[Cisco Elastic Services Controller API ガイド \[英語\]](#) を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、[REST ノースバウンド API](#) を参照してください。ESC ポータルを使用したイメージの追加と削除の詳細については、[ESC ポータルを使用したリソースの管理](#) を参照してください。

## ボリュームの管理

ボリュームは、Nova のブロックデバイスに似たストレージデバイスです。ESC は、ESC によって作成されたボリュームとアウトオブバンドボリュームの両方をサポートします。さらに、ESC は、ESC によって作成されたブート可能ボリュームと、アウトオブバンドのブート可能ボリュームもサポートします。





(注) nova boot コマンドを使用して VM に接続できるボリュームの最大数は 2 つだけです。

### ESC によって作成されたボリューム

VM グループの一部としてボリュームを作成するには、<size> および <sizeunits> パラメータを、展開要求のボリュームセクションで指定する必要があります。ボリュームタイプは、Cinder のデフォルトのボリュームタイプです。

次の例は、展開要求で ESC ボリュームを作成する方法を示しています。

```
<volumes>
  <volume>
    <name>example</name>
    <valid>1</valid>
    <bus>ide</bus>
    <size>1</size>
    <sizeunit>GiB</sizeunit>
  </volume>
</volumes>
```

### ESC によって作成されたブート可能ボリューム

ブート可能ボリュームは、ルートディスクとして使用されるボリュームです。ESCは、展開要求のイメージ参照名またはUUIDを使用して、ブート可能なボリュームを作成します。ボリュームからインスタンスを起動するには、boot\_index を指定します。指定しない場合、インスタンスは接続されたボリュームのみになります。

次の例を参考にしてください。

```
<volumes>
  <volume>
    <name>cinder-vol1X</name>
    <valid>1</valid>
    <image>cirrosX1.75</image>
    <bus>ide</bus>
    <type>lvm</type>
    <size>1</size>
    <sizeunit>GiB</sizeunit>
    <boot_index>0</boot_index>
  </volume>
</volumes>
```

### アウトオブバンドボリューム

アウトオブバンド（既存）ボリュームは、展開要求の <type> 属性を使用して指定できます。<type> 属性が指定されている場合、ESC は指定されたタイプのボリュームを照合します。

ESCは、展開要求のボリュームセクションで設定された値に基づいて、ESCによって作成されたボリュームとアウトオブバンドボリュームを区別します。VMに関連付けられたボリューム（ボリュームがESCによって作成された場合のみ）は、サービスが展開解除されるか、VMがスケールダウンされると削除されます。



- (注) アウトオブバンドボリュームを使用する場合のスケールイン/スケールアウトのサポートは使用できなくなります。

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

<type> 属性が指定されていない場合、ESC はタイプのないボリュームを照合します。

ESC は、同じ名前のボリュームを照合します。同じ名前のボリュームが複数ある場合、ESC の要求は失敗します。

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

### アウトオブバンドブート可能ボリューム

アウトオブバンドブート可能ボリューム (OpenStack のみ) は、指定されたボリュームがルートディスクとして使用される、アウトオブバンドボリュームの一種です。VM は、イメージではなくそのボリュームから起動されます。<boot\_index> 属性は、展開要求のアウトオブバンドブート可能ボリュームを指定します。

次の例を参考にしてください。

```
<volumes>
  <volume>
    <name>pre-existing</name>
    <volid>0</volid>
    <bus>ide</bus>
    <type>lvm</type>
    <boot_index>0</boot_index>
  </volume>
</volumes>
```

アウトオブバンドブート可能ボリュームには、アウトオブバンドボリュームと同様に <type> 属性の有無があります。

### パラメータの説明

- [名前 (Name) ]: 既存のボリュームの表示名を指定します。
- [Volid]: ボリュームが接続される順序を指定します。これらは、各 VM グループの 0 または 1 から始まる連続した番号です。

- [バス (Bus) ] : 接続するボリュームのバスタイプを指定します。
- [タイプ (Type) ] : (任意) <type> を指定すると、ESC は指定されたタイプのボリュームを照合します。
- [サイズ (Size) ] および [サイズ単位 (Sizeunits) ] : ESC によって作成されたボリュームを定義します。
- boot\_index : (任意) ブート順序を指定します。VM をイメージからブートする場合と同様に、任意のボリュームからブートするには 0 に設定します。この設定を機能させるには、OpenStack でそのボリュームの「ブート可能」プロパティを true に設定する必要があります。

### テナントボリューム API

テナントボリューム API を使用すると、展開要求の外部でボリュームを作成および削除できます。テナントボリューム API は、テナントの直下にボリュームを作成します。ボリュームを作成するには、テナントの詳細を入力する必要があります。

テナントボリューム NETCONF API 要求のサンプルは次のとおりです。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>admin</name>
      <volumes>
        <volume>
          <name>some-volume</name>
          <type>lvm</type>
          <size>1</size>
          <sizeunit>GiB</sizeunit>
        </volume>
      </volumes>
    </tenant>
  </tenants>
</esc_datamodel>
```

テナントボリューム API を使用して、既存のテナントを使用するボリュームを作成することもできます。この場合、ボリューム名はそのテナントに対して一意である必要があります。



- (注)
- テナントボリューム API は、NETCONF API と REST API の両方でサポートされています。
  - テナントボリューム API を使用して、エフエメラルボリュームまたはアウトオブバンドボリュームを作成または削除することはできません。
  - ESC によってのみ管理されるボリュームは削除できます。
  - テナントボリューム API を使用して既存のボリュームを更新することはできません。

### テナントボリューム API によって作成されたボリュームによる展開

ESCは、テナントボリュームAPIによって作成されたボリュームをアウトオブバンドボリュームとして扱います。テナントボリュームAPIによって作成されたボリュームを展開するには、展開データモデルで<size> および <sizeunit> パラメータを指定する必要があります。<size> および <sizeunit> パラメータが使用できない場合、ESCはテナントボリュームAPIによって作成されたボリュームを検索します。存在しない場合、ESCは他のESCまたは他のユーザによって作成された他のアウトオブバンドボリュームを探します。アウトオブバンドボリュームが使用できない場合、展開要求は拒否されます。

テナントボリュームAPIを使用して作成されたボリュームによる展開要求の例を次に示します。

```
<?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>admin</name>
      <deployments>
        <deployment>
          <name>admin-with-volume</name>
          <vm_group>
            <name>cirros</name>
            <bootup_time>60</bootup_time>
            <recovery_wait_time>0</recovery_wait_time>
            <image>Automation-Cirros-Image</image>
            <flavor>Automation-Cirros-Flavor</flavor>
            <volumes>
              <volume>
                <name>some-volume</name>
                <valid>1</valid>
                <bus>ide</bus>
              </volume>
            </volumes>
            <interfaces>
              <interface>
                <nicid>0</nicid>
                <network>mynetwork</network>
              </interface>
            </interfaces>
            <scaling>
              <min_active>1</min_active>
              <max_active>1</max_active>
              <elastic>true</elastic>
            </scaling>
            <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_collector>
                  <type>ICMPPing</type>
                  <nicid>0</nicid>
                  <poll_frequency>3</poll_frequency>
                  <polling_unit>seconds</polling_unit>
                  <continuous_alarm>>false</continuous_alarm>
                </metric_collector>
              </kpi>
            </kpi_data>
          </deployment>
        </deployments>
      </tenant>
    </tenants>
  </esc_datamodel>
```

```
        </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 />
      </vm_group>
    </deployment>
  </deployments>
</tenant>
</tenants>
</esc_datamodel>
```

ボリュームの `<size>` および `<sizeunit>` パラメータを指定すると、ESC は展開の一部としてこれらの値を使用する新しいボリュームを作成します。新しいボリュームはエフェメラルボリュームとして扱われます。



---

(注) エフェメラルボリュームの場合、最小および最大のスケーリング値は1以上にできますが、テナントおよびアウトオブバンドボリュームの場合の値は1のみです。

---

