既存環境への導入

OpenStack および ESC データ調整をサポートするためのブラウンフィールドの機能拡張

ブラウンフィールド展開：

ブラウンフィールド展開は、VIM 上のライブ VNF をターゲット ESC VM が管理できるようにする ESC VNF 展開です。

ブラウンフィールド展開は、実際のライブ VNF を中断することなく、ライブ VNF 管理をソース ESC VM からターゲット ESC VM に移行するのに役立ちます。ブラウンフィールド展開プロセスでは、新規および既存の ESC API を使用して、VIM 上に実際にリソースを作成することなく、ターゲット ESC VM 上の ESC データストア内に展開データが作成されるため、必要に応じて、既存の VIM リソースを検証するだけで済みます。

ブラウンフィールド展開のクイックスタート：

my-brownfield-import.xml および my-brownfield-deployment.xml であるブラウンフィールド XML ファイルが存在する場合は、以下のように、ブラウンフィールド API を使用してターゲット ESC VM に展開を作成します。

ブラウンフィールドデータの作成：

データを作成し、エラーを修正します（エラーが返された場合）。

ペイロードの例

```
admin@esc_vm]$ esc_nc_cli import-deployment-data CREATE my-tenant my-deployment
/tmp/my-brownfield-import.xml
Import Deployment Data
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=esc-nc-admin
 --privKeyFile=/home/admin/.ssh/confd_id_rsa --privKeyType=rsa
--rpc=/tmp/tmp_esc_nc_cli.jtQHTuOubE
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <imported_data xmlns="http://www.cisco.com/esc/esc">
```

```
    <import>
      <deployment_name>dep-complete</deployment_name>
      <project_name>dave-2000</project_name>
      <vms>
        <vm_details>

<generated_name>my-deployment_vm-gro_0_a4e82d3e-a3a5-403e-b321-cc0d7b1a779e</generated_name>
"<!-- Output removed for brevity --> "
        </vm_details>
      </vms>
    </import>
  </imported_data>
</rpc-reply>
```

VNF の展開

VNF を展開したら、SERVICE_ALIVE 通知を待ちます。エラーが発生した場合は、VNF を展開解除し、エラーを修正して VNF を再展開します。

ペイロードの例：

```
[admin@esc_vm]$ esc_nc_cli edit-config /tmp/my-brownfield-deployment.xml
Configure
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=esc-nc-admin
 --privKeyFile=/home/admin/.ssh/confd_id_rsa --privKeyType=rsa
--edit-config=/tmp/tmp_esc_nc_cli.53L6syLBhl
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <ok/>
</rpc-reply>
```

展開の完了：

この状態では、ESC VM が VNF を管理します。

ペイロードの例：

```
[admin@esc_vm]$ esc_nc_cli import-deployment-data FINALIZE my-tenant my-deployment
Import Deployment Data
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=esc-nc-admin
 --privKeyFile=/home/admin/.ssh/confd_id_rsa --privKeyType=rsa
--rpc=/tmp/tmp_esc_nc_cli.LY8Ai01yuz
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <imported_data xmlns="http://www.cisco.com/esc/esc">
    <import>
      <deployment_name>dep-complete</deployment_name>
      <project_name>dave-2000</project_name>
      <vms>
        <vm_details>

<generated_name>my-deployment_vm-gro_0_a4e82d3e-a3a5-403e-b321-cc0d7b1a779e</generated_name>
"<!-- Output removed for brevity --> "
        </vm_details>
      </vms>
    </import>
  </imported_data>
</rpc-reply>
```

インポートデータの削除：

（注） 次の手順は省略可能です。

```
[admin@esc_vm]$ esc_nc_cli import-deployment-data DELETE my-tenant my-deployment
Import Deployment Data
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=esc-nc-admin
 --privKeyFile=/home/admin/.ssh/confd_id_rsa --privKeyType=rsa
--rpc=/tmp/tmp_esc_nc_cli.LY8Ai01yuz
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <imported_data xmlns="http://www.cisco.com/esc/esc">
    <import>
      <deployment_name>dep-complete</deployment_name>
      <project_name>dave-2000</project_name>
      <vms>
        <vm_details>

<generated_name>my-deployment_vm-gro_0_a4e82d3e-a3a5-403e-b321-cc0d7b1a779e</generated_name>
"<!-- Output removed for brevity --> "
        </vm_details>
      </vms>
    </import>
  </imported_data>
</rpc-reply>
```

必要なファイル：

ブラウンフィールド展開では、**展開 XML** と**インポート XML** である VNF を指定するために、少なくとも 2 つのファイルが必要です。元の展開時に、GEN_VPC_CHASSIS_ID および GEN_VPC_SSH_KEYS である VPC ライフサイクル管理（LCM）スクリプトを VNF が使用する場合、生成された値を追加のファイルで指定します。同じことが、「chassisID および user_key」セクションで説明されています。

- 展開 XML：

展開 XML には、元の展開中に使用された XML データの詳細が示されます。XML は、テナントの vim_mapping 属性が true に設定されている元の dep.xml と同一であるのが理想的ですが、必要に応じて、ConfD API を使用して、ソース ESC VM で展開 XML を正確に生成できます。

- インポート XML：

インポート XML には、VNF VM とその一時リソース（インバンドリソースとも呼ばれる）を記述する XML データの詳細が示されます。リソースデータは OpenStack に存在し、ESC はブラウンフィールド展開中にリソースデータを選択します。ボリューム、ネットワーク、サブネット、ポートなど、すべてのインバンドリソースがサポートされています。

（注） インポート XML と展開 XML の両方で、ターゲット展開に必要なすべてのインバンドリソースと VIM リソースが指定されているため、両方の XML が互いに同期していることが非常に重要です。

chassisID と user_key：

ブラウンフィールド展開中に、元の VNF 展開でタイプ GEN_VPC_CHASSIS_ID、GEN_VPC_SSH_KEYS、またはその両方の LCM アクションが指定されている場合、LCM アクションの実行によって作成された値をファイルで指定し、適切に参照する必要があります。

ブラウンフィールド展開の前に、GEN_VPC_CHASSIS_ID LCM アクションは chassisID と呼ばれる単一の文字列を生成し、生成された値はターゲット ESC VM のローカルファイルに保存されます。

GEN_VPC_SSH_KEYS アクションは、user_key と呼ばれる SSH 公開キーと秘密キーのペアである 2 セットの SSH 関連データを生成し、生成された値はターゲット ESC VM 上のローカルファイルに保存されます。

サポートされている API：

ブラウンフィールド展開には、**ConfD API と ConfD または ESCManager REST API** の 2 つの主要な API が必要です。

- ConfD API では CREATE および FINALIZE キーワードを使用します。ConfD API へのアクセスには、esc_nc_cli スクリプトを使用するのが最も一般的です
- ConfD または ESCManager REST API では、展開 XML を処理します。

ConfD：ブラウンフィールド CREATE：

ConfD ブラウンフィールド CREATE API は、VM の VNF リソースデータであるインポート XML データとその一時データを ESC にロードして、VNF 展開中に参照可能にするために使用されます。

ConfD ブラウンフィールド CREATE API には、テナント名、展開名、インポート XML を指定するファイルの 3 つの必須引数が必要です。

使用例：

```
esc_nc_cli import-deployment-data CREATE my-tenant my-deployment
/tmp/my-brownfield-import.xml
```

呼び出し時に、インポート XML コンテンツの構造と XML 構文が検証され、ESC データベースに保存されます。インポート XML 内の実際のデータは展開時にのみ検証できるため、正確性について検証されません。以下を参照してください。

検証：

呼び出し時に、以下の検証手順を実行し、エラーがある場合は、データを修正して再送信します。

- XML 構文を検証し、必須の値が存在することを確認します。
- インポート XML で、テナント名と展開名の両方を検証して、値が一致することを確認します。
- インデックスが指定されている場合は、VM グループの 0 から開始する必要があります。

ConfD または ESCManager REST API：DEPLOY

インポート XML データが読み込まれて検証されたら、ConfD または ESCManager REST API を使用して、非ブラウンフィールド展開データの指定方法と同じ方法で、展開 XML データを展開します。

使用例：

```
esc_nc_cli edit-config /tmp/my-brownfield-deployment.xml
```

検証：

ESC は、呼び出し時に、ConfD ブラウンフィールド CREATE 中に作成された同じテナントと展開に対するブラウンフィールド CREATE 操作がすでに存在するか確認し、展開が存在する場合は、最初にすべてのリソースデータを検証して、次のことを確認します。

- 以前読み込まれたインポート XML に、展開 XML で指定されたすべての一時リソースが含まれているかどうか。
- すべてのリソースが VIM に存在するかどうか。

次のいずれかが失敗すると、使い慣れたメッセージングを使用して展開自体が失敗します。

初期の検証に合格した展開は、非ブラウンフィールド展開と同じ通知サイクルを経て、最終的にワークフローの適切なポイントで SERVICE_ALIVE 通知またはエラー通知が生成されます。

（注） リソースの不足や VIM 接続の問題など、展開中にエラーが発生した場合、ブラウンフィールド展開は標準の ESC 展開解除 API を使用して展開を解除し、根本的な問題が修正されると、再展開が試行されます。サイクルは次のとおりです。

`展開 --> エラー --> 展開解除 --> 問題の修正 --> 展開` は、エラーのない SERVICE_ALIVE 通知が受信されるまで無制限に実行できます。

ConfD：ブラウンフィールド FINALIZE

ConfD ブラウンフィールド FINALIZE API は、非ブラウンフィールドの VNF に従って VIM で VNF を管理可能なターゲット ESC VM に通知するために使用されます。

テナント名と展開名の 2 つの必須引数が必要です。

ブラウンフィールドの CREATE および DEPLOY API が使用されている期間中、この FINALIZE API が呼び出される前に、ターゲットの ESC VM は VNF をモニターしますが、モニタリングが失敗した場合、リカバリシナリオはトリガーされません。

たとえば、VNF が突然 ping 不能になった場合、ターゲットの ESC VM はリカバリポリシーを呼び出さず、通知も生成しないため、VIM 上の VNF がアクティブなときにこの最終ステップが発生することが重要です。

さらに、ターゲット ESC VM は、START、STOP、RECOVER、REBOOT、ENABLE/DISABLE MONITOR などの VNF アクションを実装していません。これらのアクションが試行されると、エラーが返されます。

検証

SERVICE_ALIVE ステータスを持つブラウンフィールド展開は「ファイナライズ済み」です。

使用例：

```
esc_nc_cli import-deployment-data FINALIZE my-tenant my-deployment
```

ConfD：ブラウンフィールド DELETE インポートデータ

ConfD ブラウンフィールド DELETE API は、インポートテーブルからすべてのブラウンフィールドデータを削除するために使用されます。

テナント名と展開名の 2 つの必須引数が必要です。

使用例

```
esc_nc_cli import-deployment-data DELETE my-tenant my-deployment
```

検証

「ファイナライズ済み」のブラウンフィールド展開のみデータを削除できます。

インポート XML の例：

以下は、インポート XML ファイルまたはデータスニペットの例の一部です。

基本 VM、およびエフェメラルボリュームとポート：

以下は、単一の VM、1 つのエフェメラルボリューム、および 2 つのエフェメラルポートがある VNF を示しています。一方のポートは単一スタック構文を使用して指定され、もう一方のポートはデュアルスタック構文を使用して指定されます。

基本 VM

```
<?xml version="1.0"?>
<import>
  <deployment_name>my-deployment</deployment_name>
  <project_name>my-tenant</project_name>
  <vms>
    <vm_details>
      <name>my-vm</name>
      <uuid>f4cad63c-a1c1-48ef-a3cd-8dd20abd2118</uuid>
      <vm_group>my-vm-group</vm_group>
      <attached_volume>
        <volume_id>df49a4a5-7def-450c-9881-b886b1abbe7f</volume_id>
        <volume_name>my-inband-volume</volume_name>
      </attached_volume>

<generated_name>my-deployment_vm-gro_0_2b92d247-7c08-48b1-a9f4-ff0849a82153</generated_name>

      <port>
        <port_id>4c2aa65d-e3fa-4529-a3f5-099031ffe6c3</port_id>
        <nicid>0</nicid>
      </port>
      <port>
        <port_id>2c627b23-ce8d-482a-9ea2-21d77df611b9</port_id>
        <fixed_ips>
          <address_id>0</address_id>
          <ip_address>192.26.13.442</ip_address>
        </fixed_ips>
        <nicid>1</nicid>
      </port>
    </vm_details>
```

```
  </vms>
</import>
```

関連のない構造を除いた、関連する展開 XML は次のように表示されます。

元の展開 XML

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>my-tenant</name>
      <vim_mapping>false</vim_mapping>  <!-- NOTE: Must always be "false" so that ESC
does not try to create a new tenant -->
      <deployments>
        <deployment>
          <name>my-deployment</name>
          <vm_group>
            <name>my-vm-group</name>
            <image>Automation-Cirros-Image</image>
            <flavor>Automation-Cirros-Flavor</flavor>
            <vim_vm_name>my-vm</vim_vm_name>
            <bootup_time>180</bootup_time>
            <recovery_wait_time>180</recovery_wait_time>
            <volumes>
              <volume>
                <name>my-inband-volume</name>
                <volid>1</volid>
                <bus>virtio</bus>
                <size>2</size>
                <sizeunit>GiB</sizeunit>
                <type>LVM</type>
              </volume>
              <volume> <!-- NOTE: out of band resources do not need to be detailed in
the import XML -->
                <name>my-out-of-band-volume</name>
                <volid>0</volid>
                <bus>virtio</bus>
                <type>LVM</type>
              </volume>
            </volumes>
            <interfaces>
              <interface>
                <nicid>0</nicid>
                <network>esc-net</network>
              </interface>
              <interface>
                <nicid>1</nicid>
                <network>esc-net</network>
                <addresses>
                  <address>
                    <address_id>0</address_id>
                    <subnet>esc-subnet</subnet>
                  </address>
                </addresses>
              </interface>
              <interface> <!-- NOTE: out of band resources do not need to be detailed
in the import XML -->
                <nicid>2</nicid>
                <port>my-out-of-band-port</port>
              </interface>
            </interfaces>
 "<!-- Output removed for brevity --> "
          </vm_group>
        </deployment>
```

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

基本 VM とエフェメラルネットワーク：

以下は、デュアルスタック構文を使用して指定された単一の VM、単一のエフェメラルネットワーク、および単一のエフェメラルポートがある VNF を示しています。

基本 VM とエフェメラルネットワークおよび基本 VM とエフェメラルボリュームの違いは、エフェメラルネットワークをインポート XML と展開 XML で詳細に指定する必要がある点です。

基本 VM

```
<?xml version="1.0"?>
<import>
  <deployment_name>my-deployment</deployment_name>
  <project_name>my-tenant</project_name>
  <vms>
    <vm_details>
      <name>my-vm</name>
      <uuid>f4cad63c-a1c1-48ef-a3cd-8dd20abd2118</uuid>
      <vm_group>my-vm-group</vm_group>

<generated_name>my-deployment_vm-gro_0_2b92d247-7c08-48b1-a9f4-ff0849a82153</generated_name>

      <port>
        <port_id>2c627b23-ce8d-482a-9ea2-21d77df611b9</port_id>
        <fixed_ips>
          <address_id>0</address_id>
          <ip_address>10.120.04.198</ip_address>
        </fixed_ips>
        <nicid>0</nicid>
      </port>
    </vm_details>
  </vms>
  <network>
    <network_id>8abd5cb3-5107-4a63-bfd4-117a6d7e3824</network_id>
    <subnet>
      <subnet_id>a74bc215-172e-41f8-9f83-f578b4fb88d2</subnet_id>
      <name>my-suvnet</name>
    </subnet>
    <name>my-network</name>
  </network>
  <network>
    <network_id>xxxxx</network_id>
      </network>
</import>
```

関連のない構造を除いた、関連する展開 *XML* を以下に示します。

元の展開 XML

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>my-tenant</name>
      <vim_mapping>false</vim_mapping>  <!-- NOTE: Must always be "false" so that ESC
does not try to create a new tenant -->
      <deployments>
        <deployment>
          <name>my-deployment</name>
```

```
          <networks>
            <network>
              <name>my-network</name>
              <shared>false</shared>
              <locator>
                <vim_id>default_openstack_vim</vim_id>
                <vim_project>davwebst</vim_project>
              </locator>
              <subnet>
                <name>my-subnet</name>
                <ipversion>ipv4</ipversion>
                <dhcp>true</dhcp>
                <address>192.168.1.150</address>
                <netmask>255.255.255.0</netmask>
                <gateway>192.168.1.1</gateway>
              </subnet>
            </network>
          <vm_group>
            <name>my-vm-group</name>
            <image>Automation-Cirros-Image</image>
            <flavor>Automation-Cirros-Flavor</flavor>
            <vim_vm_name>my-vm</vim_vm_name>
            <bootup_time>180</bootup_time>
            <recovery_wait_time>180</recovery_wait_time>
            <interfaces>
              <interface>
                <nicid>0</nicid>
                <network>my-network</network>
                <addresses>
                  <address>
                    <address_id>0</address_id>
                    <subnet>my-subnet</subnet>
                  </address>
                </addresses>
              </interface>
            </interfaces>
 "<!-- Output removed for brevity --> "
          </vm_group>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```

スケールアウトされた展開：

以下は、単一の VM とエフェメラルポートを使用した VNF を示しています。一方は、初期展開またはスケーリング KPI トリガーを通じてもう一方によってスケーリングされます。

ペイロードの例：

```
<?xml version="1.0"?>
<import>
  <deployment_name>my-deployment</deployment_name>
  <project_name>my-tenant</project_name>
  <vms>
    <vm_details>
      <index>0</index> <!-- NOTE: index must start at zero, and is mandatory if multiple
 vm details against the same vm group are specified -->
      <name>my-vm</name>
      <vm_group>my-vm-group</vm_group>

<generated_name>my-deployment_vm-gro_0_a8738b6c-1e0e-47b0-a469-db7cc63e6f92</generated_name>
```

```
      <port>
        <port_id>e974e20c-2e88-4321-beb9-5dd66e103865</port_id>
        <nicid>0</nicid>
      </port>
      <uuid>c06ab441-f895-4bd9-ab5a-9f731d42a3c1</uuid>
    </vm_details>
    <vm_details>
      <index>1</index> <!-- NOTE: index must be incremented by one if specifying vm
details for a vm in the same vm group -->
      <name>my-vm_1</name>
      <vm_group>my-vm-group</vm_group>

<generated_name>deployment-brown_vm-gro_1_a5b5bb8b-e90e-47d4-95f0-e7481abce12e</generated_name>

      <port>
        <port_id>94290268-569d-46a2-bf73-0e81d8b18019</port_id>
        <nicid>0</nicid>
      </port>
      <uuid>317369cf-f722-4052-976b-035436fee303</uuid>
    </vm_details>
  </vms>
</import>
```

（注） スケールアウトされた展開は、次のように、元の展開XMLでスケーリングの最小値と最大値を「2」に設定することによって実行されます。

```
<scaling>
  <min_active>2</min_active>
  <max_active>2</max_active>
</scaling>
```

ChassisID とユーザーキーの仕様：

ブラウンフィールド展開中に、元の VNF 展開でタイプ GEN_VPC_CHASSIS_ID、GEN_VPC_SSH_KEYS、または両方のLCMアクションが指定されている場合、LCMアクションの実行によって作成された値は、ファイルで指定され、適切に参照される必要があります。

値は、メタデータ構成エントリを使用して指定されます。次のタイプキー名がサポートされています。

- chassisID - the chassis id
- user_key - the private SSH user generated key
- user_kep.pub - the pubic SSH user generated key

以下は、単一の VM とエフェメラルポート、および GEN_VPC_CHASSIS_ID アクションと GEN_VPC_SSH_KEYS アクションの両方を使用するため、chassisID, user_key および user_key.pub がファイル内に存在し、インポート XML で参照される 3 つの値が必要なポートがある VNF を示しています。

（注） ファイル内には実際のデータが存在し、暗号化されていない値である必要があります。

ペイロードの例：

```
<?xml version="1.0"?>
<import>
  <deployment_name>my-deployment</deployment_name>
  <project_name>my-tenant</project_name>
  <vms>
    <vm_details>
      <name>my-vm</name>
      <uuid>5d892d0d-c127-40f7-8ecd-d7660461b96b</uuid>
      <vm_group>my-vm-group</vm_group>

<generated_name>deployment-brown_vm-gro_0_c0083a56-bcd9-46c9-93f2-3c0405915963</generated_name>

      <metadata>
        <configuration>
          <entry>
            <type>chassisID</type>
            <file>file:///tmp/vpc_data/chassisID</file>
          </entry>
          <entry>
            <type>user_key</type>
            <file>file:///tmp/vpc_data/user_key</file>
          </entry>
          <entry>
            <type>user_key.pub</type>
            <file>file:///tmp/vpc_data/user_key.pub</file>
          </entry>
        </configuration>
      </metadata>
      <port>
        <port_id>5c0293f9-27cf-4054-98ad-20fd8e7ed5fd</port_id>
        <nicid>0</nicid>
      </port>
    </vm_details>
  </vms>
</import>
```

（注） ファイル名のパスは、値を指定する唯一の方法です。<file> 値は「file:///」で始まる**必要があります**。

（注） 指定するVMが複数ある場合は、VMごとにブロックが繰り返されます。これは、スクリプトアクションが展開レベルにあるため、展開中にVMごとに実行され、インポートXMLでVMごとに指定する必要があるためです。

関連のない構造を除いた、関連する展開 XML を以下に示します。

元の展開 XML

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>my-tenant</name>
      <vim_mapping>false</vim_mapping>
      <deployments>
        <deployment>
          <name>my-deployment</name>
```

```
            <policies>
              <policy>
                <name>instantiate</name>
                <conditions>
                  <condition>
                    <name>LCS::PRE_DEPLOY</name>
                  </condition>
                </conditions>
                <actions>
                  <action>
                    <name>GEN_VPC_CHASSIS_ID</name>
                    <type>SCRIPT</type>
                    <properties>
                      <property>
                        <name>CHASSIS_KEY</name>
                        <value>164c03a0-eebb-44a8-87fa-20c791c0aa6d</value>
                      </property>
                      <property>
                        <name>script_filename</name>

<value>file:///opt/cisco/esc/esc-scripts/esc_vpc_chassis_id.py</value>
                      </property>
                    </properties>
                  </action>
                </actions>
              </policy>
              <policy>
                <name>instantiate_start</name>
                <conditions>
                  <condition>
                    <name>LCS::PRE_DEPLOY</name>
                  </condition>
                </conditions>
                <actions>
                  <action>
                    <name>GEN_VPC_SSH_KEYS</name>
                    <type>SCRIPT</type>
                    <properties>
                      <property>
                        <name>script_filename</name>

<value>file:///opt/cisco/esc/esc-scripts/esc-vpc-di-internal-keys.sh</value>
                      </property>
                    </properties>
                  </action>
                </actions>
              </policy>
            </policies>
            <vm_group>
              <name>my-vm-group</name>
              <image>Automation-Cirros-Image</image>
              <flavor>Automation-Cirros-Flavor</flavor>
              <vim_vm_name>my-vm</vim_vm_name>
              <bootup_time>180</bootup_time>
              <recovery_wait_time>180</recovery_wait_time>
              <interfaces>
                <interface>
                  <nicid>0</nicid>
                  <network>esc-net</network>
                </interface>
              </interfaces>
 "<!-- Output removed for brevity --> "
            </vm_group>
          </deployment>
```

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

ライフサイクル管理スクリプトの実行：

LCM スクリプトはブラウンフィールド展開中には実行されません。これは、実行されたスクリプトによりリソースが作成されるか、VM 上で 1 回実行される他のアクションが実行されるためです。ブラウンフィールド展開中に LCM スクリプトを 2 回実行すると、リソースの重複エラーまたはリソースのリークが発生します。

インポート XML 内で、すべてのスクリプトを実行しないデフォルトの設定を切り替えることで、すべてのポリシーに対してすべての LCM スクリプトをデフォルトで実行できます。次に、最上位のインポート要素である新しい親ポリシー要素を示します。

```
<import>
  <deployment_name>my-deployment</deployment_name>
  <project_name>my-tenant</project_name>
  <vms>
   "<!-- Output removed for brevity --> "
  </vms>
  <policies>
    <execute>true</execute> <!-- run all LCM scripts for all policies -->
  </policies>
</import>
```

スクリプトアクションは、VM グループごと、またはポリシー名を含む VM グループごとに指定されます。たとえば、次のスニペットは、インスタンス化ポリシーと終了ポリシーの一部であるスクリプトの my-vm-group-2 vm グループ内のスクリプトを除くすべてのスクリプトについて、デフォルトのスクリプト実行ポリシーを true に変更する例を示しています。

```
<import>
  <deployment_name>my-deployment</deployment_name>
  <project_name>my-tenant</project_name>
  <vms>
"<!-- Output removed for brevity --> "
  </vms>
  <policies>
    <execute>true</execute> <!-- run all LCM scripts for all policies -->
    <policy>
      <execute>false</execute>
      <name>instantiate</name>
      <vm_group>my-vm-group-2</vm_group>  <!-- this is optional -->
    </policy>
    <policy>
      <execute>false</execute>
      <name>terminate</name>
    </policy>
  </policies>
</import>
```

制限事項：

ブラウンフィールドモードで展開する場合、次の制限事項に注意する必要があります。

- ETSI 展開は現在サポートされていません。
- ESC ConfD のみが CREATE および FINALIZE アクションをサポートしています。つまり、ブラウンフィールド展開は、ローカルで展開する場合は confd_cli または esc_nc_cli API を

使用し、リモートで展開する場合は ConfD API を直接呼び出します。ESCManager REST API は、CREATE または FINALIZE アクションをサポートしていませんが、展開ステップをサポートするために使用できます。

- テナントの仕様では、テナントがすでに存在するために VIM エラーが発生することになるテナントの作成を回避するために、vim_mapping 属性が false に設定されています。
- VNF に複数の VM が含まれているが、ブラウンフィールド インポート中にすべての VM データが指定されていない場合は、調整されたデータを使用して VNF を展開解除して再展開します。
- OpenStack VIM でのブラウンフィールド展開のみがサポートされ、CVIM および VMWare ブラウンフィールド展開は適切なエラーで失敗します。
- ブラウンフィールド API を使用すると、1 回の呼び出しで 1 つの展開を指定できます。

ブラウンフィールドデータの生成：

ブラウンフィールドデータではインポート XML と展開 XML の両方が使用されますが、API は手動で作成されるため、生成用の特定の API はありません。

展開とテナント名を考慮して、ソース ESC VM からデータを生成する主な方法は 2 つあります。

- **ConfD API を介したインポートおよび展開 XML の生成**：

ESC VM は、ConfD API を使用して、元の展開 XML を生成し、インポート XML を手動で生成するための情報を提供します。

- **展開 XML**

次のコマンドを実行して、テナント my-tenant および展開 my-deployment の元の展開 XML を抽出して保存します。

```
[admin@esc_vm]$ echo "show running-config esc_datamodel tenants tenant my-tenant
deployments deployment my-deployment | display xml" | sudo
/opt/cisco/esc/confd/bin/confd_cli -u admin -C > /tmp/my-tenant.my-deployment.config.xml
```

結果のファイルは、元の展開 XML の複製ですが、テナントの下で vim_mapping は false に設定されます。

- **インポート XML**

以下を実行して、テナント my-tenant および展開 my-deployment のすべての運用データを抽出して保存します。

```
[admin@esc_vm]$ echo "show esc_datamodel opdata tenants tenant my-tenant deployments
my-deployment" | sudo /opt/cisco/esc/confd/bin/confd_cli -u admin -C >
/tmp/my-tenant.my-deployment.opdata.xml
```

このコマンドは、最終的な XML インポートファイルを手動で作成するために使用される、展開の運用データの概要を示す構造化 XML ドキュメントを生成します。

- **スクリプトを使用したインポートおよび展開 XML の生成**：

ブラウンフィールドインポートでは、スクリプト `/opt/cisco/esc/escscripts/export_brownfield_data.py` を使用して、変更なしで使用される展開 XML ファイルとインポート XML ファイルの両方を作成します。

スクリプトはソース ESC VM で実行されるため、このスクリプトがない古い ESC の場合は、最初にスクリプトをソース ESC VM にコピーし、ConfD API を使用して、テナントと展開名が指定されたすべての関連データを抽出する必要があります。

したがって、展開自体はConfDデータストア内に存在する必要がありますが、データは展開のステータスに関係なく抽出されるため、必ずしもSERVICE_ALIVE状態である必要はありません。

このスクリプトには、テナント名と展開名の2つの必須引数が必要です。ConfDアクセスはデフォルトでRSAキーベースのアクセスになりますが、これが有効になっていない場合は、名前とパスワードを使用して認証を実行できます。

さらに、ConfDデータのみからは収集できないメタデータに基づいて最終的なインポートXMLの出力を調整するための追加の引数が存在するため、オペレーターのアクションが必要です。

[使用状況（Usage）] ページの出力を次に示します。

export_brownfield_data.py：**[使用状況（Usage）]**

```
[admin@esc_vm] > /opt/cisco/esc/esc-scripts/export_brownfield_data.py.local -h

Usage: export_brownfield_data.py -t <tenant> -d <deployment>
                                 [-u <confd_user>] [-p <confd_password>] [-l
<local_data_directory>]
                                 [-e <[True|False]>]
[--policy:<name>:<[True|False]>:[<vm_group>]]...
                                 [-c <config_entry_path>]

Mandatory Arguments:
-t <tenant>       The deployment tenant name.
-d <deployment>   The deployment name.

Optional Arguments:
-u <confd_user>            When connecting to the ConfD API, use <confd_user>.  Defaults
 to 'admin'.
-p <confd_password>         When connecting to the ConfD API, use <confd_password> for
username/password
                            access with the <confd_user>.

NOTE: If <confd_password> is **not** set, then RPC authentication is assumed to have
been enabled and is
                            used instead when accessing the ConfD API (and <confd_user>
is ignored if set).

-l <local_data_directory>  The full path to a local directory that contains two ESC data
 files specifying
                            what the files generated by the ConfD API - the ConfD API is
 not used.
                            The two file names need to be in the format:
                              <tenant>.<deployment>.config.xml - configuration data from
 ConfD
                               <tenant>.<deployment>.opdata.xml - operational data from
ConfD

-e <[True|False]>          Execute all scripts in every policy for every VM group.
Defaults to 'False'
```

```
--policy [--policy <name>,<[True|False]>,[<vm_group>]],...[--policy
<name>,<[True|False]>,[<vm_group>]]

                          1 or more policies to execute all scripts for.  The policy
name and a boolean
                          indicating if scripts should be run for that policy.
                          If the scripts are at a VM group level, then the VM group
must be specified,
                          otherwise it is optional.

                          Incorrectly specified policies will be ignored.  Examples:

                          --policy instantiate,False --policy
VM_PRE_DEPLOY,True,vm_group_xyzzy

-c <config_entry_path>     Generates a <configuration> block under <metadata> for every
 VM in the
                         import XML with three <entry> children for chassidID, user_key
 and user_key.pub
                          The path/to/target is mandatory and used as the <file> value
 for each <entry>.

                          Specify the path using an absolute path.  Example:

                          -c /tmp/target/data
```

使用例

export_brownfield_data.py：使用例

```
[admin@esc-vm]$ ./export_brownfield_data.py -t my-tenant -d my-deployment

*** Parse and validate arguments ...
`--> Tenant = my-tenant
`--> Deployment = my-deployment
`--> Execute all scripts for all policies for all VM groups = None

*** Python version 2 ***
*** Current working directory is /home/admin/BROWNFIELD ***
*** Writing temporary files to /var/tmp/tmp4fk4Sq ***

*** Generate configuration and opdata tempfiles ...
`--> config data ...
`--> Adding <vim_mapping>false</vim_mapping> to config data XML as <locator> tag found
in the body.
`--> operational data ...

*** Generate configuration data for import - i.e. dep.xml ...
`--> Writing to /home/admin/BROWNFIELD/my-tenant.my-deployment.config.xml
`--> Creating vm_group: inband port map (if any exist) ...
`--> Found 3 interfaces.
`--> Found IN BAND port for vm group vm-group-complete with nicid 0
`--> Found IN BAND port for vm group vm-group-complete with nicid 1
`--> Creating vm_group: inband volume map (if any exist) ...
`--> Looking at vm_group name vm-group-complete
`--> Found 2 volumes.
`--> Found IN BAND volume for vm group vm-group-complete with volid 1
`--> Adding ephermeral networks if they exist ...
`--> In band networks found
`--> Adding policies if they were specified ...
`--> Skipping policies as none were specified ...
`--> Writing to /home/admin/BROWNFIELD/my-tenant.my-deployment.import.xml
*** Done ***
```

```
[admin@esc-vm]$ ls -l *.xml
total 2
-rw-r--r--.  1 admin admin  4383 May 17 11:21 my-tenant.my-deployment.config.xml
-rw-r--r--.  1 admin admin  1250 May 17 11:21 my-tenant.my-deployment.import.xml

[admin@esc-vm]$ head -20 my-tenant.my-deployment.import.xml
<?xml version="1.0"?>
<import>
  <deployment_name>my-deployment</deployment_name>
  <project_name>my-tenant</project_name>
  <vms>
    <vm_details>
      <index>0</index>
      <name>my-vm</name>
      <vm_group>my-vm-group</vm_group>
      <attached_volume>
        <volume_id>828051ba-fda8-4526-910a-caf36562cc26</volume_id>
        <volume_name>my-in-band-volume</volume_name>
      </attached_volume>

<generated_name>my-deployment-vm-gro_0_a4e82d3e-a3a5-403e-b321-cc0d7b1a779e</generated_name>

      <port>
        <port_id>5e6b0e08-8639-4965-b82f-237ad6c9fe26</port_id>
        <nicid>0</nicid>
      </port>
      <port>
        <port_id>69a69157-f47f-4514-af0c-23395185b5e8</port_id>
```

結果として得られる 2 つのファイルは、ターゲットの ESC VM に直接コピーされ、変更なしでブラウンフィールド API への入力として使用されます。

ソース ESC VM からの管理の削除：

VNF がソース ESC VM からターゲット ESC VM に正常に移行すると、VNF はソース ESC VM から削除されます。この時点で、OpenStack 上の 1 つの VNF を管理する 2 つの ESC VM があり、VNF が到達不能になった場合、両方の ESC VM が応答するため、この削除が必要になります。

考えられる 4 つの手法は次のとおりです。

- 状況によっては、他の VNF を管理していない場合、ソース ESC VM 全体がシャットダウンされます。
- ESC VM は、他の VNF を管理していない場合、データが消去されます。
- ESC VM が機能し続ける必要があり、VM がスタンドアロン構成である場合、VimManager サービスはメンテナンス期間中にシャットダウンし、テナントと展開の組み合わせに対してサポートされている ESC API を介して展開が送信されます。この結果、VimManager に到達できず、VIM が VNF の削除を確認できないために、内部 ESC 警告が発生します。ただし、これは単なる警告であり、すべての ESC および ConfD データは展開用に削除されます。
- メンテナンス期間は停止操作ができない H/A または A/A 環境で VimManager サービスを停止できない場合、展開で使用する特定の VimManager 接続は、パスワードまたは URL を間違った値に変更することで無効にできます。次に、テナントと展開の組み合わせに対し

てサポートされているESC APIを介して送信された展開を試行できます。この結果、再び内部警告が発生しますが、ESC および ConfD データは展開用に削除されます。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。