Cisco Elastic Services Controller 5.1 アドミニストレーション ガイド

初版：2020 年 1 月 20 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

目 次

このマニュアルについて

このガイドは、基本的な設定、ESCの正常性のモニタリング、システムログの表示など、ESC管理関連のタスクを実行するのに役立ちます。

• 対象読者 （v ページ）

対象読者

このガイドは、VNFのプロビジョニング、設定、およびモニタリングを担当するネットワーク管理者を対象としています。Cisco Elastic Services Controller（ESC）とその VNF は、仮想インフラストラクチャ マネージャ（VIM）に展開されます。現在、OpenStack、VMware vCenter、VMware vCloud Director、CSP 2100/5000、および Amazon Web Services（AWS）は、サポートされている VIMs です。管理者は、VIM レイヤ、vCenter、OpenStack および AWS のリソース、ならびに使用するコマンドに精通している必要があります。

Cisco ESC は、サービスプロバイダー（SP）および大企業を対象としています。ESC は、効果的かつ最適なリソース使用率を実現することにより、ネットワークの運用コストの削減に役立ちます。大企業向けに、ESCはネットワーク機能のプロビジョニング、設定、およびモニタリングを自動化します。

用語および定義

次の表で、このガイドで使用されている用語を定義します。

表 *1*：用語および定義

用語	定義
AWS	Amazon Web Services（AWS）はセキュアなクラウドサービス プラットフォームであり、コンピューティング、データベースストレージ、コンテンツ配信、その他の機能を提供します。
ESC	Elastic Services Controller（ESC）は仮想ネットワーク機能マネージャ（VNFM）であり、仮想ネットワーク機能のライフサイクル管理を実行します。

用語	定義
ETSI	欧州電気通信標準化機構（ETSI）は、欧州内の情報通信技術（ICT）の標準開発において貢献してきた独立標準化機関です。
ETSI 展開フレーバ	展開フレーバの定義には、VNF インスタンスに適用するアフィニティ関係、スケーリング、最小/最大 VDU インスタンス、その他のポリシーと制限に関する情報が含まれています。VNF 記述子（VNFD）で定義された展開のフレーバは、インスタンス化 VNF LCM 操作時に InstantiateVNFRequest ペイロードで *flavour_id* 属性を渡すことによって選択する必要があります。
HA	ESC 高可用性（HA）は、ESC のシングルポイント障害を防止し、ESC のダウンタイムを最小限に抑えるためのソリューションです。
KPI	重要業績評価指標（KPI）は、パフォーマンス管理を測定します。KPI は、どのようなパラメータをいつ、どのように測定するかを指定します。KPI には、特定のパラメータのソース、定義、測定、計算に関する情報が組み込まれています。
MSX	Cisco Managed Services Accelerator（MSX）は、企業とサービスプロバイダーの両方の顧客にクラウドベースのネットワーキングサービスを迅速に導入できるようにするサービスの作成と配信のプラットフォームです。
NFV	ネットワーク機能仮想化（NFV）は、仮想ハードウェアの抽象化を使用して実行するネットワーク機能をハードウェアから分離する原則です。
NFVO	NFV オーケストレータ（NFVO）は、ネットワークサービス（NS）のライフサイクルを管理し、NS ライフサイクル、VNF ライフサイクル（VNFM でサポート）、NFVI リソース（VIM でサポート）の管理を調整して、必要なリソースと接続の割り当てを最適化します。
NSO	Cisco Network Services Orchestrator（NSO）は、サービス アクティベーションのためのオーケストレータであり、純粋な物理ネットワーク、ハイブリッドネットワーク（物理および仮想）、および NFV の使用をサポートします。
OpenStack コンピューティングのフレーバ	フレーバで、Nova コンピューティング インスタンスのコンピューティング、メモリ、およびストレージ容量を定義します。フレーバは、サーバに使用可能なハードウェア設定です。起動可能な仮想サーバのサイズを定義します。
サービス	サービスは、1 つまたは複数の VNF で構成されます。
VDU	仮想化展開ユニット（VDU）は、情報モデルで使用できる構成要素であり、VNF のサブセットの展開と運用動作の説明、またはサブセットにコンポーネントとして含まれていない場合は VNF 全体の説明をサポートします。

用語	定義
VIM	仮想インフラストラクチャ マネージャ（VIM）は、データセンターハードウェアの管理レイヤを追加します。このノースバウンド API は、インスタンス化、終了、スケールインとスケールアウトの手順、ならびに障害とパフォーマンスのアラームの物理リソースと仮想リソースを管理するために、他のレイヤによって使用されます。
VM	仮想マシン（VM）は、オペレーティングシステム OS またはソフトウェアにインストールされているアプリケーションであり、専用ハードウェアを模倣します。エンドユーザは、仮想マシン上でも専用ハードウェア上と同じように操作できます。
VNF	仮想ネットワーク機能（VNF）は、ネットワーク機能仮想化（NFV）インフラストラクチャに展開可能なさまざまなソフトウェアとプロセスを備えた 1 つの VM または 1 つのグループの VM で構成されます。
VNFC	仮想ネットワーク機能コンポーネント（VNFC）は、VNF の複合部分であり、VDU と同義で、VM またはコンテナとして実装できます。
VNFM	仮想ネットワーク機能マネージャ（VNFM）は、VNF のライフサイクルを管理します。

関連資料

Cisco ESC のドキュメントセットは、さまざまな API を使用した VNF のインストール、設定、ライフサイクル管理操作、修復、スケーリング、モニタリング、メンテナンスの実行に役立つ次のガイドから構成されています。

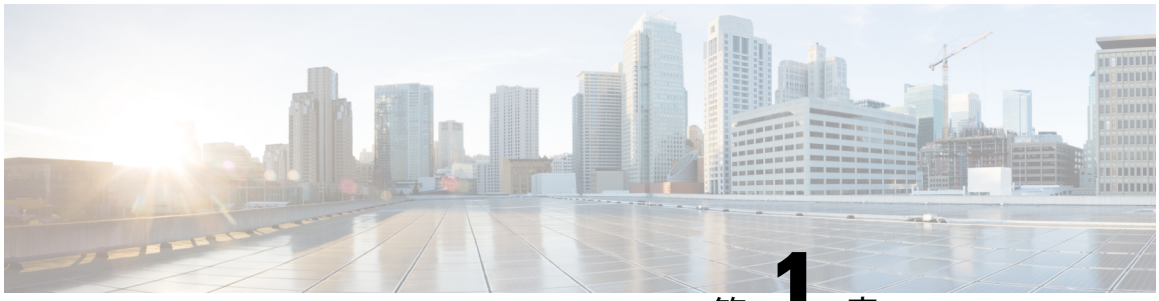
ガイド	このガイドに記載されている情報
Cisco Elastic Services Controller Release Notes	新機能とバグ、既知の問題が記載されています。
Cisco Elastic Services Controller Install and Upgrade Guide	新規インストールとアップグレードのシナリオ、インストール前後のタスク、ESC 高可用性（HA）展開の手順が記載されています。
Cisco Elastic Services Controller User Guide	VNF のライフサイクル管理操作、モニタリング、修復、スケーリングが記載されています。
Cisco Elastic Services Controller ETSI NFV MANO User Guide	ETSI API を使用した VNF のライフサイクル管理操作、モニタリング、修復、スケーリングが記載されています。
Cisco Elastic Services Controller 5.1 Administration Guide	メンテナンス、ESC の正常性のモニタリング、および ESC が生成したシステムログに関する情報が記載されています。

ガイド	このガイドに記載されている情報
Cisco Elastic Services Controller NETCONF API Guide	Cisco Elastic Services Controller NETCONF ノースバウンド API に関する情報とそれらの使用方法が記載されています。
Cisco Elastic Services Controller REST API Guide	Cisco Elastic Services Controller RESTful ノースバウンド API に関する情報とそれらの使用方法が記載されています。
Cisco Elastic Services Controller ETSI REST API Guide	Cisco Elastic Services Controller ETSI API に関する情報と、それらの使用方法が記載されています。
Cisco Elastic Services Controller Deployment Attributes	展開データモデルで使用される展開属性に関する情報が記載されています。
Cisco Elastic Services Controller Open Source	Cisco Elastic Services Controller で使用されているオープンソースソフトウェアのライセンスと通知に関する情報が記載されています。

ドキュメントの入手方法

マニュアルの入手、Cisco Bug Search Tool（BST）の使用、サービス要求の送信、追加情報の収集の詳細については、『What's New in Cisco Product Documentation』を参照してください。このドキュメントは、http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html から入手できます。

『What's New in Cisco Product Documentation』に登録します。ここには、すべての新規および改訂済みの Cisco テクニカル マニュアルが RSS フィードとして掲載されており、コンテンツはリーダー アプリケーションを使用してデスクトップに直接配信されます。RSS フィードは無料のサービスです。



第 1 章

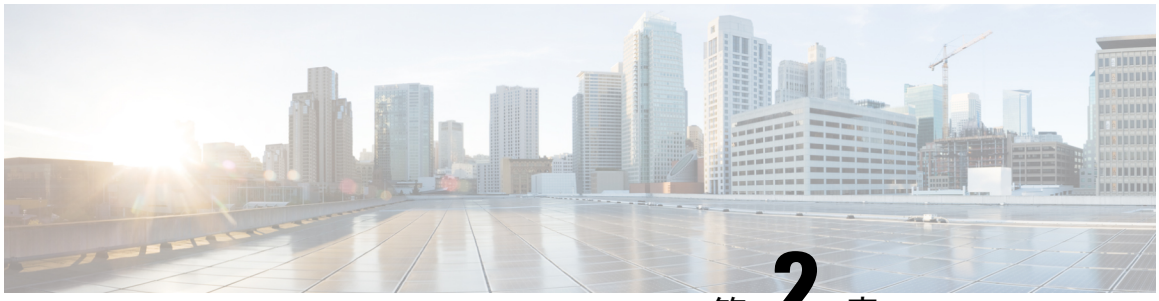
Elastic Services Controller の概要

Elastic Services Controller の概要

Cisco Elastic Services Controller（ESC）は、仮想ネットワーク機能（VNF）のライフサイクルを管理する仮想ネットワーク機能マネージャ（VNFM）です。ESC では、仮想サービスをプロビジョニングすることによって、エージェントレスのマルチベンダー VNF 管理を行えます。ESC は VNF の正常性を監視し、ネットワーク機能仮想化（NFV）環境の俊敏性、柔軟性、およびプログラマビリティを向上させます。この機能は、これらのルールの結果に基づいてトリガーされるアクションを監視し、関連付けるためのルールを定義するための柔軟性を提供します。モニタリングの結果に基づいて、ESC は VNF でスケールインまたはスケールアウトの操作を実行します。VM 障害が発生した場合、ESC は自動 VM リカバリもサポートします。

ESC は、シスコおよびその他のサードパーティ製アプリケーションと完全に統合されています。スタンドアロン製品として、ESC を VNF マネージャとして展開できます。ESC は Cisco Network Services Orchestrator（NSO）と統合し、オーケストレーションとともに VNF 管理を提供します。ESC は VNF マネージャとして、仮想マネージドサービスと、仮想パケットコア、仮想ロードバランサ、仮想セキュリティサービスなどのすべてのサービスプロバイダーの NFV 展開を対象とします。複雑なサービスには複数の VM が含まれており、それらの間に依存関係がある単一のサービスとして調整されています。

Elastic Services Controller の概要



第 2 章

インターフェイスの設定

インターフェイス設定

インターフェイス設定を使用すると、ネットワーク、サブネット、IP アドレス、MAC アドレス、VIMインターフェイス名、モデルなど、インターフェイスのさまざまな設定を選択できます。

この項では、Elastic Services Controller（ESC）の基本的インターフェイス設定および高度なインターフェイス設定と、これらを設定する手順について説明します。

基本的なインターフェイスの設定

ESC データモデルでは、インターフェイスは VM に接続されている VNIC を参照します。VM グループの下に 1 つ以上のインターフェイスを追加できます。interface セクションには、VNIC を設定するための詳細が表示されます。

この項では、Elastic Services Controller（ESC）の基本的なインターフェイス設定について説明します。

基本的なインターフェイス設定

この項では、次のような基本的なインターフェイス設定について説明します。

- ネットワーク
- サブネット
- IP アドレス
- MAC アドレス
- Elastic Services Controller（ESC）の場合は、VIM インターフェイス名など。

インターフェイス名の設定

VIM インターフェイス名を設定するには、展開 XML ファイル内のインターフェイスに属性 <vm_interface_name> を指定します。インターフェイス名を生成するときに特定の名前を使用するには、<vm_interface_name> を使用します。これらの属性が指定されていない場合、ESC はインターフェイス名を自動生成します。この名前は、deployment_name、group_name、およびランダムな UUID 文字列の組み合わせになります。例：my-deployment-na_my-gro_0_8053d7gf-hyt33-4676-h9d4-9j4a5599472t。

（注） この機能は現在、OpenStack でのみサポートされています。

VM グループに伸縮性があり、vim_interface_name が指定されている場合は、2 番目のインターフェイス名以降のインターフェイス名の後に数値インデックスが追加されます（最初の名前は変更されません）。たとえば、指定したインターフェイス名が <vim_interface_name>interface_1</vim_interface_name> と設定されており、スケーリングが 3 に設定されている場合は、3 つの異なるインターフェイス名（interface_1、interface_1_1、および interface_1_2）で 3 つの VM が作成されます。VM グループの VM が 1 つのみの場合は、カスタムインターフェイス名に「_<index>」は追加されません。単一の展開に複数の VM グループを含めることができます。また、必要に応じて、個々の VM グループで異なる vim_interface_name 値を指定できます。たとえば、展開に 2 つの VM グループがある場合、最初のグループで vim_interface_name を指定すると、すべての VM に前述のようにその名前が生成されます。2 番目の VM グループでは vim_interface_name を指定しないため、このグループから作成されたすべての VM 名が自動生成されます。同じインターフェイス名は、必要に応じて、同じ VM グループ内の別の interface セクション、展開内の別の VM グループ、または異なる展開内で使用できます。

属性 <vim_interface_name> または <port> を同じインターフェイスに使用した場合、vim_interface_name 値は無視され、port 属性内の値が使用されます。

```
<esc_datamodel xmlns="https://www.cisco.com/esc/esc"> <tenants><tenant>
<name>Admin</name>
<deployments>
<deployment>
<deployment_name>NwDepModel_nosvc</deployment_name>
<interface>
<nicid>0</nicid>
<vim_interface_name>interface_1</vim_interface_name>
<network>my-network</network>
</interface>
```

（注） インターフェイス名には最大 61 文字を使用できます。特殊文字は使用できず、英数字と「_」および「-」のみを使用できます。次に、カスタムポート名を使用した出力例を示します。展開時に vim_interface_name を設定した場合は、同じ値が出力に表示されます。展開時にこの値を設定しなかった場合は、ESC がポート名を自動生成します。

- 次に、カスタムインターフェイス名を追加した後に esc_nc_cli スクリプトを使用して取得した出力の運用データの例を示します。vim_interface_name という新しい要素がインターフェイス要素の下に表示されます。

```
[admin@esc-3-1-xxx]$ esc_nc_cli get esc_datamodel/opdata
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
. . .
    <interface>
        <nicid>0</nicid>
        <type>virtual</type>
        <port_id>e4111069-5d00-493b-8ea9-1a2ca134b5c8</port_id>
       <vim_interface_name>interface_1</vim_interface_name>      <!-- NEW IN OUTPUT
 -->
        <network>c7fafeca-aa53-4349-9b60-1f4b92605420</network>
        <subnet>255.255.255.0</subnet>
        <ip_address>192.168.2.1</ip_address>
        <mac_address>fa:16:3e:d7:5e:da</mac_address>
        <netmask>255.255.240.0</netmask>
        <gateway>192.168.2.255</gateway>
    </interface>
```

- 次に、REST API を使用して取得した出力の運用データの例を示します。

```
GET http://localhost:8080/ESCManager/v0/deployments/example-deployment-123
| xmllint --format -
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<deployments>
. . .
    <interface>
        <network_uuid>c7fafeca-aa53-4349-9b60-1f4b92605420</network_uuid>
        <gateway>172.16.0.1</gateway>
        <ip_address>172.16.12.251</ip_address>
        <mac_address>fa:16:3e:30:0c:99</mac_address>
        <netmask>255.255.240.0</netmask>
        <nic_id>0</nic_id>
        <port_forwarding/>
        <port_uuid>1773cdbf-fe5f-4af1-adff-3a9c1dd1c47d</port_uuid>
        <vim_interface_name>interface_1</vim_interface_name>         <!-- NEW IN
OUTPUT -->
        <security_groups/>
        <subnet_uuid>7b2ce63b-eb20-4ff8-8d49-e46ee8dde0f5</subnet_uuid>
        <type>virtual</type>
    </interface>
```

上記のすべてのシナリオでは、vim_interface_name が deployment .xml に指定されていない場合でもこの要素は出力に含められますが、インターフェイス名は内部生成されます。例：

```
<vim_interface_name>vm-name-deployme_Grp1_1_0f24cd7e-cae7-402e-819a-5c84087103ba</vim_interface_name>
```

MAC アドレスの割り当て

VMware vCenter での ESC の展開では、MAC アドレスプールから MAC アドレスの範囲または MAC アドレスリストを使用して MAC アドレスを割り当て、VM をネットワークに展開できます。

MAC アドレスは次の方法で割り当てることができます。

インターフェイスの使用

```
<interfaces>
    <interface>
 <nicid>1</nicid>
 <network>MANAGEMENT_NETWORK</network>
 <ip_address>172.16.0.11</ip_address>
 <mac_address>fa:16:3e:73:19:a0</mac_address>
    </interface>
</interfaces>
```

スケーリング時に、MAC アドレスリストまたは MAC アドレスの範囲を MAC アドレスプールから割り当てることができます。

```
<scaling>
 <min_active>2</min_active>
 <max_active>2</max_active>
 <elastic>true</elastic>
 <static_ip_address_pool>
  <network>MANAGEMENT_NETWORK</network>
  <ip_address>172.16.0.11</ip_address>
  <ip_address>172.16.0.12</ip_address>
  <ip_address>172.16.0.13</ip_address>
 </static_ip_address_pool>
 <static_mac_address_pool>
  <network>MANAGEMENT_NETWORK</network>
  <mac_address>fa:16:3e:73:19:a0</mac_address>
  <mac_address>fa:16:3e:73:19:a1</mac_address>
  <mac_address>fa:16:3e:73:19:a2</mac_address>
 </static_mac_address_pool>
</scaling>
```

MAC アドレスの範囲を使用して MAC アドレスを割り当てます。

```
<scaling>
 <min_active>2</min_active>
 <max_active>2</max_active>
 <elastic>true</elastic>
 <static_ip_address_pool>
  <network>MANAGEMENT_NETWORK</network>
  <ip_address_range>
   <start>172.16.0.25</start>
   <end>172.16.0.27</end>
  </ip_address_range>
 </static_ip_address_pool>
 <static_mac_address_pool>
  <network>MANAGEMENT_NETWORK</network>
  <mac_address_range>
   <start>fa:16:3e:73:19:b0</start>
   <end>fa:16:3e:73:19:b2</end>
  </mac_address_range>
 </static_mac_address_pool>
</scaling>
```

（注） 既存の展開内や、サービスアップデート内のVMインスタンスのスケーリング時（最小値および最大値が 1 よりも大きい場合）は、MAC または IP プールを変更できません。

VMware vCenter では、MAC アドレスの割り当て中に、「Generated」または「Assigned」の指定値が正しい範囲内にないか、または重複していると判断された場合、サーバがその値をオーバーライドすることがあります。このため、ESC が MAC アドレスを割り当てることができない場合、その展開は失敗します。

インターフェイスのサブネットの設定

サブネットはデータモデルを介して渡すことができます。インターフェイス内のサブネットは、展開 XML ファイルの interface セクションで指定できます。データモデルにサブネットが指定されていない場合、ESC は OpenStack にインターフェイスを作成するためのサブネットを選択し、OpenStack によって作成されたポートのサブネットを使用します。

```
<interface>
   <nicid>0</nicid>
   <network>my-network</network>
  <subnet>my-subnet</subnet>
    </interface>
```

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>
```

アウトオブバンドポートの設定

また、ESC を使用すると、アウトオブバンドポートを VNF に接続することもできます。これを行うには、サービス要求を開始している間に展開要求ファイルでUUID またはポートの名前を渡します。詳細については、『Cisco Elastic Services Controller User Guide』の「Out-of-band Volumes」の項を参照してください。

（注） VNF を展開解除または復元している間は、その VNF に接続されているポートは切り離されるだけで、削除されません。ESC は、VM グループのアウトオブバンドポートを使用している間はスケーリングを許可しません。VM グループには、VM のインスタンスを 1 つだけ設定できます。アウトオブバンドポートが使用されている間は、展開の更新時に VM グループのスケーリング値を更新できません。

```
<esc_datamodel xmlns="https://www.cisco.com/esc/esc">
    <name>tenant</name>
    <deployments>
        <deployment>
            <name>depz</name>
            <vm_group>
                <name>g1</name>
                <image>Automation-Cirros-Image</image>
                <flavor>Automation-Cirros-Flavor</flavor>
                <bootup_time>100</bootup_time>
                <reboot_time>30</reboot_time>
                <recovery_wait_time>10</recovery_wait_time>
                <interfaces>
                    <interface>
                        <nicid>0</nicid>
                        <port>057a1c22-722e-44da-845b-a193e02807f7</port>
                        <network>my-network</network>
                    </interface>
                </interfaces>
            </vm_group>
        </deployment>
    </deployments>
</esc_datamodel>
```

デュアルスタックのサポート

デュアル スタック ネットワークを使用すると、複数の IP アドレスを割り当てることができます。これらの複数の IP アドレスは、ESC を使用して、VNF 展開内の所定のインターフェイスへの異なるサブネットに割り当てることができます。

ESC では、デュアルスタックの次の機能がサポートされています。

- ネットワークとサブネットのリストの設定
- ネットワークとサブネットおよび IP アドレスのリストの設定
- ネットワークとアドレスのリストの設定（サブネットなし）
- ネットワークとサブネットおよび IP のリストの指定（同じサブネットで IP が異なる）

（注） 現在、ESC は OpenStack でのみデュアルスタックをサポートしています。ESC は、OpenStack 展開のためのエンドツーエンド IPv6 をサポートしています。

新しいコンテナ要素の名前付きアドレスがインターフェイスに追加されます。このコンテナには、アドレス要素のリストが含まれています。アドレス要素には address_id（キー）が必要で

す。サブネットおよび固定 IP アドレスのフィールドはオプションですが、いずれか 1 つを指定する必要があります。

コンテナアドレスは次のとおりです。

```
container addresses {
  list address {
    key "address_id";
    leaf address_id {
      description "Id for the address in address list.";
      type uint16;
      mandatory true;
    }
    leaf subnet {
      description "Subnet name or uuid for allocating IP to this port";
      type types:escnetname;
    }
    leaf ip_address {
      description "Static IP address for this specific subnet";
      type types:escipaddr;
      must "../../../../../scaling/max_active = 1"
      {
        error-message "Only single VM per group supported with multiple address option.";

      }
    }
  }
}
```

デュアルスタックでは、KPI のモニタリングがサポートされるようになりました。新しい子要素 address_id が metric_collector 要素に追加されました。これは、KPI のモニタリングに使用される、指定された NICID 内のアドレスをポイントする値を受け入れます。つまり、インターフェイスの下に定義されているアドレスの 1 つを KPI のモニタリングに使用できます。

```
…
    <interface>
        <nicid>1</nicid>
        <network>demo-net</network>
        <addresses>
            <address>
                <address_id>0</address_id>
                <subnet>demo-subnet</subnet>
            </address>
        </addresses>
    </interface>
 <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>5</metric_occurrences_true>
               <metric_occurrences_false>5</metric_occurrences_false>
              <metric_collector>
                  <type>ICMPPing</type>
                  <nicid>1</nicid>
                  <address_id>0</address_id>
                  <poll_frequency>10</poll_frequency>
                  <polling_unit>seconds</polling_unit>
                 <continuous_alarm>false</continuous_alarm>
                </metric_collector>
              </kpi>
```

```
            </kpi_data>
...
```

（注） metric_collector 要素の下にある address_id は、インターフェイスの下にある address_id の 1 つと同じである必要があります。

デュアルスタック インターフェイスは、day-0 変数の置換で使用できるようになりました。つまり、1 つのインターフェイスの下で定義されている複数のアドレスから値を置き換えることができます。Day 0 設定は、config_data タグの下にあるデータモデルで定義されます。

複数の IP アドレスを持つデュアルスタックの場合、変数は NICID_<n>_<a>_<PROPERTY> 形式になります。それぞれの意味は次のとおりです。

- <n> は、インターフェイスの NICID です。
- <a> は、そのインターフェイス内のアドレスの address_id です。

次に、デュアルスタックからの使用可能な day-0 置換変数のリストを示します。

NICID_n_a_IP_ALLOCATION_TYPE	FIXED | DHCP を含む文字列	ipv4 または ipv6
NICID_n_a_IP_ADDRESS	IP アドレス	ipv4 または ipv6
NICID_n_a_GATEWAY	ゲートウェイ アドレス	ipv4 または ipv6
NICID_n_a_CIDR_ADDRESS	CIDR プレフィックスアドレス	ipv4 または ipv6
NICID_n_a_CIDR_PREFIX	CIDR プレフィックス長の整数	ipv4 または ipv6
NICID_n_a_NETMASK	IPv4 CIDR アドレスとプレフィックスが存在する場合、ESC は自動的にネットマスク変数を計算して入力します。これは、IPv6 アドレスの場合は置換されないため、使用しないでください。	ipv4 のみ

1 つの IP アドレスの day-0 設定の詳細については、『Cisco Elastic Services Controller User Guide』の「Day Zero Configuration」の章を参照してください。

次に、day-0 設定の config_data で定義されているテンプレートファイルを示します。

```
NICID_0_NETWORK_ID=${NICID_0_NETWORK_ID}
NICID_0_MAC_ADDRESS=${NICID_0_MAC_ADDRESS}

NICID_0_0_IP_ALLOCATION_TYPE=${NICID_0_0_IP_ALLOCATION_TYPE}
NICID_0_0_IP_ADDRESS=${NICID_0_0_IP_ADDRESS}
NICID_0_0_GATEWAY=${NICID_0_0_GATEWAY}
NICID_0_0_CIDR_ADDRESS=${NICID_0_0_CIDR_ADDRESS}
NICID_0_0_CIDR_PREFIX=${NICID_0_0_CIDR_PREFIX}
NICID_0_0_NETMASK=${NICID_0_0_NETMASK}
```

```
NICID_0_1_IP_ALLOCATION_TYPE=${NICID_0_1_IP_ALLOCATION_TYPE}
NICID_0_1_IP_ADDRESS=${NICID_0_1_IP_ADDRESS}
NICID_0_1_GATEWAY=${NICID_0_1_GATEWAY}
NICID_0_1_CIDR_ADDRESS=${NICID_0_1_CIDR_ADDRESS}
NICID_0_1_CIDR_PREFIX=${NICID_0_1_CIDR_PREFIX}
```

次に、データモデルを示します。

```
<?xml version="1.0" encoding="ASCII"?>
<esc_datamodel xmlns="https://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>dep-tenant</name>
      <deployments>
        <deployment>
          <name>cirros-dep</name>
          <vm_group>
            <name>Grp1</name>
            <bootup_time>600</bootup_time>
            <recovery_wait_time>30</recovery_wait_time>
            <flavor>Automation-Cirros-Flavor</flavor>
            <image>Automation-Cirros-Image</image>
           <interfaces>
              <interface>
                <!-- No dual stack support on mgmt interface in ESC 4.1 -->
                <nicid>0</nicid>
                <network>my-network</network>
              </interface>
              <interface>
                <nicid>1</nicid>
                <network>ent-network1</network>
                <addresses>
                  <address>
                    <!-- IPv4 Dynamic -->
                    <address_id>0</address_id>
                    <subnet>v4-subnet_A</subnet>
                  </address>
                  <address>
                    <!-- IPv6 Dynamic -->
                    <address_id>1</address_id>
                    <subnet>v6-subnet_B</subnet>
                  </address>
                </addresses>
              </interface>
              <interface>
                <nicid>2</nicid>
                <network>ent-network2</network>
                <addresses>
                  <address>
                    <!-- IPv4 Static -->
                    <address_id>0</address_id>
                    <subnet>v4-subnet_C</subnet>
                    <ip_address>172.16.87.8</ip_address>
                  </address>
                  <address>
                    <!-- IPv6 Static -->
                    <address_id>1</address_id>
                    <subnet>v6-subnet_D</subnet>
                    <ip_address>fd07::110</ip_address>
                  </address>
                </addresses>
              </interface>
              <interface>
                <nicid>3</nicid>
```

```
                    <network>ent-network3</network>
                    <addresses>
                      <address>
                        <!--  Only ip config -  ipv6 but no subnet  -->
                        <address_id>0</address_id>
                        <ip_address>fd07::110</ip_address>
                      </address>
                      <address>
                        <!--  Only ip config - ipv4 but no subnet  -->
                        <address_id>1</address_id>
                        <ip_address>172.16.88.9</ip_address>
                      </address>
                    </addresses>
                  </interface>
                  <interface>
                    <nicid>4</nicid>
                    <network>ent-network4</network>
                    <addresses>
                      <address>
                        <!-- ipv4 same subnet as address_id 6 -->
                        <address_id>0</address_id>      //
                        <subnet>v4-subnet_F</subnet>
                        <ip_address>172.16.86.10</ip_address>
                      </address>
                      <address>
                        <!-- ipv4 same subnet as id 5 -->
                        <address_id>1</address_id>
                        <subnet>v4-subnet_F</subnet>
                        <ip_address>172.16.86.11</ip_address>
                      </address>
                    </addresses>
                  </interface>
                </interfaces>
                <kpi_data>
...
```

複数の IP を使用して正常に展開された後、ESC はアドレスのリストを通知または opdata として提供します。

次を含む <address>親 <interface> 要素の下にある複数の要素のリスト：

- **address_id**：入力 XML で指定されたアドレス ID
- **サブネット要素**：サブネット名または UUID
- **ip_address 要素**：そのサブネット上のポートに割り当てられている IP
- **プレフィックス**：サブネット CIDR プレフィックス
- **ゲートウェイ**：サブネット ゲートウェイ アドレス
- ESC 静的 IP サポート

通知:

```
                    <vm_id>1834124d-b70b-41b9-9e53-fb55d7c901f0</vm_id>
                    <name>jenkins-gr_g1_0_e8bc9a81-4b9a-437a-807a-f1a9bbc2ea3e</name>

<generated_name>jenkins-gr_g1_0_e8bc9a81-4b9a-437a-807a-f1a9bbc2ea3e</generated_name>

<host_id>dc380f1721255e2a7ea15932c1a7abc681816642f75276c166b4fe50</host_id>
```

```
                    <hostname>my-server</hostname>
                    <interfaces>
                      <interface>
                        <nicid>0</nicid>
                        <type>virtual</type>

<vim_interface_name>jenkins-gr_g1_0_e8bc9a81-4b9a-437a-807a-f1a9bbc2ea3e</vim_interface_name>

                        <port_id>4d57d4a5-3150-455a-ad39-c32fffbb10b1</port_id>
                        <mac_address>fa:16:3e:d2:50:a5</mac_address>
                        <network>45638651-2e92-45fb-96ce-9efdd9ea343e</network>
                        <address>
                          <address_id>0<address_id>
                          <subnet>6ac36430-4f58-454b-9dc1-82f7a796e2ff</subnet>
                          <ip_address>172.16.0.22</ip_address>
                          <prefix>24</prefix>
                          <gateway>172.16.0.1</gateway>
                        </address>
                        <address>
                          <address_id>1<address_id>
                          <subnet>8dd9f501-19d4-4782-8335-9aa9fbd4dab9</subnet>
                          <ip_address>2002:dc7::4</ip_address>
                          <prefix>48</prefix>
                          <gateway>2002:dc7::1</gateway>
                        </address>
                        <address>
                          <address_id>2<address_id>
                          <subnet>a234501-19d4-4782-8335-9aa9fbd4caf6</subnet>
                          <ip_address>172.16.87.8</ip_address>
                          <prefix>20</prefix>
                          <gateway>172.16.87.1</gateway>
                        </address>
                      </interface>
```

opdata の例：

```
  <interfaces>
   <interface>
      <nicid>0</nicid>
      <type>virtual</type>

<vim_interface_name>jenkins-gr_g1_0_e8bc9a81-4b9a-437a-807a-f1a9bbc2ea3e</vim_interface_name>

      <port_id>4d57d4a5-3150-455a-ad39-c32fffbb10b1</port_id>
      <mac_address>fa:16:3e:d2:50:a5</mac_address>
      <network>45638651-2e92-45fb-96ce-9efdd9ea343e</network>
      <address>
         <address_id>0</address_id>
         <subnet>6ac36430-4f58-454b-9dc1-82f7a796e2ff</subnet>
         <ip_address>172.16.0.22</ip_address>
         <prefix>24</prefix>
         <gateway>172.16.0.1</gateway>
      </address>
      <address>
         <address_id>1</address_id>
         <subnet>8dd9f501-19d4-4782-8335-9aa9fbd4dab9</subnet>
         <ip_address>2002:dc7::4</ip_address>
         <prefix>48</prefix>
         <gateway>2002:dc7::1</gateway>
      </address>
   </interface>
</interfaces>
```

また、day-0の代入値が出力データで置き換えられていることも確認できます。次に、day-0設定に値が入力された出力データの例を示します。

```
NICID_0_NETWORK_ID=45638651-2e92-45fb-96ce-9efdd9ea343e
NICID_0_MAC_ADDRESS=fa:16:3e:d2:50:a5

NICID_0_0_IP_ALLOCATION_TYPE=DHCP
NICID_0_0_IP_ADDRESS=172.16.0.22
NICID_0_0_GATEWAY=172.16.0.1
NICID_0_0_CIDR_ADDRESS=172.16.0.0
NICID_0_0_CIDR_PREFIX=24
NICID_0_0_NETMASK=255.255.255.0

NICID_0_1_IP_ALLOCATION_TYPE=DHCP
NICID_0_1_IP_ADDRESS=2002:dc7::4
NICID_0_1_GATEWAY=2002:dc7::1
NICID_0_1_CIDR_ADDRESS=2002:dc7::/48
NICID_0_1_CIDR_PREFIX=48
```

静的 IP をサポートするデュアルスタック

ESCは、静的IPをサポートするデュアルスタックをサポートします。初期設定の一部として、ユーザは設定するサブネットと IP を指定できます。

（注） ESC は、スケーリングが false または minimum/maximum = 1 の場合にのみ、静的 IP をサポートします。

アウトオブバンド ネットワークを使用して VM を作成し、静的 IP を持つサブネットのリスト（ネットワークに複数のサブネットがある）を指定すると、ESCはサブネットと対応する静的 IP の両方を適用します。

次に、2 つのサブネット（ipv4 と ipv6）が 1 つのインターフェイスに追加された例を示します。

```
  <?xml version="1.0" encoding="ASCII"?>
<esc_datamodel xmlns="https://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>dep-tenant</name>
      <deployments>
        <deployment>
          <name>cirros-dep</name>
          <vm_group>
            <name>Grp1</name>
            <bootup_time>600</bootup_time>
            <recovery_wait_time>30</recovery_wait_time>
            <flavor>Automation-Cirros-Flavor</flavor>
            <image>Automation-Cirros-Image</image>
            <interfaces>
              <interface>
                <nicid>0</nicid>
                <network>ent-network2</network>
                <addresses>
                  <address>
                    <!-- IPv4 Static -->
                    <address_id>0</address_id>
```

```
            <subnet>v4-subnet_C</subnet>
            <ip_address>172.16.87.8</ip_address>
          </address>
          <address>
            <!-- IPv6 Static -->
            <address_id>1</address_id>
            <subnet>v6-subnet_D</subnet>
            <ip_address>fd07::110</ip_address>
          </address>
        </addresses>
      </interface>
    </interfaces>
  <kpi_data>
```

VNF の展開については、「OpenStack での仮想ネットワーク機能の展開」を参照してください。

高度なインターフェイス設定

この項では、Elastic Services Controller（ESC）用の複数のインターフェイス設定と、ハードウェアのインターフェイスを設定する手順について説明します。

基本的なインターフェイス設定の詳細については、「基本的なインターフェイス設定」を参照してください。

高度なインターフェイスの設定

ESC での SR-IOV の設定

Single Root I/O Virtualization（SR-IOV）により、さまざまなゲスト オペレーティング システムを実行している複数の VM が、ホスト サーバ内の単一の PCIe ネットワーク アダプタを共有できるようになります。また、VM がネットワークアダプタとの間で直接データを移動でき、ハイパーバイザをバイパスすることで、ネットワークのスループットが増加しサーバの CPU 負荷が低下します。

OpenStack 用の ESC での SR-IOV の設定

OpenStack に ESC で SR-IOV を設定する前に、正しいパラメータを使用してハードウェアと OpenStack を設定します。

OpenStack に ESC で SR-IOV を有効にするには、インターフェイスの `type` を `direct` として指定します。次のスニペットは、データモデルの例を示しています。

```
<interfaces>
 <interface>
  <nicid>0</nicid>
  <network>my-network</network>
  <type>direct</type>
  </interface>
</interfaces>
...
```

VMware 用の ESC での SR-IOV の設定

VMware に ESC で SR-IOV を設定する前に、次の点を考慮してください。

- 目的の ESXi ホスト上で SR-IOV 物理機能を有効にします。詳細については、VMware のマニュアルを参照してください。
- SR-IOV を有効にする前に、次の重要な点を考慮してください。
  - SR-IOV が VMware でサポートされている物理ネットワークアダプタのリストを確認します。VMware のマニュアルを参照してください。
  - SR-IOV が設定されている VM でサポートされていない VM 機能のリストを確認します。VMware のマニュアルを参照してください。
  - SR-IOV を使用したクラスタ展開（データモデルの「zone」で定義）では、各 ESXi ホストに同じ物理機能があり、SR-IOV の選択ができるようになっていることを確認します。たとえば、VM が物理機能として vmnic7 を使用する場合は、各ホストに vmnic7 があり、各 vmnic7 の SR-IOV ステータスが有効になっていることを確認します。

    VMware に対して ESC で SR-IOV を有効にするには、展開データモデルでインターフェイスの `<type>` を `direct` とし、拡張子の `<name>` を `sriov_pf_selection` として指定します。インターフェイスタイプの `direct` は、SR-IOV デバイスを示し、拡張子名の `sriov_pf_selection` は物理機能を示します。次のスニペットは、データモデルの例を示しています。

```
<vm_group>
...
<interface>
  <nicid>2</nicid>
  <network>MgtNetwork</network>
  <type>direct</type>
</interface>
<interface>
  <nicid>3</nicid>
  <network>MgtNetwork</network>
  <type>direct</type>
</interface>
...
<extensions>
 <extension>
 <name>sriov_pf_selection</name>
 <properties>
 <property>
 <name>nicid-2</name>
 <value>vmnic1,vmnic2</value>
 </property>
 <property>
 <name>nicid-3</name>
 <value>vmnic3,vmnic4</value>
 </property>
 </properties>
 </extension>
 </extensions>
</vm_group>
```

許可済みアドレスペアの設定

Cisco Elastic Services Controller を使用すると、ネットワークに関連付けられているサブネットに関係なく、指定されたポートを通過するように展開データモデル内にアドレスペアを指定できます。

アドレスペアは、次の方法で設定されます。

- ネットワークのリスト：特定のインターフェイスにネットワークのリストを指定すると、ESC はこれらのネットワークの OpenStack からサブネットの詳細を取得し、対応するポートまたはインターフェイスに追加します。次に、ネットワークのリストとしてアドレスペアを設定する例を示します。

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

- アドレスのリスト：アドレスのリストを指定した場合、ESC はこれらのアドレスを対応するインターフェイスに追加します。次の例では、アドレスのリストとしてアドレスペアを設定する方法について説明します。

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

セキュリティグループのルールの設定

Cisco Elastic Services Controller（ESC）を使用すると、セキュリティグループのルールを OpenStack で展開されているインスタンスに関連付けることができます。これらのセキュリティグループのルールは、展開データモデルで必要なパラメータを指定することによって設定されます。セキュリティグループのルールの設定に加えて、いずれかの VNF インスタンスで障害が発生した場合、ESC はインスタンスを復旧し、再展開されている VNF のセキュリティグループのルールを適用します。

セキュリティグループのルールを設定するには、次の手順を実行します。

始める前に

- ESC を使用してテナントを作成したことを確認します。
- セキュリティグループが作成されていることを確認します。
- セキュリティグループの名前または UUID があることを確認します。

ステップ 1 ルートユーザとして ESC VM にログインします。

ステップ 2 次のコマンドを実行して特定のセキュリティグループの UUID を確認します。

```
nova --os-tenant-name <NameOfTheTenant> secgroup-list
```

ステップ 3 展開データモデルでは、次の引数を渡します。

```
<interfaces>
 <interface>
  <nicid>0</nicid>
  <network>my-network</network><!-- depends on network name -->
  <security_groups>
  <security_group>0c703474-2692-4e84-94b9-c29e439848b8</security_group>
  <security_group>bbcdbc62-a0de-4475-b258-740bfd33861b</security_group>
  </security_groups>
</interface>
<interface>
 <nicid>1</nicid>
 <network>sample_VmGrpNet</network><!--depends on network name  -->
 <security_groups>
 <security_group>sample_test_SQL</security_group>
 </security_groups>
</interface>
```

ステップ 4 次のコマンドを実行して、セキュリティグループが VM インスタンスに関連付けられているかどうかを確認します。

```
nova --os-tenant-name <NameOfTenant> show <NameOfVMinstance>
```

ハードウェアアクセラレーションのサポート（OpenStack のみ）

flavor データモデルを使用して OpenStack のハードウェア アクセラレーション機能を設定できます。次のハードウェア アクセラレーション機能を設定できます。

- **vCPU ピニング**：vCPU（仮想中央処理装置）またはある範囲の CPU へのバインディングおよびバインディング解除を可能にし、プロセスが任意の CPU ではなく、指定した CPU 上でのみ実行されるようにします。

- **大規模なページおよび不均等なメモリアクセス（NUMA）の OpenStack パフォーマンスの最適化**：大規模なページや NUMA のシステムパフォーマンス、つまり、高い負荷を受け入れ、高い負荷を処理するようにシステムを変更するようにシステムの能力を向上させることができます。
- **PCIe パススルーインターフェイスに対する OpenStack サポート**：OpenStack 上のインスタンスへの PCI デバイスの割り当てを可能にします。

次に、フレーバデータモデルを使用してハードウェアアクセラレーション機能を設定する方法について説明します。

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

VMware vSphere NUMA 属性の追加パラメータの作成

ESC は、追加の設定パラメータを追加することによって VMware vSphere の NUMA を拡張します。

この機能拡張によって、day-0 コンフィギュレーション ファイルを介してこれらの値を渡すのではなく、設定パラメータを渡すためのプレフィックスとして、VMware vSphere の追加設定または高度な設定が追加されます。

プレフィックス：`extConfigParam`

例：

```
<configuration>
    <dst>extConfigParam:mgmt-ipv4-addr</dst>
    <data>$NICID_1_IP_ADDRESS/16</data>
</configuration>
```

追加設定は、データモデルの変更を最小限に抑え、設定変更を VIM レイヤに制限するのに便利です。

VMware vCenter での PCI または PCIe デバイスのパススルーの設定

ESC は VMware vCenter PCI または PCIe デバイスパススルー（VMDirectPath I/O）をサポートします。これにより、I/O メモリ管理ユニットが搭載されたプラットフォーム上の物理 PCI 機能への VM アクセスが可能になります。

はじめる前に

ホスト VM の PCI/PCIe デバイスでパススルーを有効にするには、vSphere 管理者が vCenter でこれらのデバイスをマークする必要があります。

（注） PCI 設定後にホストをリブートする必要があります。ホストをメンテナンスモードにし、電源をオフにするか、またはすべての VM を他のホストに移行します。

ESC 展開で PCI デバイスパススルー要求を指定するには、値を ***passthrough*** に設定して <type> 属性を含めます。次に、データモデルを示します。

```
<tenants>
  <tenant>
   <name>admin</name>
   <deployments>
    <deployment>
     <name>test</name>

     <vm_group>
      <name>test-g1</name>
      <image>uLinux</image>
      <bootup_time>300</bootup_time>
      <recovery_wait_time>10</recovery_wait_time>
      <interfaces>
       <interface>
        <nicid>1</nicid>
        <network>MgtNetwork</network>
        <ip_address>192.168.0.102</ip_address>
       </interface>
       <interface>
        <nicid>2</nicid>
        <network>VM Network</network>
        <type>passthru</type>
        <ip_address>172.16.0.0</ip_address>
       </interface>
       <interface>
        <nicid>3</nicid>
        <network>VM Network</network>
        <type>passthru</type>
        <ip_address>192.168.46.117</ip_address>
       </interface>
      </interfaces>
```

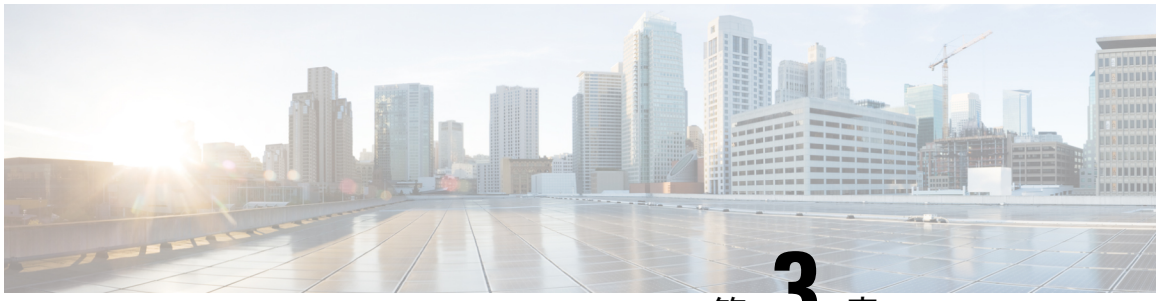
展開が正常に完了すると、***passthru*** 値が通知の interface セクションと運用データ内に設定されます。

PCI または PCIe PassThrough デバイスの自動選択

ESC では、特定の PCI ID を使用せずに、各展開に 1 つ以上の PCI または PCIe パススルーデバイスを接続する必要があります。ESC は最初にホストを選択します。ESC は、次に使用可能な PCI または PCIe パススルー対応デバイスを選択し、展開時に接続します。使用可能な PCI または PCIe パススルー対応デバイスがない場合、ESC は展開に失敗します。vSphere 管理者は、ターゲット コンピューティング クラスタ内のすべてのコンピューティングホストに、十分な数の PCI または PCIe パススルー対応デバイスがあることを確認する必要があります。

（注）

- PCI または PCIe パススルーは、ESC 配置アルゴリズムでは考慮されません。たとえば、ESC は PCI または PCIe パススルー要求を完了するために使用可能なリソースがあるため、ホストを選択しません。
- ESC は PCI または PCIe パススルーデバイスをランダムに選択します。ESC では、デバイスのタイプまたは仕様を考慮しません。リストから次に使用可能な PCI または PCIe デバイスを選択します。
- ESC が ESC 配置アルゴリズムに基づいて選択したコンピューティングホストに対して VNF が回復される場合に、そのコンピューティングホストに使用可能な PCI または PCIe パススルー対応デバイスがないと、リカバリは失敗します。
- パススルーが機能するには、DRS をオフにする必要があります。



第 3 章

ESC 正常性のモニタリング

ESC とそのサービスの正常性を監視するには、次のいずれかを使用します。

- ヘルス API のモニタリング
- SNMP トラップ

REST API を使用した ESC の正常性のモニタリング

ESC は、ESC およびそのサービスの正常性を監視するためのサードパーティ製ソフトウェアに REST API を提供します。サードパーティ製ソフトウェアは API を使用して ESC が正常な状態であるかを定期的に照会し、ESC が稼働中であるかどうかを確認できます。クエリへの応答として、API はステータスコードとメッセージを提供します。詳細については、表 2 : ESC ヘルス API のステータスコードとメッセージ （24 ページ）を参照してください。HA セットアップでは、仮想 IP（VIP）をモニタリング IP として使用する必要があります。戻り値で、ESC HA ペアの全体的な状態が示されます。詳細については、表 3 : スタンドアロン ESC と HA のヘルス API ステータスメッセージ （26 ページ）を参照してください。

ESC の正常性を監視する REST API は次のとおりです。

```
GET to https://<esc_vm_ip>:60000/esc/health
```

（注） ヘルス API のモニタリングは、既存の REST の基本的な HTTP 認証を使用して保護されます。ユーザは ESC REST API クレデンシャルを使用してレポートを取得できます。

次に、エラー状態のヘルス API のモニタリングの応答を示します。

```
<?xml version="1.0" encoding="UTF-8" ?>
<esc_health_report>
<status_code>{error status code}</status_code>
<message>{error message}</message>
</esc_health_report>
```

XML 応答と JSON 応答は、ヘルス API のモニタリングでもサポートされています。

API 応答が成功すると、 *stage* という追加のフィールドが導入されます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<esc_health_report>
<status_code>{success status code}</status_code>
<stage>{Either INIT or READY}</stage>
<message>{success message}</message>
</esc_health_report>
```

stage フィールドには、INIT パラメータまたは READY パラメータが含まれています。

INIT：INIT パラメータは ESC が設定パラメータの設定や VIM コネクタの登録などの事前プロビジョニング要求を受け入れる初期段階のものです。

READY：ESC は、このパラメータを使用した展開、展開解除などのあらゆるプロビジョニング要求に対応できます。

ESC の正常性の状態が次のステータスコードとメッセージで示されます。2000 シリーズのステータスコードは、ESC が動作していることを意味します。5000 シリーズのステータスコードは、1 つ以上の ESC コンポーネントが稼働していないことを意味します。

（注） ESC のヘルス API は VIM のステータスを確認しません。これは、複数 VIM の展開が ESC リリース 3.0 で導入されたためです。

表 *2: ESC* ヘルス *API* のステータスコードとメッセージ

ステータス コード	メッセージ
2000	ESC サービスが実行されています。（ESC services are running.）
2010	ESC サービスは実行されていますが、ESC ハイアベイラビリティノードに到達できません。（ESC services are running, but ESC High Availability node is not reachable.）
2020	ESC サービスが実行されています。（ESC services are running.）1 つ以上の VIM サービス（KeystoneやNova）に到達できません。（One or more VIM services (for example, keystone and nova) not reachable.） （注） ESC リリース 3.0 以降はサポートされていません。

ステータス コード	メッセージ
2030	ESC サービスは実行されていますが、VIM クレデンシャルが指定されていません。（ESC services are running, but VIM credentials are not provided.） （注） ESC リリース 3.0 以降はサポートされていません。
2040	ESC サービスが実行されています。VIM が設定されており、ESC が VIM への接続を初期化しています。（ESC services running. VIM is configured, ESC initializing connection to VIM.）
2100	ESC サービスは実行されていますが、ESC ハイアベイラビリティノードに到達できません。1 つ以上の VIM サービス（Nova など）に到達できません。（ESC services are running, but ESC High-Availability node is not reachable. One or more VIM services (for example, nova) are not reachable.） （注） ESC リリース 3.0 以降はサポートされていません。
5010	ESC サービス、ESC_MANAGER が実行されていません。（ESC service, ESC_MANAGER is not running.）
5020	ESC サービス、CONFD が実行されていません。（ESC service, CONFD is not running.）
5030	ESC サービス、MONA が実行されていません。（ESC service, MONA is not running.）
5040	ESC サービス、VIM_MANAGER が実行されていません。（ESC service, VIM_MANAGER is not running.）
5090	複数の ESC サービス（ConfD や Mona など）が実行されていません。（More than one ESC service (for example, confd and mona) are not running.）

（注） ESC HA モードでは、DRBD セットアップでのみ ESC HA を参照します。ESC HA セットアップの詳細については、『Cisco Elastic Services Controller Install Guide』を参照してください。

次の表では、スタンドアロン ESC のステータスメッセージと、成功シナリオと障害シナリオの HA について説明します。ESC のスタンドアロンおよび HA のセットアップの詳細については、『Cisco Elastic Services Controller Install Guide』を参照してください。

表 *3:* スタンドアロン *ESC* と *HA* のヘルス *API* ステータスメッセージ

	Success	Partial Success	Failure
スタンドアロン ESC	応答はヘルス API のモニタリングから収集され、ステータスコードは 2000 になります。	NA	• モニタは、ヘルス API のモニタリングからの応答を取得できません。 • 応答はヘルス API のモニタリングから収集され、ステータスコードは 5000 シリーズで返されます。

	Success	Partial Success	Failure
HA の ESC（アクティブ/スタンバイ）	応答はヘルス API のモニタリングから収集され、ステータスコードは 2000 になります。	応答はヘルス API のモニタリングから収集され、ステータスコードは 2010 になります。これは、ESC スタンバイノードが ESC HA の ESC マスターノードに接続できないことを示します。ただし、これはノースバウンドへの ESC サービスには影響しません。	

	Success	Partial Success	Failure
			• モニタは、2 分以上にわたってヘルス API のモニタリングの応答を取得できません。 （注） HA スイッチオーバー時の特定の期間は ESC のヘルス API のモニタリングが使用できない場合があります。モニタリングソフトウェアは、このシナリオでサービス障害を報告するように適切なしきい値を設定する必要があります。 • 応答はヘルス API

	Success	Partial Success	Failure
			のモニタリングから収集され、ステータスコードは 5000 シリーズで返されます。

SNMP トラップ通知を使用した ESC の正常性のモニタリング

また、SNMP エージェントを使用し、SNMP トラップを介してさまざまな ESC コンポーネントの正常性に関する通知を設定することもできます。このエージェントは、標準の ESC インストールの一部としてインストールされ、SNMP バージョン 2c プロトコルをサポートしています。SNMP トラップは現在、ESC で管理されている VNF ではなく、ESC 製品の状態のみをサポートしています。この項では、ESC SNMP エージェントを設定するために必要な手順について説明します。また、通知の一部としてトリガーされるイベントについても説明します。

始める前に

- **CISCO-ESC-MIB** ファイルと **CISCO-SMI MIB** ファイルがシステムで使用できることを確認します。これらのファイルは /opt/cisco/esc/snmp/mibs ディレクトリにあります。これらのファイルを SNMP マネージャマシンにダウンロードし、$HOME/.snmp/mibs ディレクトリに配置します。
- SNMP エージェントを設定します。SNMP エージェントを設定するには、次の 3 つの方法があります。これらの方法については、次の項で詳しく説明します。

SNMP エージェントの設定

SNMP トラップを受信するには、SNMP エージェントパラメータを設定します。エージェントは、この項で説明する 3 つの異なる方法を使用して設定できます。使用する最良または最適な方法は、用途によって異なります。

手順

- **BootVM を介した ESC のインストール時の SNMP エージェントの設定**：ESC のインストール中に、次の追加パラメータを使用して SNMP エージェントを設定します。

```
% bootvm.py <esc_vm_name> --image <image-name> --net <net-name> --enable-http-rest
--ignore-ssl-errors
--managers "udp:ipv4/port" or "udp:[ipv6]/port"
```

- **ESCADM を使用した設定**：ESCADM ツールを使用して、マネージャや ignoreSslErrors プロパティなどの SNMP エージェント設定パラメータを変更できます。

```
sudo escadm snmp set --ignore_ssl_errors=true --managers="udp:ipv4/port" or
"udp:[ipv6]/port"
```

- **コンフィギュレーション ファイルの更新**：設定は、/opt/cisco/esc/esc_database/snmp.conf ファイルにあります。このファイルは JSON 形式です。次に例を示します。

```
{"sysDescr": "ESC SNMP Agent",
    "listeningPort": "2001",
    "managers": [
        "udp:[ipv4]/port",
        "udp:[ipv6]/port"
    ],
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysLocation": "Unspecified",
    "sysName": "system name",
    "pollSeconds": "15",
    "listeningAddress": "0.0.0.0",
    "healthUrl": "https://<esc_vm_ip>:60000/esc/health",
    "sysContact": "root@localhost"}
```

次の表で、設定可能なプロパティについて説明します。

（注） BootVM と ESCADM ツールを使用すると、ignoreSslErrors とマネージャのパラメータのみを設定できます。

表 *4:* エージェント設定パラメータ

変数	説明
listeningAddress	リッスンするネットワーク インターフェイスを指定します。0.0.0.0 はすべてのインターフェイスを意味します。
listeningPort	リッスンする UDP ポートを指定します。
ignoreSslErrors	SSL接続の証明書を検証する場合は、「no」を指定します。信頼できない自己署名証明書の場合は、「yes」に設定します。
healthUrl	ヘルスモニタ API の URL を指定します。 （注） エージェントは、Java-8 がインストールされているラップトップなど、ESCマシンの外部で使用できます。この場合、ヘルス URL は ESC マシンを指し、ESC モニタで認証するための ESC ユーザ名/パスワード（authUser/authPass）を指定する必要があります。

変数	説明
authUser	ヘルスモニタ API の HTTP-BASIC ユーザ名を指定します。このパラメータは、エージェントが ESC マシンの外部にある場合にのみ使用します。
authPass	ヘルスモニタ API の HTTP-BASIC パスワードを指定します。このパラメータは、エージェントが ESC マシンの外部にある場合にのみ使用します。
managers	「udp:ipv4/port」または「udp:[ipv6]/port」の形式で、SNMP トラップを配信する文字列の配列を指定します。エージェントの実行中にこれらが変更されると、設定がリロードされ、新しいマネージャが今後のトラップに使用されます。
pollSeconds	このパラメータは、スケジューラで使用されます。このパラメータを使用して、ヘルスモニタ API へのポーリング間隔を秒数で指定します。
logLevel	レベルを INFO、ERROR、WARN、DEBUG、TRACE、ALL、および OFF として指定します。このパラメータは、実行時にのみ変更できます。
sysName	（SNMPv2-MIB）このノードの名前を示すオプションの値。デフォルトでは、ホスト名が使用されます。
sysDescr	（SNMPv2-MIB）このエージェントを説明するオプションの値。
sysLocation	（SNMPv2-MIB）このノードの物理的な場所を指定するオプションの値。
sysContact	（SNMPv2-MIB）このノードに関する照会の連絡先の名前または電子メールアドレスを指定するオプション値

ESC SNMP MIB の定義

次の表で、ESC MIB の内容について説明します。これらの値は、snmp.conf ファイルで設定できます。

変数	Simple IOD	説明
sysName	SNMPv2-MIB::sysName.0	ESC マシンの名前を指定します。デフォルトでは、ホスト名が取得されます。

変数	Simple IOD	説明
sysDescr	SNMPv2-MIB::sysDescr.0	SNMP エージェントの名前を指定します。
sysLocation	SNMPv2-MIB::sysLocation.0	ESC マシンが配置されている場所を指定します。
sysContact	SNMPv2-MIB::sysContact.0	管理者の連絡先を指定します。

SNMP トラップ通知の有効化

ESCADM ツールを使用して、SNMP サービスを開始します。

```
sudo escadm snmp start
```

また、ESCADM ツールを使用してステータスの取得を停止したり、SNMP エージェントの設定を変更できます。

```
sudo escadm snmp stop
sudo escadm snmp status
sudo escadm snmp restart
```

ESC での SNMP トラップの管理

この項の内容は、次のとおりです。

- ESC での SNMP 通知タイプについて
- ネットワークからの SNMP トラップメッセージの直接受信
- トラップエンドポイントの管理（SNMP マネージャ）
- HA 環境での ESC SNMP の管理
- ESC での自己署名証明書の管理

手順

- **ESC での SNMP 通知タイプについて**：次の表に、このバージョンの SNMP エージェントでサポートされているすべてのイベントを示します。これらのステータスコードとメッセージは、ESC の状態が変更された場合にのみ、登録されたマネージャに SNMP トラップを介して返されます。2000 シリーズのステータスコードは、ESC が動作していることを意味します。5000 シリーズのステータスコードは、1 つ以上の ESC コンポーネントが稼働していないことを意味します。2000 シリーズおよび 5000 シリーズのステータスコードの詳細については、「REST API を使用した ESC の正常性のモニタリング」の項を参照してください。

ステータス コード	SNMP エージェント固有のメッセージ
5100	ESC モニタ API の使用時に HTTP エラーが発生しました。
5101	ESC モニタは応答しましたが、データを理解できませんでした。
5102	エージェントは ESC モニタ API へのネットワーク接続を作成できませんでした。
5199	未処理のエラーが発生しました（詳細はメッセージに示されます）。
5200	HA 環境では、HA マスターノードに変更があると、エージェントはこの通知を送信します。

- **ネットワークからの SNMP トラップメッセージの直接受信**：snmpget snmpwalk や snmptrapd などの基本的な SNMP UNIX ツールを使用して、ネットワークから SNMP トラップメッセージを直接受信します。使用例：

```
snmptrapd -m ALL -f -Lo -c snmptrapd.conf <port>
```

これにより、ポート 12113 で SNMP トラップデーモンが開始されます。シスコおよび ESC MIB が ~/.snmp/mibs に存在することを確認します。参照された snmptrapd.conf は次のようになります。

```
disableAuthorization yes
authCommunity   log,execute,net public
# traphandle default /Users/ahanniga/bin/notify.sh esc

createUser myuser MD5 mypassword DES myotherpassword

format2 %V\n% Agent Address: %A \n Agent Hostname: %B \n Enterprise OID: %N \n Trap
 Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n PDU Attribute/Value
 Pair Array:\n%v \n -------------- \n
```

トラップには、statusCode と statusMessage の 2 つのエントリが含まれます。このトラップは、ステータスが変化したときに送信されます。

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (3971) 0:00:39.71
SNMPv2-MIB::snmpTrapOID.0 = OID: CISCO-ESC-MIB::statusNotif
SNMPv2-MIB::sysDescr.0 = STRING: ESC SNMP Server
CISCO-ESC-MIB::escStatusCode.0 = STRING: "2000"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
```

- **トラップエンドポイントの管理（SNMP マネージャ）**：SNMP エージェントは、変更の設定ファイルを監視し、変更が行われたときにリロードします。コンフィギュレーションファイルに対してマネージャのエンドポイントを追加または削除し、以降のトラップでは新しい設定が使用されます。

- **HA 環境での ESC SNMP エージェントの管理**：2 つ以上の ESC ノードを HA 設定で展開し、SNMP エージェントがこの設定をサポートします。ただし、HA 展開では次の点を考慮してください。
  - 1 つの ESC ノード（マスターノード）のみが SNMP トラップを送信できます。
  - バックアップノードがマスターになった場合は、SNMP エージェントが稼働している必要があります。
  - マスター設定に加えた変更は、バックアップノードにも適用する必要があります。
  - フェールオーバーが原因でノードがマスターノードになると、トラップが生成されます。

- **自己署名証明書の管理**：ESC が展開されて SNMP エージェントが ESC のヘルス API を使用する場合は、サーバにルート信頼証明書をインストールしておくことを推奨します。環境が既知であり、信頼できるものである場合、設定パラメータ「ignoreSslErrors」を使用してこれらのエラーを無視することができます。ただし、この設定をよりセキュアなデフォルトに維持する場合は、ESC 証明書を JVM 信頼ストアにインポートすることによって、自己署名証明書をインストールできます。次の項では、これを実行する手順について説明します。

  a) localhost の代替名として esc を追加します。ファイル「/etc/hosts:」で次のように追加します（または、「esc」が最後に追加されていることを確認します）。

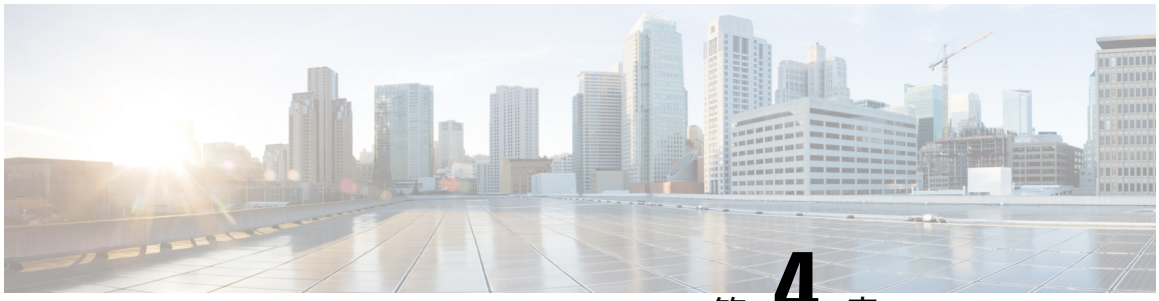
  ```
  127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
  esc
  ```

  b) SNMP エージェント コンフィギュレーション ファイル「/opt/cisco/esc/esc_database/snmp.conf」では、healthUrl が ESC を指している必要があります。

  ```
  "healthUrl": "https://esc:60000:/esc/health"
  ```

  c) 証明書をトラストストアにインポートします。次に、$JAVA_HOME is/usr/lib/jvm/jre-1.8.0-openjdk.x86_64 を想定し、証明書をインポートする例を示します。

  ```
  cd /opt/cisco/esc/esc-config
  sudo openssl x509 -inform PEM -in server.pem -outform DER -out server.cer
  sudo keytool -importcert -alias esc -keystore $JAVA_HOME/lib/security/cacerts
  -storepass changeit -file server.cer
  ```



第 4 章

ESC のシステムログ

ESC ログメッセージの表示

ログメッセージは、VNF ライフサイクル全体にわたって ESC イベント用に作成されます。これらには、外部メッセージ、ESC から他の外部システムへのメッセージ、エラーメッセージ、警告、イベント、障害などがあります。ログファイルは、`/var/log/esc/escmanager_tagged.log` にあります。

次に、ログメッセージの形式を示します。

```
date=<time-date>] [loglevel=<loglevel>] [tid=<transactionid>] [cl=<classifications>]
[tags=<tags>] [msg=<message>
```

次に、ログの例を示します。

```
date=15:43:58,46022-Nov-2016]
[loglevel=ERROR ] [tid=0793b5c9-8255-47f3-81e6-fbb59f6571f7] [cl=OS ]
[tags=wf:create_vm,eventType:VM_DEPLOY_EVENT,tenant:CSCvd94541,depName:test-dep,vmGrpName:test-VNF,
vmName:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0]
[tags=wf:create_vm,eventType:VM_DEPLOY_EVENT,tenant:test,depName:test-dep,vmGrpName:test-VNF,
vmName:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0] [msg=sleepingfor5seconds
to allow vm to become ACTIVE instance id:
162344f7-78f9-4e45-9f23-34cf87377fa7
name:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0
```

要求を受信すると、一意のトランザクション ID を自動生成する RequestDetails オブジェクトが作成されます。この値は、すべてのスレッドで転送されます。分類とタグは任意です。これらは、読みやすくするためにログメッセージに追加されたプレフィックスであり、デバッグに役立ちます。分類とタグを使用すると、ログメッセージを簡単に解析し、ログ分析ツールでフィルタリングすることができます。

次に、サポートされている分類を示します。

NBI	「com.cisco.esc.rest」「com.cisco.esc.filter」（ノース バウンド インターフェイス：証明書）

SBI	「com.cisco.esc.rest」：ソースはコールバックハンドラまたは「EventsResource」（サウスバウンドインターフェイス、ESC と VIM 間）
SM	「com.cisco.esc.statemachines」は StateMachine を意味します。この分類は、StateMachine カテゴリのログを示します。
MONITORING	「com.cisco.esc.monitoring」「com.cisco.esc.paadaptor」（MONA 関連ログ）
DYNAMIC_MAPPING	「com.cisco.esc.dynamicmapping」「com.cisco.esc.db.dynamicmapping」（MONA 関連ログ）
CONFD	「com.cisco.esc.confd」
CONFD_NOTIFICATION	「com.cisco.esc.confd.notif」「com.cisco.esc.confd.ConfdNBIAdapter」
OS	「com.cisco.esc.vim.openstack」
LIBVIRT	「com.cisco.esc.vim.vagrant
VIM	「com. esc. vim」
REST_EVENT	「ESCManager_Event」「com.cisco.esc.util.RestUtils」。ログ内の REST 通知を示します。
WD	「com.cisco.esc.watchdog」
DM	「com.cisco.esc.datamodel」「com.cisco.esc.jaxb.parameters」（データモデルとリソースオブジェクト）
DB	「com.cisco.esc.db」（データベース関連ログ）
GW	「com.cisco.esc.gateway」
LC	「com.cisco.esc.ESCManager」（スタートアップ関連ログ）
SEC	「com.cisco.esc.jaas」
MOCONFIG	「com.cisco.esc.moconfig」（MOCONFIG オブジェクト関連ログ。これは ESC 開発者用の内部ログです）
POLICY	「com.cisco.esc.policy」（サービス/VM ポリシー関連ログ）
TP	「com.cisco.esc.threadpool」
ESC	「com.cisco.esc」上記にないその他のパッケージ

次に、サポートされているタグを示します。

- **ワークフロー [wf:]**：RequestDetails オブジェクトのアクションとリソースを使用して生成されます。例：「wf: create_network」
- **イベントタイプ [eventType:]**：現在のアクションをトリガーしたイベント。例：「eventType:VM_DEPLOY_EVENT」
- **リソースベース**：これらの値は、イベントで使用されるパラメータのタイプに基づいて生成されます。階層 （テナント、VM グループなど）がログに追加されます。

テナント	[tenant:<tenant name>]
ネットワーク	[tenant:<tenant id>, network:<network name>] （注） テナントは、該当する場合にのみ表示されます。
サブネット	[tenant:<tenant name or id>, network:<network name or id>, subnet:<subnet name>] （注） テナントは、該当する場合にのみ表示されます。
ユーザ	[tenant:<tenant name>, user:<user name or id>] （注） テナントは、該当する場合にのみ表示されます。
イメージ	[image:<image name>]
フレーバ	[flavor:<flavor name>]
配置	[tenant:<tenant name or id>, depName:<deployment name>]
DeploymentDetails	[tenant:<tenant name or id>, depName:<deployment name>, vmGroup:<vm group name>, vmName:<vm name>]
スイッチ	[tenant:<tenant name or id>, switch:<switch name>]
音量	[volume:<volume name>]
サービス	[svcName:<Service Registration name>]

さらに、分析とログの管理を促進するため、ESC ログを rsyslog サーバに転送することもできます。

ConfD API を使用したログのフィルタリング

ConfD API に導入されたログフィルタを使用して、ESC でログ（展開ログやエラーログなど）を照会および取得できます。テナント、展開名、およびVM名の新しいフィルタが導入されました。これにより、ConfD APIのログフィルタを使用して、最新のエラーログのESCログをさらに照会することができます。ESC と OS 間の通信に関連する ESC ログを取得することもできます（分類タグを「OS」に設定します）。

次に、ConfD API ログを取得するためのログ形式を示します。

```
date=<time-date>] [loglevel=<loglevel>] [tid=<transactionid>] [cl=<classifications>]
[tags=<tags>] [msg=<message>
```

次に、サンプルログの例を示します。

```
date=15:43:58,46022-Nov-2016] [loglevel=ERROR ] [tid=0793b5c9-8255-47f3-81e6-fbb59f6571f7]
 [cl=OS ]
[tags=wf:create_vm,eventType:VM_DEPLOY_EVENT,tenant:test,depName:test-dep,vmGrpName:test-VNF,
vmName:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0]
[msg=sleepingfor5seconds to allow vm to become ACTIVE instance id:
162344f7-78f9-4e45-9f23-34cf87377fa7
name:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0
```

ログレベル、分類、およびタグのパラメータは、ログを取得するために相互に依存します。次の組み合わせを使用してログを正常に取得できます。

- log_level=ERROR, classifications=OS, tags=(depName:test-dep)
- log_level=ERROR, classifications=OS, tags=(tenant: test)

ログフィルタは、次の条件がすべて満たされたときに値を返します。

- ログ レベル
- 分類（指定されている場合）
- タグ（指定されている場合）

（注） 複数の分類がリストされている場合は、1 つ以上の分類に一致する必要があります。同じことが、タグにも適用されます。

たとえば、次のログフィルタ条件では、前述のログサンプルを返しません。

```
log_level=ERROR, classifications=VIM, tags=(depName:test-dep)
```

ログレベルとタグが一致していても、分類の VIM が一致していないので値は返されません。

次に、データモデルを示します。

```
rpc filterLog {
    description "Query and filter escmanager logs using given parameters";
    tailf:actionpoint escrpc;
    input {
      leaf log_level {
        mandatory false;
       description "One of DEBUG / INFO / WARNING / ERROR / TRACE / FATAL. Results will
 include all logs at and
                     above the level specified";
        type types:log_level_types;
        default ERROR;
      }
      leaf log_count {
        mandatory false;
        description "Number of logs to return";
        type uint32;
        default 10;
      }
      container classifications {
        leaf-list classification {
          description "Classification values to be used for the log filtering.  For
example: 'OS', 'SM'.
                      Logs containing any of the provided classification values will be
```

```
returned.";
         type types:log_classification_types;
       }
     }
     container tags {
       list tag {
         key "name";
         leaf name {
           mandatory true;
         description "Tag name to be used for the log filtering.  For example: 'tenant',
'depName'.
                      Logs containing any of the provided tag name plus the tag values
will be returned.";
           type types:log_tag_types;
         }
         leaf value {
           mandatory true;
          description "Tag value pairs to be used for the log filtering.  For example:
'adminTenant', 'CSRDeployment'";
           type string;
         }
       }
     }
   }
   output {
     container filterLogResults {
       leaf log_level {
         description "Log level used to filter for the logs.";
         type types:log_level_types;
       }
       list logs {
         container classifications {
           leaf-list classification {
             description "Classifications used to filter for the logs.";
             type types:log_classification_types;
           }
         }
         container tags {
           list tag {
             key "name";
             leaf name {
               mandatory true;
               description "Tag name used to filter for the logs.";
               type types:log_tag_types;
             }
             leaf value {
               mandatory true;
               description "Tag value used to filter for the logs.";
               type string;
             }
           }
         }
         leaf log_date_time {
           description "Timestamp of the log.";
           type string;
         }
         leaf log_message {
           description "The log message.";
           type string;
         }
       }
     }
   }
```

```
}
```

NETCONF コンソールまたは esc_nc_cli を使用して、ConfD API ログを照会できます。

- NETCONF コンソールを使用して、次のクエリを実行します。

```
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=admin
--privKeyFile=/home/admin/.ssh/confd_id_dsa --privKeyType=dsa --rpc=log.xml
```

- esc_nc_cli を使用して、次のクエリを実行します。

```
./esc_nc_cli filter-log log.xml
```

次に、log.xml の例を示します。

```
<filterLog xmlns="https://www.cisco.com/esc/esc">
  <log_level>INFO</log_level>
  <log_count>1</log_count>
  <classifications>
    <classification>OS</classification>
    <classification>SM</classification>
  </classifications>
  <tags>
    <tag>
      <name>depName</name>
      <value>CSR_ap1</value>
    </tag>
    <tag>
      <name>tenant</name>
      <value>admin</value>
    </tag>
  </tags>
</filterLog>
```

応答は次のとおりです。

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <filterLogResults xmlns="https://www.cisco.com/esc/esc">
    <log_level>INFO</log_level>
    <logs>
      <classifications>
        <classification>OS</classification>
        <classification>SM</classification>
      </classifications>
      <tags>
        <tag>
          <name>depName</name>
          <value>CSR_ap1</value>
        </tag>
        <tag>
          <name>tenant</name>
          <value>admin</value>
        </tag>
      </tags>
      <log_date_time>13:06:07,575 31-Oct-2016</log_date_time>
      <log_message> No pending work flow to start.</log_message>
    </logs>
  </filterLogResults>
</rpc-reply>
```

（注） ロギング API の応答は XML 形式です。ログメッセージに XML 文字が含まれている場合はその文字がエスケープされるため、XML 準拠は解除されません。

ESC ログファイルの表示

次の表に、さまざまな ESC コンポーネントのログを示します。

ファイル	コンポーネント	説明	ローテーションサイズ	バックアップファイルの数	アクティブ/アクティブ展開
/var/log/esc/escmanager.log	ESCManager	これには、ワークフロー、要求、および持続性を含む ESC マネージャのログが含まれています。	150 MB	10	応対可
/var/log/esc/escmanager_tagged.log	ESCManager	これは escmanager.log と同じですが、netconf ロギング API 用に読みやすい形式が使用されており、必要に応じて他のパーサーで使用できます。	150 MB	10	応対可

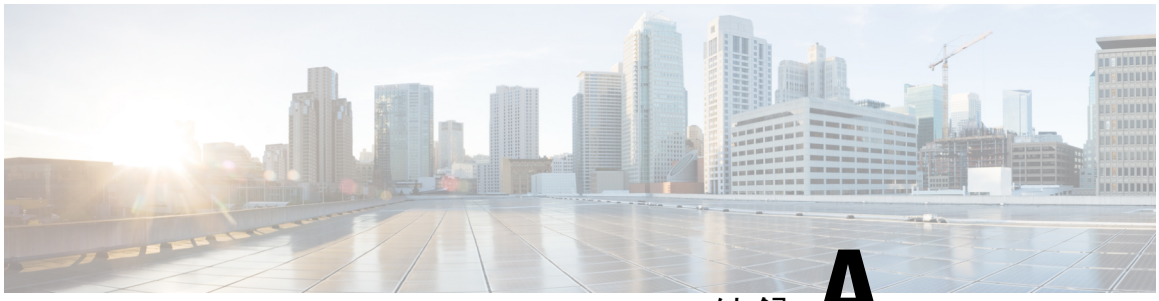
ファイル	コンポーネント	説明	ローテーションサイズ	バックアップファイルの数	アクティブ/アクティブ展開
/var/log/esc/yangesc.log	ESCManager	これには、netconf 要求と通知に関連するログが含まれています。	150 MB	10	応対可
/var/log/esc/error_escmanager.log	ESCManager	すべてのエラーログエントリ。	150 MB	10	応対可
/var/log/esc/event_escmanager.log	ESCManager		150 MB	10	応対可
/var/log/esc/trace/event_escmanager.log	ESCManager		150 MB	10	応対可
/var/log/esc/trace/escdatabase.log	ESCManager	データベース関連のログエントリ			応対可
/var/log/esc/trace/debug_yangesc.log	ESCManager		51 MB	2	応対可
/var/log/esc/trace/esc_rest.log	ESCManager		150 MB	10	応対可
/var/log/esc/mona/mona.log	MONA		150 MB	10	応対可
/var/log/esc/mona/actions_mona.log	MONA		150 MB	10	応対可
/var/log/esc/mona/rules_mona.log	MONA		150 MB	10	応対可
/var/log/esc/vimmanager/vimmanager.log	VIM マネージャサービス	詳細な VIM マネージャのログ。	150 MB	10	応対可
/var/log/esc/vimmanager/operations_vimmanager.log	VIM マネージャサービス	VIM マネージャの操作が処理されたことのみを示す簡単なログ。	150 MB	10	応対可

ファイル	コンポーネント	説明	ローテーションサイズ	バックアップファイルの数	アクティブ/アクティブ展開
/var/log/esc/trace/vimmanager/vim_vimmanager.log	VIM マネージャサービス	VIM マネージャと VIM 間の Raw HTTP 要求/応答（ヘッダーを含む）。このログを追跡する OpenStrack では、ログレベルが VIM のデバッグに設定されている必要があることに注意してください。。	150 MB	10	応対可
/var/log/esc/<timestamp>-esc-portal-be.log	ESC UI		10 MB	4	応対可
/var/log/esc/confd/audit.log	confd		10 MB	4	応対可
/var/log/esc/confd/browser.log	confd		10 MB	4	応対可
/var/log/esc/confd/confd.log	confd		10 MB	4	応対可
/var/log/esc/confd/devel.log	confd		10 MB	4	応対可
/var/log/esc/confd/netconf.log	confd		10 MB	4	応対可
/var/log/esc/confd/netconf.trace	confd		10 MB	4	応対可
/var/log/esc/confd/global.data			循環なし		応対可
/var/log/esc/esc_monitor.log	ESC インフラまたは HA		10 MB	4	使用不可
/var/log/esc/esc_monitor_output.log	ESC インフラまたは HA		10 MB	4	使用不可

ファイル	コンポーネント	説明	ローテーションサイズ	バックアップファイルの数	アクティブ/アクティブ展開
/var/log/esc/esc_confd.log	ESC インフラまたは HA		10 MB	4	使用不可
/var/log/esc/pgstartup.log	ESC インフラまたは HA		10 MB	4	使用不可
/var/log/esc/spy.log	ESC インフラまたは HA		ログなし（サイズ0）	ESC で生成されたログはありません。	使用不可
/var/log/esc/catalina.out	Tomcat		循環なし	ESC で生成されたログはありません。エラーのみ。	応対可
/var/log/esc/esc_dbtool.log	DB ツール		循環なし		応対可
/var/log/esc/snmp/snmp.log	SNMP エージェント		循環なし		使用不可
/var/log/esc/etsi-vnfm/etsi-vnfm.log	ETSI サービス	これは、要求、応答、ペイロード、および必要に応じた一般的なロギング情報を含む、ETSI 処理の主要なログファイルです。	150MB	10	応対可

ファイル	コンポーネント	説明	ローテーションサイズ	バックアップファイルの数	アクティブ/アクティブ展開
/var/log/esc/etsi-vnfm/events-etsi-vnfm.log	ETSI サービス	API 要求のみ受信と発信の両方についてログに記録します。	150MB	10	応対可
/var/log/esc/etsi-vnfm/event-details-etsi-vnfm.log	ETSI サービス	実際の JSON ペイロードとともに、API 要求（受信と発信）の両方をログに記録します。	150MB	10	応対可
/var/log/esc/escadm.log	Escadm サービス	escadm.py から手動と自動の両方のメッセージとエラーをキャプチャするためにログに記録します。このログは、ESC のスタートアップおよび設定変更を追跡する場合に役立ちます。	10 MB	4	応対可
/var/log/esc/elector-<pid>	Elector サービス	ログエントリは、リーダーシップの決定を記録します。	循環なし		アクティブ/アクティブのみで使用可能

ファイル	コンポーネント	説明	ローテーションサイズ	バックアップファイルの数	アクティブ/アクティブ展開
/var/log/esc/consul_agent-<timestamp>	Consul エージェント	Consul サーバを使用して ESC Consul エージェントを記録するログエントリ。	循環なし		アクティブ/アクティブのみで使用可能



付録 A

ESC のエラー状態

ESC 操作のエラー状態

ESC 操作のエラー状態

ESC で操作が失敗した場合、ユーザはその操作をキャンセルする必要があります。ESC は、操作をキャンセルするために自動的にロールバックすることはありません。次の表に、エラー状態とリカバリの詳細を示します。

エラー状態の通知またはロギングの詳細

通常、すべてのエラー状態について、REST インターフェイスを使用している場合はコールバック、NETCONF インターフェイスを使用している場合は netconf 通知で、障害が発生した要求のエラー通知が NB クライアント（ESC ユーザ）に送信されます。syslog が設定されている場合は、エラーログが生成され、syslog に送信されます。

エラー状態	リカバリ
テナント作成要求に失敗しました（Failed create tenant request）	NB クライアント（ESC ユーザ）は、同じテナント作成要求で送信を試みる前に、テナント削除要求で送信する必要があります。
ネットワーク作成要求に失敗しました（Failed create network request）	NB クライアント（ESC ユーザ）は、同じネットワーク作成要求で送信を試みる前に、ネットワーク削除要求で送信する必要があります。
サブネット作成要求に失敗しました（Failed create subnet request）	NB クライアント（ESC ユーザ）は、同じサブネット作成要求で送信を試みる前に、サブネット削除要求で送信する必要があります。

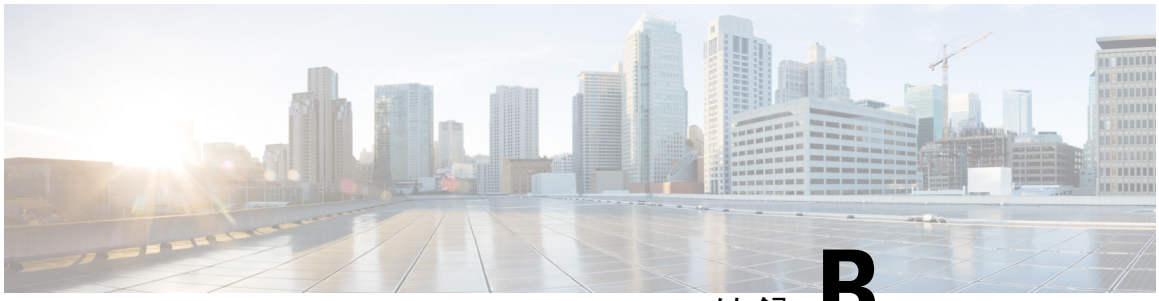
ESC のエラー状態

エラー状態	リカバリ
展開要求に失敗しました（Failed deployment request）	NB クライアント（ESC ユーザ）は、同じ展開要求で送信を試みる前に、展開解除要求で送信する必要があります。 展開が失敗した場合、ESC は、展開解除要求を受信するまで、データベース内の情報（エラー状態のある）を更新します。展開が解除されると、エラー状態にあるオブジェクトが削除されます。
リカバリが失敗しました（Filed Recovery）	既存の展開は使用できなくなりました。NB クライアント（ESC ユーザ）は、展開解除要求で送信した後、同じ展開要求で送信する必要があります。
スケールアウト/インに失敗しました（Failed Scale Out/In）	対処不要です。既存の展開がまだ機能しています。後の段階で展開解除がトリガーされた場合は、失敗したスケールアウトとスケールインで影響を受けた部分の VM をクリーンアップします。
サービスの更新に失敗しました（Failed Service Update）	対処不要です。既存の展開がまだ機能しています。その更新の再試行は実行されません。後の段階で、展開解除がトリガーされた場合は、失敗した更新で作成された VM の部分がクリーンアップされます。
VM 操作が失敗しました（開始、停止、リブート、モニタの有効化、モニタの無効化）（Failed VM Operations (Start, Stop, Reboot, Enable Monitor, Disable Monitor)）	対処不要です。既存の展開がまだ機能しています。NB クライアント（ESC User）は、失敗した操作を再試行できます。
VNF/サービスの操作が失敗しました（開始、停止、リブート、モニタの有効化、モニタの無効化）（Failed VNF/Service Operations (Start, Stop, Reboot, Enable Monitor, Disable Monitor)）	対処不要です。既存の展開がまだ機能しています。NB クライアント（ESC User）は、失敗した操作を再試行できます。
テナント削除要求に失敗しました（Failed delete tenant request）	VIM でリソースのリークが発生する可能性があります。VIM でリソースのリークを解消するには、手動による介入が必要になる場合があります。
ネットワーク要求の削除に失敗しました（Failed delete network request）	VIM でリソースのリークが発生する可能性があります。VIM でリソースのリークを解消するには、手動による介入が必要になる場合があります。

ESC のエラー状態

エラー状態	リカバリ
サブネット削除要求に失敗しました（Failed delete subnet request）	VIM でリソースのリークが発生する可能性があります。VIM でリソースのリークを解消するには、手動による介入が必要になる場合があります。
展開解除要求に失敗しました（Failed undeployment request）	VIM でリソースのリークが発生する可能性があります。VIM でリソースのリークを解消するには、手動による介入が必要になる場合があります。

ESC のエラー状態



付録 B

テクニカルサポートに連絡する前に

追加の支援を受けるために、テクニカルサポート担当者またはCisco TACへの問い合わせが必要になることがあります。この項では、問題の解決にかかる時間を短縮するために、次のレベルのサポートに連絡する前に実行する必要がある手順について概説します。

ESC からのログのダウンロード

トラブルシューティングのために、ESC からログファイルをダウンロードできます。

CLI を使用してログファイルを収集するには、次のコマンドを使用します。

```
sudo escadm log collect --output <output_directory>
```

VM の設定データを収集するには、次のコマンドを使用します。

```
esc_nc_cli get-config esc_datamodel > <file-name>
```

次に例を示します。

```
esc_nc_cli get-config esc_datamodel > /var/tmp/esc_datamodel.txt
```

ESC システムレベルの設定の詳細については、『Cisco Elastic Services Controller User Guide』の「Downloading Logs from the ESC Portal」の項を参照してください。

TAC に問い合わせる前にすべきこと

テクニカルサポート担当者に連絡する前に、次の質問に回答してください。

1. CLI（システムログファイル）およびGUIを使用して、システム情報と設定を収集します。手順については、「ログファイルのダウンロード」を参照してください。
2. ESC でエラーが発生した場合は、エラーのスクリーンショットを取得します。Windows では、Alt+PrintScreen キーを押してアクティブなウィンドウをキャプチャするか、または PrintScreen キーを押してデスクトップ全体をキャプチャします。スクリーンショットを新

テクニカルサポートに連絡する前に

しいMicrosoftのペイント（または同様のプログラム）セッションに貼り付けて、ファイルを保存します。

3. ESCまたはCLIのいずれかからメッセージログに表示される正確なエラーコードをキャプチャします。
4. テクニカルサポート担当者に連絡する前に、次の質問に回答してください。
   - ネットワーク内にあるのはどのESCのバージョン、オペレーティングシステムのバージョン、ストレージ デバイス ファームウェアか。
   - このイベントの発生前または発生時に環境に変更を加えたか（VLAN、アップグレード、またはモジュールの追加）。
   - 同様の設定がされた他のデバイスで、この問題が発生したか。
   - 問題の発生したデバイスの接続先はどこか（どのデバイスまたはインターフェイスか）。
   - この問題が最初に発生したのはいつか。
   - この問題が最後に発生したのはいつか。
   - この問題の発生頻度はどの程度か。
   - 問題発生時にキャプチャした出力のトレースまたはデバッグを行ったか。
   - どのようなトラブルシューティングの手順を試みたか。
5. 問題がソフトウェアアップグレードの試行に関連している場合は、次の質問に回答してください。
   - Cisco ESC の元のバージョンは何か。
   - 新しい Cisco ESC のバージョンは何か。
   - 次のコマンドの出力を収集し、カスタマーサポート担当者に転送します。
     - `esc_nc_cli get-config esc_datamodel > <file-name>`
     - `esc_version`
     - `health.sh`
     - `escadm status`
     - `escadm vim show`