Cisco Elastic Services Controller 5.9 アドミニストレーション ガイド

初版：2022 年 11 月 25 日

最終更新：2022 年 3 月 6 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、Amazon Web Services（AWS）、および VMware NSX-T がサポートされる VIM 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。

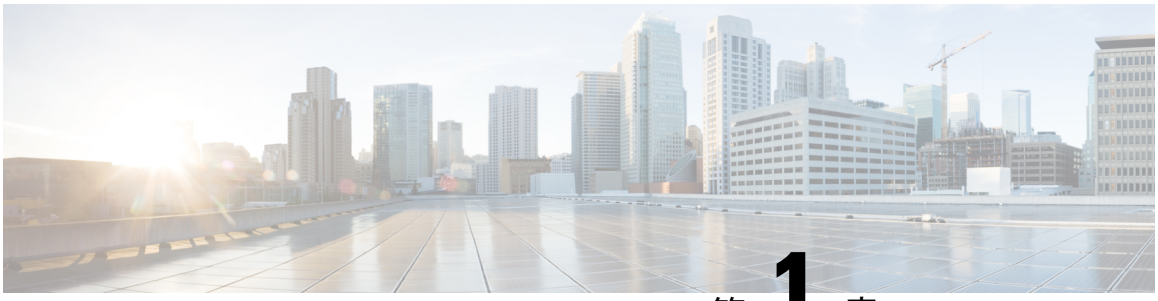
ガイド	このガイドに記載されている情報
Cisco Elastic Services Controller Release Notes	新機能とバグ、既知の問題が記載されています。
Cisco Elastic Services Controller Install and Upgrade Guide	新規インストールとアップグレードのシナリオ、インストール前後のタスク、ESC 高可用性（HA）展開の手順が記載されています。
Cisco Elastic Services Controller User Guide	VNF のライフサイクル管理操作、モニタリング、修復、スケーリングが記載されています。
Cisco Elastic Services Controller ETSI NFV MANO ユーザーガイド	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

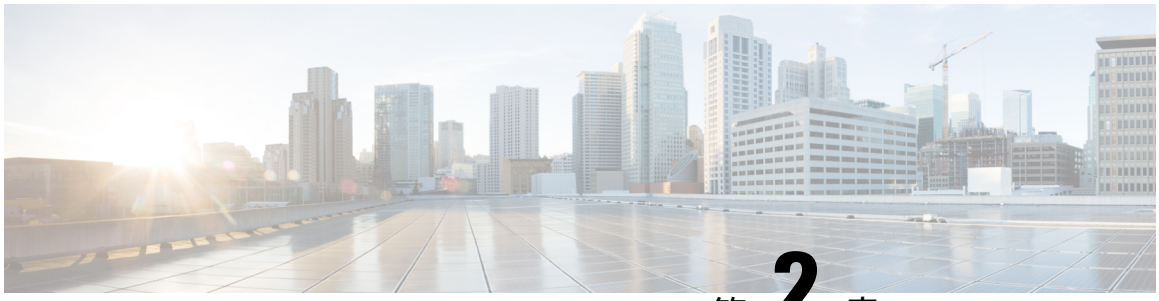
Elastic Services Controller の概要

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、専用仮想ネットワーク機能マネージャ（SVNFM）として、Cisco Mobility VNF と緊密に統合されます。また、ESC は汎用仮想ネットワーク機能マネージャ（GVNFM）としても使用でき、シスコとサードパーティ両方の VNF のライフサイクル管理を提供します。

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 <vim_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、<vim_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 --user <username> --password <password> 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 or
count(../../../../../scaling/static_ip_address_pool) > 0"
{
error-message "Static ip address pools must be configured when static ip addresses are
configured.";
}
}
```

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>custom_vim_name

<host_id>dc380f1721255e2a7ea15932c1a7abc681816642f75276c166b4fe50</host_id>
                    <hostname>my-server</hostname>
```

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

SR-IOV インターフェイスに関する制限事項

- VNF を起動すると、SR-IOV インターフェイスが、ESXi で表示される順序とは逆の順序で表示されることがあります。これにより、特定の VNF のネットワーク接続が失われ、インターフェイス構成エラーが発生します。これは、VMware 6.5 の既知の問題です。
- ESXi 7.0 での複数の SR-IOV インターフェイスを使用したサービスの展開は、ESC 5.7 バージョン以降でサポートされています。

⚠

注意 VNF でSR-IOVネットワークインターフェイスの構成を開始する前に、インターフェイスマッピングを確認します。これにより、ネットワーク インターフェイス構成が VM ホスト上の正しい物理 MAC アドレスインターフェイスに適用されます。

VNF が起動したら、MAC アドレスとインターフェイスのマッピングを確認できます。`show interface` コマンドを使用して、インターフェイスの MAC アドレスなど、インターフェイスの詳細情報を確認します。インターフェイス割り当てが正しいことを確認するには、`show kernel ifconfig` コマンドの結果と MAC アドレスを比較します。

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>
```

SNAT ルーターとフローティング IP の構成

このセクションでは、ルータを作成、更新、および削除する方法について説明します。

概要

送信元のネットワークアドレス変換（SNAT）ルータは OSP 機能です。SNAT ルータは、プライベートネットワークからパブリックネットワークへのトラフィックを許可します。プライベートネットワークで起動された仮想マシンは、SNAT を実行するためにゲートウェイを介してインターネットにアクセスできます。ゲートウェイは、元のパケットの送信元 IP をパブリック側の IP に置き換えます。

ルータの作成

管理状態、SNAT 有効化プロパティを備えた外部ゲートウェイ、内部インターフェイス、スタティックルート、配布など、さまざまな仕様を持つアウトオブバンドとしてルータを作成します。デフォルトでは、SNAT プロパティは有効になっています。

次の 2 つの方法でルータを作成します。

1. 1 つの作成要求で、単純なアウトオブルータを作成します。
2. インターフェイスが追加されたアウトオブルータを作成する：最初にルータ作成の要求を作成し、続いてルータにインターフェイスを接続し、スタティックルートをルータに追加するための更新要求を作成します。

esc_nc_cli スクリプトを使用したルータの作成：

esc_nc_cli スクリプトを使用してルータを作成するには、ルータ名と必要な構成プロパティを含む XML ペイロードが渡されます。

```
<?xml version='1.0' encoding='ASCII'?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
    <routers>
        <router>
           <name>testRouter</name>
           <admin_state>true</admin_state>
           <external_network>internet-net</external_network>
           <snat_enable>true</snat_enable>
           <distribution>false</distribution>
           <description>check for desc</description>
           <interfaces>
               <interface>
                  <subnet>automation_subnet</subnet>
                  <port_id>18b6e6df-fc48-49dc-842e-a1cee546173e</port_id>
               </interface>
    </interfaces>
    <static_routes>
                <route>
                   <route_name>RouteA</route_name>
                   <destination>172.26.0.0/24</destination>
                   <next_hop>10.85.103.93</next_hop>
                 </route>
             </static_routes>
```

```
        </router>
     </routers>
</esc_datamodel>
```

ルータが正常に作成されると、単一の <ok/> 要素で XML ペイロードを受け取ります。ルータが作成されず、アクションが失敗した場合、検証エラーまたは OpenStack API エラーを示すエラーメッセージが表示されます。

ESC REST API を使用したルータの作成：

ルータを作成するには、ESCManager API に HTTP POST 操作を指定します。

```
POST: /ESCManager/v0/<tenant-id>/routers/<internal-router-id>
```

ルータが正常に作成されると、HTTP 200 コードを受け取ります。ルータが作成されず、アクションが失敗した場合は、適切な HTTP エラーコードとエラーメッセージを含む検証エラーまたは OpenStack API エラーを受け取ります。

次に例を示します。

```
{
        "name": "rout0020",
        "admin_state": true,
        "route": [{
                        "route_name": "RouteA",
                        "destination": "172.26.0.0/24",
                        "next_hop": "10.85.103.93"
                }
        ],
        "interface":[{
                        "subnet":"automation_subnet"
                }
        ],
        "external_network": "internet-net"
}

[admin@localhost]$ curl --user admin:P@55w0rd! -k -X POST -d @rest.json -H 'Content-Type:
 application/json' -H 'callback: https://localhost:9009' -H 'Callback-ESC- Events:
https://localhost:9009'
https://localhost:8443/ESCManager/v0/SystemAdminTenantId/routers/testRouterId
```

（注） 内部インターフェイスを追加するには、次を実行します。

1. <subnet> (or) <subnet> & <port_id> を指定できます。
2. *n* 個の有効なインターフェイスを追加できます。

次に例を示します。

```
<interface>
   <subnet>testsubnet</subnet>
   <port_id>portuuidid</port_id>
</interface>
```

スタティックルートを追加するには、次を指定します。

1. route_name：ルート間の一意性を維持します（一意である必要があり、この名前は ESC レベルで維持されます)

2. 接続先：必要に応じて
3. next_hop：必要に応じて

（注） **esc_nc_cli と ESC REST API の両方について、次の点に注意してください。**

1. ルータ名の長さは 255 文字以下にする必要があります。
2. 存在しない外部ネットワーク、サブネット、ペイロード内のポートを使用するとエラーがスローされます。
3. スタティックルートの追加は、対応する `next_hop` に内部/外部インターフェイスが追加されている場合にのみ許可されます。
4. XML ファイル：指定されている場合、有効な XML ドキュメントが含まれている必要があります（esc_nc_cli のみ）。

ルータの更新

ルータが作成されたら、有効な構成でルータを更新します。

ルータの更新中は、次の操作がサポートされています。

- 外部ゲートウェイの追加/クリア
- インターフェイスのアタッチ/デタッチ
- スタティックルートの追加/削除
- 管理ステータスのアップ/ダウン

esc_nc_cli スクリプトを使用したルータの更新

外部ゲートウェイ、インターフェイス、スタティックルートを追加して、ルータ構成を更新できます。これらのアクションを実行するには、有効なタグをペイロードに追加します。ルータへの外部ゲートウェイを1つだけ構成します。一方、ルータには任意の数のインターフェイスとスタティックルートを追加できます。

外部ゲートウェイ、インターフェイス、スタティックルートをクリアまたは削除するには、操作＝削除タグを追加します。例：

```
<external_network operation="delete">internet-net</external_network>
```

`<interface> & <route>` の親に削除操作を付与することで、すべてのインターフェイス/ルートを一度に消去または削除できます。次に例を示します。

```
<interfaces operation="delete">
    <interface>
        <subnet>automation_subnet</subnet>
        <port_id>18b6e6df-fc48-49dc-842e-a1cee546173e</port_id>
```

```
   </interface>
</interfaces>
```

管理状態を UP または DOWN として更新するには、<admin_state> のブール値を変更します。

ルータの更新が成功すると、単一の <ok/> 要素で XML ペイロードを受け取ります。ただし、アクションが失敗した場合は、検証エラーまたは OpenStack API エラーは適切なエラーメッセージとともに表示されます。

ESC REST API を使用したルータの更新：

外部ゲートウェイ、インターフェイス、スタティックルートを追加してルータを更新できます。これらのアクションを実行するには、有効な JSON 値をペイロードに追加する必要があります。ルータへの外部ゲートウェイを 1 つだけ構成します。一方、ルータには任意の数のインターフェイスとスタティックルートを追加できます。

外部ゲートウェイ、インターフェイス、スタティックルートをクリアまたは削除するには、ペイロードからそれぞれの JSON データを削除します。

管理状態の UP または DOWN を更新するには、admin_state のブール値を変更します。ルータを更新するには、ESCManager API に HTTP PUT 操作を指定できます。

```
PUT: /ESCManager/v0/<tenant-id>/routers/<internal-router-id>
```

ペイロードには、前述のように、既存のルータ名と更新プロパティが含まれている必要があります。

成功すると、HTTP 200 コードを受け取ります。ただし、アクションが失敗したままの場合は、検証エラーまたは OpenStack API エラーは、適切な HTTP エラーコードとエラーメッセージとともに表示されます。

以下は、ルータを更新するための API 呼び出しの例です。

```
[admin@localhost]$ curl --user admin:P@55w0rd! -k -X PUT -d @rest.json -H 'Content-Type:
 application/json' -H 'callback: https://localhost:9009' -H 'Callback-ESC-Events:
https://localhost:9009'
https://localhost:8443/ESCManager/v0/SystemAdminTenantId/routers/testRouterId
```

ルータの削除

esc_nc_cli と REST API インターフェイスの両方を使用してルータを削除します。削除できるのは、現在の ESC VM で管理されているルータのみです。一度に削除できるルータは 1 つだけです。

esc_nc_cli スクリプトを使用したルータの削除：

esc_nc_cli でルータを削除するには、esc_nc_cli コマンドでルータ名を渡します。次のコマンドを使用して、ルータを削除します。

```
esc_nc_cli delete-router <router-name>
```

ルータが削除され、アクションが成功すると、単一の <ok/> 要素で XML ペイロードを受け取ります。ただし、アクションが失敗した場合は、検証エラーまたは OpenStack API エラーは適切なエラーメッセージとともに表示されます。

ESC REST API を使用したルータの削除：

ルータを削除するには、ESCManager API で HTTP DELETE 操作を使用します。

次に例を示します。

```
DELETE: /ESCManager/v0/<tenant-id>/routers/<internal-router-id>
```

ペイロードには、既存のルータ名と更新プロパティが含まれている必要があります。

ルータの削除アクションが成功すると、HTTP 200 コードを受け取ります。ただし、アクションが失敗したままの場合は、検証エラーまたは OpenStack API エラーは、適切な HTTP エラーコードとエラーメッセージとともに表示されます。

以下は、ルータを削除するための API 呼び出しの例です。

```
[admin@localhost]$  curl --user admin:P@55w0rd! -k -X DELETE -H 'callback:
https://localhost:9009' -H 'Callback-ESC-Events: https://localhost:9009'
https://localhost:8443/ESCManager/v0/SystemAdminTenantId/routers/testRouterId
```

（注） すべてのスタティックルートが削除されるまで、ルータを削除することはできません。

Notifications:

ルータの操作中に、NETCONF 通知と ESC REST コールバックメッセージの両方を受信します。

表 2:

通知（**NETCONF** または **ESC** コールバック）	通知が送信された場合
CREATE_ROUTER	OpenStack が完了すると、ルータは要求操作を作成します。これが成功したかエラーが発生したか、いずれかの結果が表示されます。
UPDATE_ROUTER	OpenStack が完了すると、ルータは要求操作を更新します。これが成功したかエラーが発生したか、いずれかの結果が表示されます。
ATTACH_INTERFACE	OpenStack により、ルータがインターフェイスに接続され、成功したかエラーが発生したか、いずれかの結果が表示されます。
DETACH_INTERFACE	OpenStack によりルータからインターフェイスが切り離され、成功したかエラーが発生したか、いずかの結果が表示されます。
ADD_STATIC_ROUTE	OpenStack によりルータにスタティックルートが追加され、成功したかエラーが発生したか、いずかの結果が表示されます。

通知（**NETCONF** または **ESC** コールバック）	通知が送信された場合
DELETE_STATIC_ROUTE	OpenStack によりルータからスタティックルートが削除され、成功したかエラーが発生したか、いずかの結果が表示されます。
DELETE_ROUTER	OpenStack が完了すると、ルータは要求操作を削除します。これが成功したかエラーが発生したか、いずれかの結果が表示されます。

次の例は、CREATE_ROUTER が成功した NETCONF 通知を示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2021-09-25T10:31:26.76+00:00</eventTime>
    <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Router successfully created.</status_message>
    <router>testRouter-1</router>
    <tenant>admin</tenant>
    <event>
        <type>CREATE_ROUTER</type>
    </event>
    </escEvent>
</notification>
```

次の例は、ATTACH_INTERFACE が成功した NETCONF 通知を示しています（他の通知も同様です）。

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2021-09-25T10:31:28.891+00:00</eventTime>
    <escEvent xmlns="http://www.cisco.com/esc/esc">
         <status>SUCCESS</status>
         <status_code>200</status_code>
         <status_message>Interface successfully attached.</status_message>
         <router>testRouter-1</router>
         <router_interface>mgmt-net-subnet</router_interface>
              <event>
                   <type>ATTACH_INTERFACE</type>
               </event>
    </escEvent>
</notification>
```

エラーが発生した場合、NETCONF 通知と ESC REST コールバックメッセージは引き続き生成されますが、次のようになります。

`<status>` 値はエラーです。

`<status_code>` は *500* で、

`<status_message>` は、内部で生成されるか、OpenStack から送り返される適切なメッセージです。

フローティング IP を VM に関連付ける

フローティング IP をブール値の true または false として割り当てます。値が true に設定されている場合、OpenStack からのフリーフローティング IP がインターフェイスに関連付けられます。インターフェイスの入力として特定のフローティング IP を割り当てることができます。インターフェイスからフローティング IP の関連付けを解除するには、フローティング IP を false として指定します。

ESC は次のアクションを実行します。

- ESC は、OpenStack から使用可能なフローティング IP のリストを収集し、展開中にフリーフローティング IP を VM に関連付けます。
- 展開の削除中に、同じフローティング IP の関連付けを解除します。
- リカバリ時には、同じフローティング IP を展開に関連付ける必要があります。
- フローティング IP が false として指定されている場合、フローティング IP はインターフェイスから切り離されます。

VM グループレベルのフローティング IP

フローティング IP が VM グループレベルで言及されている場合、OpenStack からのフリーフローティング IP は、NIC ID 0 のインターフェイスに関連付けられます。VMGroup のフローティング IP は nic ID 0 に対応するため、フローティング IP の値を VM グループとインターフェイスレベルの両方で同時に指定することはできません。指定すると、エラーメッセージが表示されます。フローティング IP の値は、列 `float_ip` の下のインターフェイステーブルで更新されます。

以下は、VM グループレベルでフローティング IP を割り当てる例です。

```
<vm_group>
 <name>cirros1</name>
 <bootup_time>60</bootup_time>
 <recovery_wait_time>0</recovery_wait_time>
 <image>Automation-Cirros-Image</image>
 <flavor>medium2</flavor>
 <floating_ip>true</floating_ip>
 ....
 ....
</vm_group>
```

インターフェイスレベルでのフローティング IP：

フローティング IP がインターフェイスレベルで言及されている場合、フローティング IP は対応するインターフェイスに関連付けられています。フローティング IP の値は、列 `float_ip` の下のインターフェイステーブルで更新されます。

次に、インターフェイスレベルでフローティング IP を割り当てる例を示します。

```
<interface>
<nicid>1</nicid>
<floating_ip>10.85.103.99</floating_ip>
<network>esc-net</network>
</interface>
```

デュアルインターフェイスでのフローティング IP：

ポートにデュアルインターフェイスがある場合、フローティング IP を特定のインターフェイスに指定できます。フローティング IP の値は、列 `float_ip` の下の interface_addresses テーブルで更新されます。

次に、デュアル インターフェイス レベルでフローティング IP を割り当てる例を示します。

```
</interface>
<interface>
<nicid>1</nicid>
<network>udhanasenet</network>
<addresses>
<address>
<address_id>0</address_id>
<floating_ip>true</floating_ip>
<subnet>udh-sub</subnet>
</address>
<address>
<address_id>1</address_id>
<floating_ip>10.85.103.95</floating_ip
<subnet>udh-sub</subnet>
</address>
</addresses>
</interface>
```

インターフェイスからのフローティング IP の関連付けの解除：

フローティング IP 値を *false* として割り当て、フローティング IP をインターフェイスから分離します。

```
<floating_ip>false</floating_ip>
```

エラーのシナリオ：

次のタスクの実行中にエラーメッセージが表示されます。

1. VM グループレベルと nic id 0 のインターフェイスレベルの両方でフローティング IP を割り当てる場合。
2. OpenStack の IPV6 アドレスにフローティング IP を割り当てようとした場合。
3. 特定のテナントの OpenStack で利用可能なフリーフローティング IP がないときに、フローティング IP を割り当てようとした場合。
4. スケーリング中は、動的に割り当てられたフローティング IP のみがサポートされます。固定のフローティング IP アドレスを指定することはできません。特定のフローティング IP を指定しようとすると、エラーメッセージが表示されます。

ハードウェア アクセラレーション サポート

フレーバーデータモデルを使用して、VIM でハードウェア 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 VMware vSphere 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 VMware vCenter 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 example.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 --user <username> --password <password>
edit-config ./example.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。特定の vm_group またはネットワークに対して選択する PCI デバイスを指定するには、*pci_id* 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>
      <interface>
                 <nicid>3</nicid>
           <type>passthru</type>
           <network>MgtNetwork</network>
```

```
        <pci_id>0000:07:10.3</pci_id>
        </interface>
    </interfaces>
```

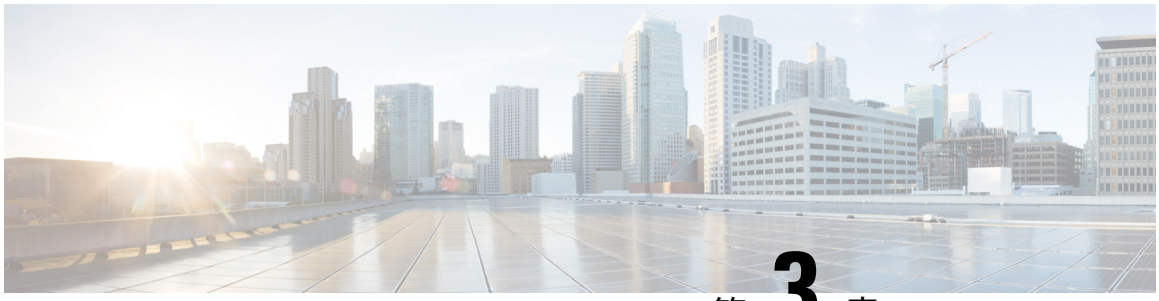
展開が正常に完了すると、***passthru*** 値が通知の interface セクションと運用データ内に設定されます。

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。

- REST API を使用した ESC の正常性のモニタリング （31 ページ）
- SNMP トラップ通知を使用した ESC の正常性のモニタリング （39 ページ）
- ESC での SNMP トラップの管理 （44 ページ）
- 自己署名証明書の管理 （61 ページ）

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 3 : スタンドアロンおよびアクティブ/スタンバイ ハイアベイラビリティにおける ESC ヘルス API のステータスコードとメッセージ （34 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 5 : 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 （36 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>:8060/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。
- ESC ヘルス API ポート番号が 60000 から 8060 に変更されました。

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。

JSON 応答の例：

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<esc_health_report>
<status_code>{error status code}</status_code>
<message>{error message}</message>
</esc_health_report>
```

ローカルアクティブ/アクティブのヘルス API のモニタリングの応答は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8" ?>
<esc_health_report>
    <status_code>2010</status_code>
    <message>ESC service is being provided. ESC AA cluster one or more node(s) not
healthy</message>
    <nodes>
        <node>
            <name>aa-esc-1.novalocal</name>
            <status>HEALTHY</status>
            <datacenter>dc1</datacenter>
            <services>
                <service>
                    <name>escmanager</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>elector</name>
                    <status>leader</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>drbd</name>
                    <status>active</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>pgsql</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                ...
            </services>
        </node>
        <node>
            <name>aa-esc-2.novalocal</name>
            <status>HEALTHY</status>
            <datacenter>dc1</datacenter>
            <services>
                <service>
                    <name>escmanager</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>elector</name>
                    <status>follower</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>drbd</name>
                    <status>standby</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>pgsql</name>
                    <status>stopped</status>
                    <is_expected>True</is_expected>
```

```
                </service>

                ...
            </services>
        </node>
        <node>
            <name>aa-esc-3.novalocal</name>
            <status>NOT_HEALTHY</status>
            <datacenter>dc1</datacenter>
            <services>
                <service>
                    <name>escmanager</name>
                    <status>stopped</status>
                    <is_expected>False</is_expected>
                </service>
                <service>
                    <name>elector</name>
                    <status>follower</status>
                    <is_expected>True</is_expected>
                </service>
                <service>
                    <name>vimmanager</name>
                    <status>running</status>
                    <is_expected>True</is_expected>
                </service>
                ...
            </services>
        </node>
    </nodes>
</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。

表 *3:* スタンドアロンおよびアクティブ/スタンバイ ハイアベイラビリティにおける *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 サービスが提供されています。（ESC services are being provided.）ESC AA クラスタの 1 つまたは複数のノードが正常ではありません。（ESC AA cluster one or more node(s) not healthy.）
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.）
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.）
[5060]	ESC サービス、ETSI が実行されていません。（ESC service, ETSI is not running.）
5070	Vim コネクタ ID [vimId_1,vimId_2,...,vimId_N] がダウンしています。（Vim Connector IDs [vimId_1,vimId_2,...,vimId_N] are down.） または 25 個のうち 6 個の VIM コネクタがダウンしています。（6 of 25 VIM Connectors are down.） （注） 6 つ以上の VIM コネクタ ID がダウンしている場合、VIM ID のリストの代わりにサマリーメッセージが出力されます。

ステータス コード	メッセージ
5080	NFVO サービスは使用できません。（The NFVO service is not available.）
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.）
5091	1 つ以上の ESC サービスが実行されていないため、NFVO サービスを使用できません。(One or more ESC services is not running and the NFVO service is not available.)
5092	VIM コネクタ ID [vim-1] がダウンしています。（VIM Connector ID [vim-1] is down.）NFVO サービスは使用できません。（The NFVO service is NOT available.）

表 *4:* アクティブ/アクティブ ハイアベイラビリティにおける *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（アクティブ/アクティブセットアップ）。（ESC services are running (Active-Active setup).）
2010	ESC サービスが提供されています。（ESC services are provided.）ESC アクティブ/アクティブクラスタの 1 つまたは複数のノードが正常ではありません。（In ESC Active/Active cluster one or more node(s) are not healthy.）
5000	ESC サービスが提供されていません。ESC AA クラスタが正常ではありません（ESC services not being provided, ESC AA cluster not healthy）

（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 *5:* 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（成功）	**Partial Success**（一部成功）	**Failure**（失敗）
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。	なし	• 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。
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 アクティブ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。	• 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 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（成功）	Partial Success（一部成功）	Failure（失敗）
HA（アクティブ/アクティブ）の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。	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。この状態は、ESC サービスは提供されているが、ESC AA クラスタ内の 1 つまたは複数のノードが正常では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。	• ローカルアクティブ/アクティブでは、モニタが 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ない場合です。 アクティブ/アクティブ GEO では、モニタが 7 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ない場合です（Heat テンプレートの設定によって異なります）。 （注） ローカルおよび GEO 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。 • GEO スイッチオーバー期間は、Heat テンプレートの設定によって異なります。デフォルトでは、スイッチオーバーはプライマリデータセンターの障害発生から 5 分後に開始されます。 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 5000 で返されます。 （注） スイッチオーバー中は、新しいリーダーが正常になるまで、ステータスコードは一時的に 5000 で返されます。

ESC ヘルスモニタの機能拡張

ESC ヘルスモニタ API の機能が次のように拡張されています。

- ESC コンポーネントのステータスが判別されます。
- 接続と認証の詳細を簡素化するために、SNMP エージェント用の単一連絡ポイントが提供されます。

ESC モニタコンポーネントにヘルスモニタ API が実装されました。この API を使用して、ダウンした ESC コンポーネントのリストが提供されます。ヘルスモニタは、各 ESC コンポーネントのパブリックおよび内部ヘルス URL を使用して、個々のステータスを判別します。たとえば、VNFM ステータスは、ヘルスモニタが次の URL を実行して判別します。

```
https://localhost:8252/etsi/health
```

URL によって ESC コンポーネントのステータスが判別され、該当するステータスコードとステータスメッセージが SNMP トラップ通知の一部としてを返されます。

VIM 接続ステータス用の ESC ヘルスモニタ API

ESC ヘルスモニタ API が拡張され、新しい ESC ヘルスモニタ API（URL）を使用して VIM コネクタの詳細を照会できるようになりました。

```
http://<escmanager-host>:8088/escmanager/vims
```

ESC スタンドアロン型および HA 設定では、アクティブノードに対して URL が実行されます。ESC アクティブ/アクティブ設定では、すべてのノードに対して URL が実行されます

ヘルスモニタペイロードは、設定されたすべての VIM コネクタのバイナリステータスを判断するために必要な追加情報を返します。VIM コネクタのステータスは、正常またはダウンです。

ESC ヘルスモニタ API は、単一の VIM コネクタが正常かどうかを判断するために、VIM コネクタが定義されている VIM に対してクエリを実行します。クエリの結果に **CONNECTION_SUCCESSFUL** の内部ステータスが含まれる場合、その VIM コネクタは正常です。

クエリに失敗した場合、その VIM コネクタはダウンしています。

さらに、返されるステータスメッセージには、ダウンしている特定の VIM ID のカンマ区切りリストが含まれます。この例は、ESC ヘルスモニタが 2 つのダウンした VIM コネクタのペイロードを返しています。

```
{
"message": "VIM Connector IDs [vim-connector-site-1A, vim-connector-site-1C] are down.",
"status_code": "5070"
}
```

VIM コネクタの SNMP トラップ通知の詳細については、SNMP トラップ通知を使用した ESC の正常性のモニタリング （39 ページ）を参照してください。

ESC ヘルスモニタは、デフォルトでは VIM コネクタのステータスをモニタしません。ESC ヘルスモニタを有効にするには、SNMP トラップ通知 （54 ページ）の「VIM および NFVO モニタリング用の SNMP トラップの有効化」を参照してください。

NFVO 接続ステータス用の ESC ヘルスモニタ API

ESC ヘルスモニタ API は、NFVO への接続状況を判別できます。ESC は、NFVO から ESC への接続状況を照会するための API を備えています。NFVO は、標準の SOL003 定義 API クエリに応答します。URL は次のとおりです。

```
https://<vnfm-host>:8252/etsi/nfvo/health
```

NFVO が正常に認証され、SOL003 定義 API に応答する場合、NFVO は到達可能で正常です。

この例は、NFVO が設定されているが到達不能な場合に ESC ヘルスモニタが返すペイロードを示しています。

```
{
"message": "The NFVO service is NOT available.",
"status_code": "5080"
}
```

ESC ヘルスモニタは、デフォルトでは NFVO の接続ステータスをモニタしません。ESC ヘルスモニタを有効にするには、SNMP トラップ通知 （54 ページ）の「VIM および NFVO モニタリング用の SNMP トラップの有効化」を参照してください。

ETSI 展開の詳細については、*Cisco Elastic Services Controller 5.2 ETSI NFV MANO* ユーザガイドを参照してください。

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 および 3 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。

1. **ESC のインストール時の SNMP エージェントの有効化および設定**：

   - **BootVM によるスタンドアロンまたはアクティブ/スタンバイ HA セットアップ**

     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-snmp-agent
      --ignore-ssl-errors
     --managers "udp:ipv4/port,udp:[ipv6]/port"
     ```

     （注） マネージャの値は、SNMPトラップが「udp:ipv4/port」または「udp:[ipv6]/port」形式で配信される場所のカンマ区切りリストです。IP とポートは実際の値に置き換える必要があります。

   - **アクティブ/アクティブ HA 設定**

     アクティブ/アクティブインストール中に SNMP エージェントを有効にできます。設定パラメータ `ignore_ssl_errors` および `managers` リストを渡して、インストール時にエージェントを設定できます。aa-params.yaml で定義するか、次のコマンドラインで渡すことができます。

     ```
     openstack stack create name-aa --template aa.yaml -e aa-params.yaml \
     --parameter nameprefix=ESC_AA \
     --parameter image_name=ESC-5_2_0_43 \
     --parameter flavor_name=m1.large \
     ...
     --parameter snmp_agent_startup: auto \
     --parameter snmp_agent_ignore_ssl_errors: true \
     --parameter snmp_agent_managers: [ "udp:ipv4/port,udp:[ipv6]/port" ]
     ```

2. **ESCADM による有効化と設定**

   - **スタンドアロンまたはアクティブ/スタンバイ HA 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,udp:[ipv6]/port"
     ```

   - **アクティブ/アクティブ HA 設定**

ESC ノード 1、ノード 2、ノード 4、およびノード 5 のすべてのリーダー対応ノードで、次のコマンドを実行します。

```
sudo escadm snmp set --startup=auto
```

（注） スタック更新によってノードが削除され、再作成された場合は、直前のコマンドを再実行する必要があります。

プライマリデータセンターの SNMP 対応ノード（ノード 1 および 2）でのみ、ESC サービスを再起動します。一度に 1 つのノードです。

```
sudo escadm stop
sudo escadm restart
```

リーダーノードが正常になり、SNMP エージェントが実行されたら、リーダーノードに SNMP エージェント設定を次のように追加できます。

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

（注） `ignore-ssl-errors` パラメータは主に、ESC VM で自己署名証明書が使用される SSL エラーを防止する開発者環境用です。

マネージャの値は、SNMP トラップが「`udp:ipv4/port`」または「`udp:[ipv6]/port`」形式で配信される場所のカンマ区切りリストです。IP とポートは実際の値に置き換える必要があります。

3. 設定ファイルの更新

この設定の更新を有効にするには、SNMP エージェントがすでに有効になっている必要があります。

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。

```
{
"publicCommunities": "public",
"users": [],
"sysDescr": "admin@localhost",
"ignoreSslErrors": "yes",
"logLevel": "INFO",
"sysName": "system name",
"managers": [{
   "privPassword": "enc:95w3hE+uZ1A3vyykaPpKEw==",
   "targetEndpoint": "udp:localhost/12000",
   "privProtocol": "AES128",
   "targetCommunity": "public",
   "label": "some manager",
   "targetProtocol": "v2",
   "authProtocol": "SHA",
   "authPassword": "enc:IYt1UIW8wug3vyykaPpKEw==",
   "authentication": "authpriv",
```

```
      "username": "admin",
      "engineId": "80:00:00:00:01:02:03:04"
   }]
   }
```

構成は、ユーザー定義のコミュニティストリングのファイル */opt/cisco/esc/esc_database/snmp.conf.* にあります。このファイルは JSON 形式です。

（注） この構成は、SNMP バージョン 2c プロトコルに適用されます。

```
{
    "publicCommunities": "test",
    "users": [],
    "sysDescr": "TestSNMPAgentConfiguration SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "dnd-admin-1208",
    "managers": []
}
```

以下を使用して、snmptrapd.conf 構成ファイルを構成します。

```
AuthCommunity log,execute,net test
disableAuthorization yes
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%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
```

出力：

```
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c test -M +/opt/cisco/esc/snmp/mibs localhost:2001
 CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
[admin@dnd-admin-1208 ~]$ snmpwalk -v2c -c test -M +/opt/cisco/esc/snmp/mibs
172.24.0.33:2001 CISCO-ESC-MIB::vnfm
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
CISCO-ESC-MIB::escStatusCode.0 = STRING: "2000"
```

構成は、*/opt/cisco/esc/esc_database/snmp.conf.* とコンマで区切られた複数のコミュニティにあります。

```
{
    "publicCommunities": "public, foo ,bar",
    "users": [],
    "sysDescr": "TestSNMPAgentConfiguration SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "dnd-admin-1208",
    "managers": []
}
```

以下を使用して、snmptrapd.conf 構成ファイルを構成します。

```
AuthCommunity log,execute,net public, foo ,bar
disableAuthorization yes
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%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
```

例：

```
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c foo -M +/opt/cisco/esc/snmp/mibs localhost:2001
 CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c public -M +/opt/cisco/esc/snmp/mibs
localhost:2001 CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
[admin@dnd-admin-1208 ~]$ snmpget -v2c -c bar -M +/opt/cisco/esc/snmp/mibs localhost:2001
 CISCO-ESC-MIB::escStatusMessage.0
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
```

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。
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。

次の表に、SNMP MIB のトラップエントリを示します。エンタープライズ OID は 1.3.6.1.4.1 です。

表 *6 : SNMP MIB* トラップエントリ

ノード	索引	親
cisco	9	エンタープライズ
ciscoMgmt	9	cisco
ciscoEscMIB	844	ciscoMgmt
escNotifs	0	ciscoEscMIB
escMIBObjects	1	ciscoEscMIB
vnfm	1	escMIBObjects
escStatusMessage	1	vnfm
escStatusCode	2	vnfm

ノード	索引	親
escPreviousStatusCode	3	vnfm
escPreviousStatusMessage	4	vnfm

SNMP トラップ通知の有効化

```
sudo escadm snmp start
```

ESCADM ツールを使用して、SNMP サービスを開始します。

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
- ESC での SNMP トラップの管理（SNMP マネージャ）
- SNMP GET/WALK の例
- 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
- アクティブ/アクティブ環境での ESC SNMP エージェントの管理
- ESC での自己署名証明書の管理

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）。
5210	「AA リーダーノードが変更されました」。（"**AA LEADER node change**".）ノードがリーダーになった AA 環境では、ノード上のエージェントがこの通知を送信します。ローカルリーダーの変更のみの場合です。
5200	「HAアクティブノードが変更されました」。（"HA ACTIVE node change"）ノードがアクティブノードになった A/S HA 環境では、エージェントがこの通知を送信します。
5220	「**GEO AA プライマリデータセンターが変更されました**」（"Geo AA Primary datacenter change" ）GEO A/A 環境では、GEO スイッチオーバー後にノードがリーダーになると、ノード上のエージェントがこの通知を送信します。GEO リーダーの変更のみの場合です。

ESC での SNMP トラップの管理（SNMP マネージャ）

SNMP マネージャは別のシステムに展開され、ESC SNMP エージェントに登録されます。たとえば、アシュアレンスシステムは ESC から受信した SNMP トラップの一般的なコンシューマです。

次の例では、*snmptrapd*、*snmpget*、*snmpwalk* などの基本的な UNIX SNMP ツールを使用します。

SNMPv2c の例

次のように SNMP トラップデーモンの構成ファイルを設定します。

```
authCommunity log,execute,net public
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%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
```

これにより、*snmptrapd* は「public」コミュニティストリングを使用して受信した通知を処理できます。端末セッションでデーモンを起動し、次のコマンドを実行します。

```
snmptrapd -f -C -c ./snmptrapd.conf -Le 12000
```

2 番目のセッションを開いて、トラップが受信されているかどうかを確認します。

```
snmptrap -v 2c -c public -n "" localhost:12000 0 linkUp.0
```

セッション 1 では、次のようになります。

```
Agent Address: somehost.somedomain
Agent Hostname: localhost (UDP: [127.0.0.1]:51331->[0.0.0.0]:0)
Enterprise OID: .
Trap Sub-Type: 0
Community/Infosec Context: TRAP2, SNMP v2c, community public
Uptime: 0
PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::snmpTrapOID.0 = OID: IF-MIB::linkUp.0
--------------
```

ESC SNMP エージェントをテストし、「snmp.config」の次のマネージャエントリを使用します。SNMP エージェントによって生成されたトラップも、デーモンによってログに記録されます。シスコおよび ESC MIB が ~/.snmp/mibs に存在することを確認します。

SNMPv2 のマネージャエントリ

```
"managers": [{
    "targetEndpoint": "udp:localhost/12000",
    "targetCommunity": "public",
    "label": "Trap test v2c",
    "targetProtocol": "v2c"
}]
```

SNMPv3 の例

snmptrapd.conf ファイルを次のように更新します。

```
disableAuthorization no
authCommunity log,execute,net public

createUser -e 0x8000000001020304 admin SHA authpassword AES privpassword
authUser log admin

format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%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
```

これにより、*admin* ユーザが追加されます。「-e」は、エンジン ID（5ー32 文字の 16 進数文字列）を示します。すべての SNMP v3 エージェントには、エージェントの一意の識別子として機能するエンジン ID があります。エンジン ID は、メッセージの認証および暗号化用のキーを生成するためのハッシュ関数とともに使用されます。

システムが通信するには、両側で同じ authProtocol（MD5 または SHA）と privProtocol（AES または DES）を使用する必要があります。一部のデバイスでは、これらの組み合わせのすべてはサポートされていません。トラップレシーバが同じように設定されていることを確認するに

は、どのサービスが使用可能になっているかを確認する必要があります。1 つの端末セッションでデーモンを再起動します。

```
snmptrapd -f -C -c ./snmptrapd.conf -Le 12000
```

2 番目のセッションで設定をテストし、ユーザ名、パスワード、エンジン ID などを照合します。*authPriv* セキュリティレベルでは、認証と暗号化の両方が選択されることに注意してください。

```
snmptrap -v 3 -n "" -a SHA -A authpassword -x AES -X privpassword -l authPriv -u admin
-e 0x8000000001020304 localhost:12000 0 linkUp.0
```

これにより、ウィンドウ 1 にトラップのログが記録されます。

出力例：

```
Agent Address: casper.cisco.com
 Agent Hostname: localhost (UDP: [127.0.0.1]:53434->[0.0.0.0]:0)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v3, user admin, context
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::snmpTrapOID.0 = OID: IF-MIB::linkUp.0
```

ESC で上記の設定を使用する際、次の例を参考にしてください。エンジン ID の数値は、トラップデーモンで使用される「0x」形式ではなく、コロンで区切られることに注意してください。

SNMPv3 のマネージャエントリ

```
"managers": [{
   "privPassword": "privpassword",
   "targetEndpoint": "udp:localhost/12000",
   "privProtocol": "AES128",
   "targetCommunity": "public",
   "label": "V3 trap test",
   "targetProtocol": "v3",
   "authProtocol": "SHA",
   "authPassword": "authpassword",
   "authentication": "authpriv",
   "username": "admin",
   "engineId": "80:00:00:00:01:02:03:04"
}],
```

v3 メッセージの ESC 出力例

```
Agent Address: casper.cisco.com
 Agent Hostname: localhost (UDP: [127.0.0.1]:52103->[0.0.0.0]:0)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v3, user admin, context 80:00:00:00:01:02:03:04
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (27252277) 3 days, 3:42:02.77
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.844.0.1
SNMPv2-MIB::sysDescr.0 = STRING: SNMP Agent
SNMPv2-SMI::enterprises.9.9.844.1.1.2.0 = STRING: "2000"
SNMPv2-SMI::enterprises.9.9.844.1.1.1.0 = STRING: "ESC services are running."
 --------------
```

トラップ出力

通常、トラップには statusCode、statusMessage、previousStatusCode、previousStatusMessage の 4 つのエントリが含まれます。

```
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."
CISCO-ESC-MIB::escPreviousStatusCode.0 = STRING: "5102"
CISCO-ESC-MIB::escPreviousStatusMessage.0 = STRING: "Warning: Could not connect to ESC
Monitor. See log for details."
```

ESC SNMP エージェントは、以前のステータスやステータスコードメッセージとともに SNMP トラップを送信します。これにより、クライアントは最新の SNMP トラップの応答先を判断できます。

以前のステータスコードやメッセージがない場合、これらの文字列は空になります。たとえば、SNMPエージェントは、以前のステータスコードとステータスメッセージの値を MIB 文字列として返します。

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "2000"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."
CISCO-ESC-MIB::escPreviousStatusCode.0 = STRING: "5090"
CISCO-ESC-MIB::escPreviousStatusMessage.0 = STRING: "More than one ESC service (confd,
etsi) not running."
```

これにより、SNMP クライアントは、すべてのサービスが実行中であること、およびこの SNMP トラップが応答している Confd と ETSI サービスは、以前に実行されておらず、現在実行中であることを認識できます。

SNMP マネージャオプション

表 *7: SNMP* マネージャオプション

キー	プロトコル	説明
targetCommunity	v2c	トラップを送信するコミュニティ。デフォルトは *public* です。
label	v2c/v3	このマネージャの名前。
targetEndpoint	v2c/v3	トラップの送信先のアドレスとポート。例：*udp:localhost/12000*。
targetProtocol	v2c/v3	このマネージャに使用する SNMP プロトコル。v2c または v3。デフォルトは v2c です。

キー	プロトコル	説明
authPassword	v3	ユーザのパスワード。プレーンテキストのパスワードを入力すると、エージェントはそのパスワードを検出して暗号化します。
authProtocol	v3	使用する認証プロトコル。SHA または MD5。
認証	v3	認証のタイプは、AuthPriv、AuthNoPriv、または NoAuthNoPriv のいずれかです。
engineId	v3	このトラップに使用するエンジン ID（16 進数）。エンジン ID は、マネージャが使用する ID と一致させる必要があります。例：80:00:00:00:01:02:03:04
privPassword	v3	暗号化（プライバシー）パスワード。プレーンテキストのパスワードを入力します。エージェントはパスワードを検出して暗号化します。
privProtocol	v3	暗号化プロトコルは、DES、AES、AES128、AES192、または AES256 のいずれかになります。
ユーザ名	v3	認証用のユーザ名（またはセキュリティ名）。

SNMP GET/WALK の例

この項では、SNMP ツールの *snmpwalk* および *snmpget* を使用した SNMP *get* の実行方法の例を示します。

（注） この例では、ESC MIB が SNMP MIB パスに追加されていることを前提としています。

SNMP GET - コマンドラインの例

表 *8*:

SNMP の例	コマンド	出力例
SNMPv2c の例	`snmpget -v2c -c public localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`	**出力例** `CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."`

SNMPv3 NoAuthNoPriv の例	次のユーザが ESC SNMP エージェントの設定に追加されます。
	V3 SNMP のユーザエントリ
	`"users": [{ "username": "admin" }],`
	コマンド
	`snmpget -v3 -l authpriv -u admin localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`
		出力例
		`CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."`

SNMPv3 AuthNoPriv の例	次のユーザが ESC SNMP エージェントの設定に追加されます。
	V3 SNMP のユーザエントリ
	`"users": [{ "username": "admin", "authProtocol": "SHA", "authPassword": "authpassword" }],`
	コマンド
	`snmpget -v3 -l authpriv -u admin -a "SHA" -A "authpassword" localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`
		出力例
		`CISCO-ESC-MIB::escStatusMessage.0 = STRING: "ESC services are running."`

SNMP の例	コマンド	出力例

SNMPv3 AuthPriv の例	次のユーザが ESC SNMP エージェントの設定に追加されます。
	`"users": [{ "username": "admin", "authProtocol": "SHA", "authPassword": "authpassword", "privProtocol": "AES128", "privPassword": "privpassword" }],`
	コマンド
	`snmpget -v3 -l authpriv -u admin -a "SHA" -A "authpassword" -x "AES128" -X "privpassword" localhost:2001 CISCO-ESC-MIB::escStatusMessage.0`
		出力例
		`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、構成ファイルに変更がないかを監視し、変更が行われるとリロードします。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 ノードが HA 設定で展開されます。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。

- SNMPを有効にするには、アクティブノードとスタンバイノードの両方を設定する必要があります。
- 1 つの ESC ノード（アクティブノード）のみが SNMP トラップを送信できます。
- スイッチオーバーが発生すると、スタンバイノードのSNMPエージェントは自動的にアクティブノードの設定を受信します。
- フェールオーバーが原因でスタンバイノードがアクティブノードになると、トラップが生成されます。

AA 環境での ESC SNMP エージェントの管理

SNMP エージェントサービスは、ローカルまたは GEO ESC アクティブ/アクティブ設定でもサポートされます。アクティブ/アクティブ展開における考慮事項は次のとおりです。

- SNMP エージェントはリーダーノードでのみトラップを実行し、送信します。

- トラップは次のシナリオで送信されます。
  - ESC ヘルス API のステータスコードの変更時。SNMP エージェントは、AA のヘルスモニタ API をポーリングします。返されたステータスコードに変更がある場合は、トラップとしてサブスクライバに送信されます。
  - ローカルスイッチオーバーを示す新しいリーダーになるノードによる、ローカルスイッチオーバーの後。
  - 新しい GEO プライマリデータセンターのリーダーになるノードによる、GEO スイッチオーバーの後。
- リーダーノードの設定に対する変更は、スイッチオーバー後に新しいリーダーによって引き継がれます。

専用の対象コミュニティによる複数のマネージャの管理

SNMP マネージャは別のシステムに展開され、ESC SNMP エージェントに登録されます。

ESC SNMP エージェントは、SNMP トラップを受信する複数のマネージャ構成の配列をサポートし、各構成には専用の対象コミュニティがあります。

次の例は、専用の対象コミュニティによる複数のマネージャのサポートを示しています。

ウィンドウ 1 を開く

SNMP エージェント構成ファイル */opt/cisco/esc/esc_database/snmp.conf* を開き、次の情報を追加します。

```
{
    "publicCommunities": "public",
    "users": [],
    "sysDescr": "TestSNMPAgentConfiguration SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "dnd-admin-1208",
    "managers": [
        {
            "targetEndpoint": "udp:localhost/12006",
            "targetCommunity": "test1",
            "label": "Trap test v2c",
            "targetProtocol": "v2c"
        },
        {
            "targetEndpoint": "udp:localhost/12004",
            "targetCommunity": "test2",
            "label": "Trap test v2c",
            "targetProtocol": "v2c"
        }
    ]
}
```

ウィンドウ 2 を開く

以下を使用して、SNMP トラップデーモン構成ファイルを構成します。

```
AuthCommunity log,execute,net test1
```

```
disableAuthorization yes

format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%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
```

ウィンドウ 3 を開く

以下を使用して、SNMP トラップデーモン構成ファイルを構成します。

```
AuthCommunity log,execute,net test2
disableAuthorization yes
format2 %V\n% Agent Address: %A \n Agent Hostname: %B (%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
```

escadm ツールを使用して、ESC VM の confd サービスを停止します。

udo escadm confd stop

escadm は、snmptrapd プロセスが「test1」コミュニティストリングを使用して通知を受信できるようにします。ウィンドウ 2 でデーモンを開始して、次のコマンドを実行します。

```
snmptrapd -f -C -c ./snmptrapdm.conf -Le 12006
```

前のコマンドを実行すると、ウィンドウ 2 に次のように表示されます。

```
Agent Address: 0.0.0.0
 Agent Hostname: dnd-admin-1208.novalocal (UDP: [127.0.0.1]:57472->[127.0.0.1]:12006)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v2c, community test1
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1043610566) 120 days, 18:55:05.66
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.844.0.1
SNMPv2-MIB::sysDescr.0 = STRING: TestSNMPAgentConfiguration SNMP Agent
SNMPv2-SMI::enterprises.9.9.844.1.1.2.0 = STRING: "5020"
SNMPv2-SMI::enterprises.9.9.844.1.1.1.0 = STRING: "ESC service ESC_CONFD not running."
SNMPv2-SMI::enterprises.9.9.844.1.1.3.0 = STRING: "2000"
SNMPv2-SMI::enterprises.9.9.844.1.1.4.0 = STRING: "ESC services are running."
```

escadm は、snmptrapd プロセスが「test2」コミュニティストリングを使用して通知を受信できるようにします。ウィンドウ 3 でデーモンを開始して、次のコマンドを実行します。

```
snmptrapd -f -C -c ./snmptrapdm.conf -Le 12004
```

前のコマンドを実行すると、ウィンドウ 3 に次のように表示されます。

```
Agent Address: 0.0.0.0
 Agent Hostname: dnd-a-1208.novalocal (UDP: [127.0.0.1]:45804->[127.0.0.1]:12004)
 Enterprise OID: .
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v2c, community test2
 Uptime: 0
 PDU Attribute/Value Pair Array:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1043610566) 120 days, 18:55:05.66
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.9.9.844.0.1
SNMPv2-MIB::sysDescr.0 = STRING: TestSNMPAgentConfiguration SNMP Agent
SNMPv2-SMI::enterprises.9.9.844.1.1.2.0 = STRING: "5020"
SNMPv2-SMI::enterprises.9.9.844.1.1.1.0 = STRING: "ESC service ESC_CONFD not running."
SNMPv2-SMI::enterprises.9.9.844.1.1.3.0 = STRING: "2000"
SNMPv2-SMI::enterprises.9.9.844.1.1.4.0 = STRING: "ESC services are running."
```

SNMP トラップ通知

VIM および NFVO モニタリング用の SNMP トラップの有効化

SNMP エージェントは ESC ヘルスモニタ API を使用して、ESC コンポーネント、VIM コネクタ、および NFVO 接続のステータスを照会します。デフォルトでは、ESC ヘルスモニタでは VIM 接続と NFVO 接続はモニタされません。また、VIM 接続と NFVO 接続に関する SNMP トラップは生成されません。

VIM および NFVO 接続ステータスの変更トラップを有効にするには、ESC ヘルスモニタの構成ファイル（`/opt/cisco/esc/esc-config/esc-config.yaml`）内に次のパラメータがあることを確認します。

```
monitor:
(2)report:
(4)nfvo:
(6)enabled: true
(4)vim_connectors:
(6)enabled: true
(6)name_threshold: 5
```

上記のパラメータが構成ファイルで指定されていない場合、VIM および NFVO 接続コンポーネントのモニタリングはデフォルトで false になります。vim_conectors および name_threshold は、一般的なメッセージの前にステータスに出力される VIM コネクタ ID の数を示します。メッセージには、ダウンしている VIM コネクタの数が示されますが、「25 個の VM コネクタの内、6 つがダウンしています。」というような詳細は示されません。

ステータスメッセージについては、「VIM コネクタの SNMP トラップ通知」を参照してください。

NFVO 接続の SNMP トラップ通知

SNMP トラップが送信されるのは、NFVO の詳細が ETSI VNFM サービス内で設定されており、ESC ヘルスモニタの設定で NFVO のモニタリングが有効になっているにもかかわらず、NFVO に到達できない場合です。

ETSI VNFM サービスは、NFVO が応答する標準 SOL003 API を使用して NFVO 接続をテストします。

NFVO に到達できない場合は、次の SNMP トラップが生成されます。

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "5080"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "The NFVO service is NOT available."
```

（注）
- NFVO に到達できるが、ログイン情報が正しくない場合、ステータスは利用不能になります。
- NFVO 接続のステータスは、ESC モニタヘルス API が実行された場合にのみ報告されます。NFVO の可用性は定期的にモニタされません。

VIM コネクタの SNMP トラップ通知

SNMP トラップが送信されるのは、VIM コネクタが ESC 内で設定されており、ESC ヘルスモニタの設定で VIM のモニタリングが 有効になっているにもかかわらず、設定されている VIM コネクタのいずれにも到達できない場合です。到達不能な VIM コネクタは、**CONNECTION_SUCCESSFUL** に相当しない内部 ESC ステータスを持つコネクタです。

- 1 つの VIM コネクタが使用できない場合は、次のトラップが生成されます。

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "5070"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "VIM Connector ID [vim-id1] is down."
```

- 2 つ以上の VIM コネクタが使用できない場合は、次のトラップが生成されます。

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "5070"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "VIM Connector IDs [vim-id1, vim-id2,
vim-id3] are down."
```

（注） VIM コネクタ数のデフォルト値は 5 です。デフォルト値は、esc-config.yaml ファイルで設定できます。「VIM および NFVO モニタリング用の SNMP トラップの有効化」を参照してください。

- 使用できない VIM コネクタ数が name_threshold を超えると、次のトラップが生成されます。

```
CISCO-ESC-MIB::escStatusCode.0 = STRING: "5070"
CISCO-ESC-MIB::escStatusMessage.0 = STRING: "6 of 25 VIM Connectors are down."
```

ESC ヘルスモニタ API 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（31 ページ）を参照してください。

結合および分割 SNMP トラップモード

SNMP エージェントは、結合または分割トラップを返すように設定されています。

- **結合トラップ**：現在、SNMP エージェントでは結合トラップが生成されます。出力が複数の ESC コンポーネントやイベントを示している場合でも、ESC ヘルスモニタからの出力が考慮されて、単一の完全なトラップとして送信されます。この出力は、SNMP エージェントの最後のポーリング期間に生成されます。複数の ESC サービスがダウンした場合も単一のトラップとして送信されます。
- **分割トラップ**：ESC リリース 5.4 以降では、各 ESC サービスやコンポーネントの**稼働**または**ダウン**イベントごとに 1 つのトラップがサポートされます。それぞれの**稼働**または**ダウン**イベントには、固有のステータスメッセージとステータスコードが割り当てられます。

（注） モニタ対象の *ESC* サービスは、既存の ESC コンポーネント（MONA、confd、ETSI、ESCMANAGER、VIMMANAGER）の正常性ステータスです。VIM コネクタの有効性と NFVO 接続は、VIM マネージャコンポーネントの一部になります（VIMMANAGER の一部としてモニタされます）。

VIM コネクタの有効性と NFVO 接続のモニタリングはともにデフォルトで無効になっています。有効にすると、ESCヘルスモニタは接続ステータスをそれぞれ自動的に報告します。SNMP エージェントはこの結果に基づいて、既存の ESC サービスと一緒にトラップを送信します。

稼働または**ダウン**イベントごとの個々のトラップが出力されると（分割トラップ）、複数の ESC サービスに対してイベントが発生したことを示すステータスコードとトラップが削除されます。そのため、次の ESC ヘルスモニタ情報は、分割モードでは SNMP トラップコードとして表示されません。ESC コンポーネント情報を結合するトラップは削除されます。

設定

結合トラップモードや分割トラップモードは *trapMode* と呼ばれる新しいプロパティによって制御されます。このプロパティは、次に示すように `/opt/cisco/esc/esc_database/snmp.conf` ファイルで設定できます。

```
{
    "publicCommunities": "public",
    "users": [],
    "sysDescr": "TestSNMPAgentTraps SNMP Agent",
    "ignoreSslErrors": "yes",
    "logLevel": "INFO",
    "sysName": "test-5-4-0-51-keep",
    "trapMode": "combined",
    "managers": []
}
```

このファイルを自動生成する場合のデフォルト値は *combined* です。これは、構成ファイルで *trapMode* が指定されていない場合もデフォルト値になります。これにより、アップグレード時の下位互換性が確保されます。

SNMP ESC コンポーネントのステータスコード

稼働 イベントトラップのステータスコード（MONA がダウンしていたが、現在は稼働状態に戻っている場合）は、新規になります。単一の ESC サービスが復元中であることを示すトラップが以前に生成されていないためです。すべての ESC サービスに対して SNMP エージェントが送信するコードのリストを以下に示します。

表 *9 : SNMP ESC* コンポーネントのステータスコード

ESC コンポーネント	稼働コード	ダウンコード	稼働コードメッセージ	ダウンコードメッセージ
すべてのサービスが稼働	2000		ESC サービスが実行されています。（ESC services are running.）	
ESC_MANAGER	2010	5010	ESC サービス、ESC_MANAGER が実行されています。（ESC service ESC_MANAGER running.）	ESC サービス、ESC_MANAGER が実行されていません。（ESC service ESC_MANAGER not running.）
ESC_CONFD	2020	5020	ESC サービス、ESC_CONFD が実行されています。（ESC service ESC_CONFD running.）	ESC サービス、ESC_CONFD が実行されていません。（ESC service ESC_CONFD not running.）
MONA	2030	5030	ESC サービス、MONA が実行されています。（ESC service MONA running.）	ESC サービス、MONA が実行されていません。（ESC service MONA not running.）
VIM_MANAGER	2040	5040	ESC サービス、VIM_MANAGER が実行されています。（ESC service VIM_MANAGER running.）	ESC サービス、VIM_MANAGER が実行されていません。（ESC service VIM_MANAGER not running.）
ETSI	2060	[5060]	ESC サービス、ETSI が実行されています。（ESC service ETSI running.）	ESC サービス、ETSI が実行されていません。（ESC service ETSI not running.）
接続サービス				

ESC コンポーネント	稼働コード	ダウンコード	稼働コードメッセージ	ダウンコードメッセージ
VIM コネクタ	2070	5070	Vim コネクタ（ID [vimid_1] ）が稼働しています。（Vim Connector ID [vimid_1] is up）	Vim コネクタ（ID [vimid_1] ）がダウンしています。（Vim Connector ID [vimid_1] is down.）
NFVO	2080	5080	NFVO サービスを使用できます。（The NFVO service is available.）	NFVO サービスは使用できません。（The NFVO service is NOT available.）

高可用性

ESC が高可用性ペアで動作している場合は、上記のステータスコードとメッセージが引き続き適用されますが、追加で適用できるステータスコードが 1 つあります。

表 *10*：

ESC コンポーネント	コード	メッセージ
すべてのサービスが稼働 - ESC HA ノード	2010	ESC サービスが実行されています。（ESC services are running.）ESC 高可用性ノードに到達できません。（ESC High-Availability node not reachable.）

この状況が発生すると、2010 の SNMP トラップが上記の詳細とともに送信されます。高可用性に相当する 5010 はありません。状況が解決されると、*2000 - ESC* サービスが実行されています。（*ESC services are running.*）のメッセージが送信されます。2010 ステータスコードの**稼働**トラップは送信されません。

アクティブ/アクティブ

分割モードのトラップは、アクティブ/アクティブ環境（GEO A/Aを含む）の結合モードのトラップと同じです。SNMP エージェントは、A/A の高レベルステータスコードを ESC コンポーネントごとに分割しません。

SNMP エージェントの内部トラップ

SNMP エージェントトラップは、エラーの状態に対しても送信されます。SNMP エージェントトラップは通常、内部接続エラーを示します。次の SNMP エージェントトラップは、受信時および状況が解決したときに送信されます。

表 *11 : SNMP* エージェントの内部トラップ

条件	停止/稼働コード	メッセージ
ESC ヘルスモニタ - HTTPエラー	2100/5100	ESC モニタ API の使用中に HTTP エラーを受信しました（A HTTP error was received when using the ESC Monitor API）（*HTTP* エラーについてはメッセージ内に示されます）
ESC ヘルスモニタ - 不明な応答	2101/5101	ESC モニタは応答しましたが、データを認識できませんでした（The ESC Monitor replied, but the data could not be understood）（データはメッセージ内に示されます）
ESC ヘルスモニタ - ヘルスモニタがダウン	2102/5102	ESC モニタに接続できませんでした。（Could not connect to ESC Monitor.）
ESC ヘルスモニタ - 不明なエラー	2199/5199	未処理のエラーが発生しました（An unhandled error occurred）（詳細はメッセージに示されます）
HA ノードの変更	5200	HA アクティブノードが変更されました（HA ACTIVE node change）（5200 は HA 環境でのみ有効です。「稼働」トラップと同等ではありません。SNMP エージェントが分割トラップ用に設定されている場合、HA ノードが変更されると、以前の機能と同様に 1 つの 5200 トラップのみがエンド SNMP に送信されます）

これらのコードはまれな状況を示しています。メッセージが可変長であるため、SNMP トラップ内のメッセージは（ESC コンポーネントメッセージとは異なり）変更されませんが、コードから状況と解決策を見つけることができます。5 シリーズのコードはエラーの状況を示し、2 シリーズのコードは以前の状況が修正されたことを示します。

SNMPトラップの重複と欠落

SNMP エージェントがすべての ESC コンポーネントのステータスを常にポーリングしている場合、ESCコンポーネントのステータスは保持されません。したがって、SNMPエージェントが再起動されると、ESC コンポーネントステータスの以前のビューは失われます。これにより、次の 2 つのシナリオが発生します。

- **SNMP トラップの重複**：SNMPエージェントが再起動される前にコンポーネントがダウンした場合、SNMPエージェントは重複したSNMP トラップを送信します。重複したSNMPトラップが送信されるのはまれです。

  たとえば、ESC マネージャrがダウンし、SNMP エージェントが再起動された場合、次のトラップが生成されます。

  5010 - ダウン、ESC マネージャ

  - SNMP エージェントがダウン
  - SNMP エージェントが起動して、ESC コンポーネントステータスを取得し、ESC マネージャがダウンしていることを確認すると、重複した SNMP トラップを生成

  5010 - ダウン、ESC マネージャ

- **SNMP トラップの欠落**：SNMPエージェントは、SNMPエージェントがダウンした場合にESC コンポーネントステータスの変更に対して生成されるはずの SNMP トラップを送信しない場合があります。まれに、有効なSNMPトラップを送信されない場合があります。
- たとえば、ESCマネージャがダウンし、SNMPエージェントが再起動された場合、次のトラップが生成されます。

  5010 - ダウン、ESC マネージャ

  - SNMP エージェントがダウン
  - ESC Manager が起動し、SNMP エージェントが **2010** を送信しない
  - SNMP エージェントが起動し、ステータスを取得し、ESC が正常であると認識すると、ESCマネージャの稼働トラップを送信しなかった場合でも、単一のトラップを送信

  2000 - 稼働、すべての ESC サービス

  SNMPエージェントはこのシナリオを管理するために、再起動時に常にトラップを生成します。トラップのステータスコードが「2000 - ESESC サービス、ETSI が実行されていません。（ESC service, ETSI is not running.）」の場合、エンドクライアントで以前の未確認トラップをクリアする必要があります。

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。

ステップ 1 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）。

例：

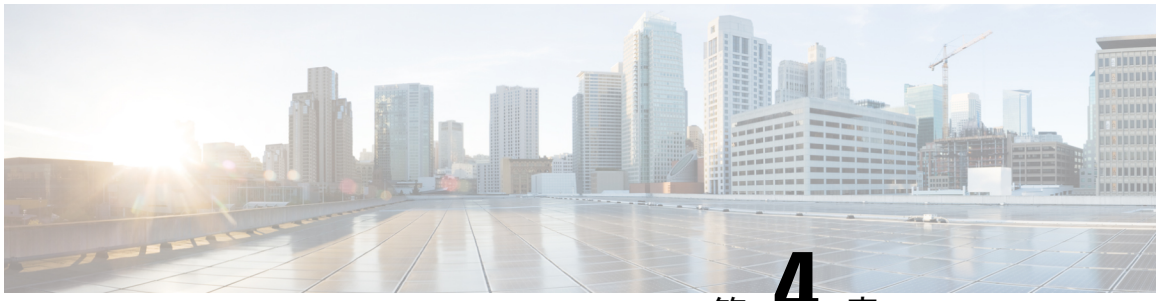
```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 esc
```

ステップ 2 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:8060:/esc/health"
```

ステップ 3 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>]
展開の詳細	[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 --user <username> --password <password> 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

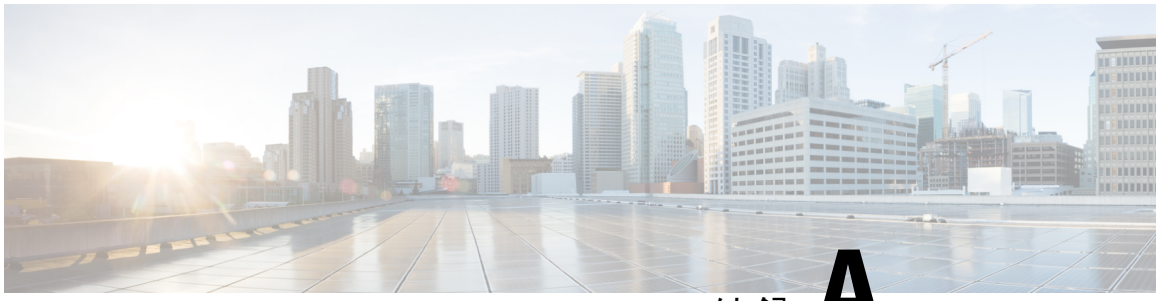
ファイル	コンポーネント	説明	ローテーションサイズ	バックアップファイルの数	アクティブ/アクティブ展開
/var/log/esc/yangesc.log	ESCManager	これには、netconf 要求と通知に関連するログが含まれています。	150 MB	10	応対可
/var/log/esc/error_escmanager.log	ESCManager	すべてのエラーログエントリ。	150 MB	10	応対可
/var/log/esc/trace/event_escmanager.log	ESCManager		150 MB	10	応対可
/var/log/esc/trace/escdatabase.log	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 マネージャ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

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_monitor_output.log	ESC インフラまたは HA		10 MB	4	使用不可
/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

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/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
/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

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/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.log	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。	150 MB	10	アクティブ/アクティブのみで使用可能
/var/log/esc/consul_agent.log	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。	150MB	10	アクティブ/アクティブのみで使用可能
/var/log/esc/geo.log	GEO サービス	ログエントリは GEO の状態と遷移を記録	150MB	10	GEO 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。

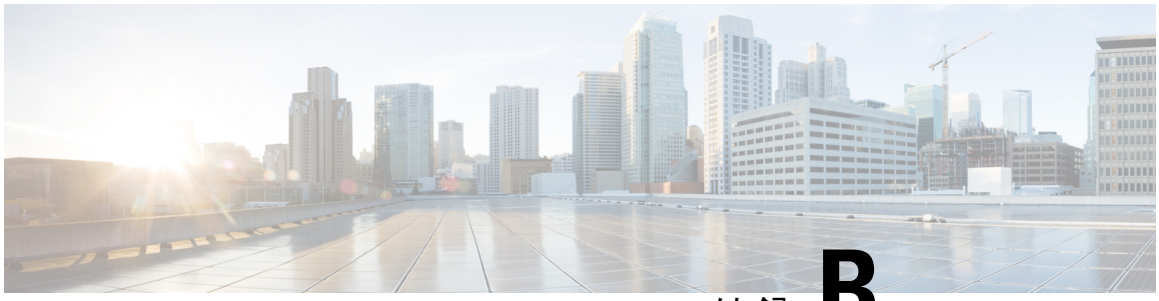
ESC のエラー状態

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
```

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 --user <username> --password <password> get-config esc_datamodel > <file-name>
```

次に例を示します。

```
esc_nc_cli --user <username> --password <password> get-config esc_datamodel >
/var/tmp/esc_datamodel.txt
```

CLI を使用して ESC アクティブ/アクティブ HA から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。

```
esc_nc_cli --host db.service.consul --user admin --password password get-config
esc_datamodel
```

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。

テクニカルサポートに連絡する前に

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 --user <username> --password <password> get-config esc_datamodel > <file-name>`
     - `esc_version`
     - `health.sh`
     - `escadm status`
     - `escadm vim show`

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。