Cisco Elastic Services Controller 5.9 ユーザーガイド

初版：2022 年 11 月 25 日

最終更新：2022 年 3 月 7 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、VNF のライフサイクル管理操作、モニタリング、修復、スケーリングなどのタスク実行を支援するためのものです。

- 対象読者 （xiii 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。

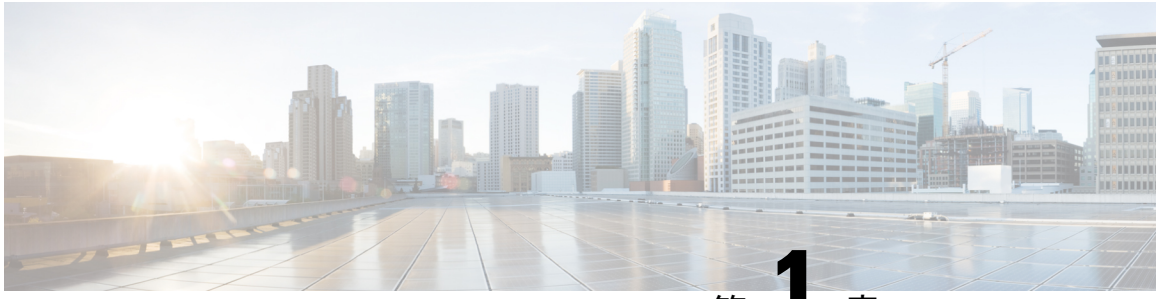
ドキュメントの入手方法

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 I 部

はじめに



第 1 章

Elastic Services Controller の概要

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

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

Elastic Services Controller の主な機能

- マルチベンダー OSS、NFVO、VNF、VIM のサポートを可能にするオープンなモジュラアーキテクチャを提供します。
- 単一の設定ポイントを使用して、仮想化サービスのエンドツーエンドの動的プロビジョニングとモニタリングを提供します。
- ライフサイクル管理のさまざまなフェーズでカスタマイズを提供し、同時に、VM、サービスアドバタイズメント、およびカスタムアクションをモニタリングします。

- 統合された Monitoring Actions（MONA）エンジンによるエージェントレス モニタリングを提供します。モニタリングエンジンは、VM のスケールインとスケールアウトを決定する単純なルールと複雑なルールを提供します。
- ネットワークの負荷に基づいてスケールインとスケールアウトのオプションを提供します。
- 修復の一環として検出されたモニタリングエラーとしきい値の条件に基づいて、VM を展開、再起動、または再展開します。
- VNF の展開とライフサイクル管理を迅速化することで、サービスの俊敏性をサポートします。
- マルチテナント環境をサポートします。
- 複数の VIM での VM の展開をサポートします。
- OpenStack で ESC ユーザの非管理者ロールをサポートします。
- OpenStack で IPv6 をサポートします。
- OpenStack でデュアル スタック ネットワークをサポートします。
- REST および NETCONF/YANG インターフェイスをサポートし、階層構成とデータのモジュール性を提供します。
- VNF ライフサイクル管理操作のサブセットに対して ETSI MANO インターフェイスをサポートします。
- ETSI パフォーマンスレポートをサポートします。
- 単一または複数の AWS VIM での VM の展開をサポートします。
- ESC REST API と ETSI API の両方を使用した VMware vCloud Director VIM への vApp の展開をサポートします。
- アクティブ/アクティブ設定での D-MONA の展開とモニタリングをサポートします。Distributed Monitoring and Actions（D-MONA）は、VNF をモニタリングするためのスタンドアロン モニタリング コンポーネントです。
- ブラウンフィールド VM の展開をサポートします。
- スケーリング中のリソース値の一貫した順序付けをサポートします。

ESC アーキテクチャ

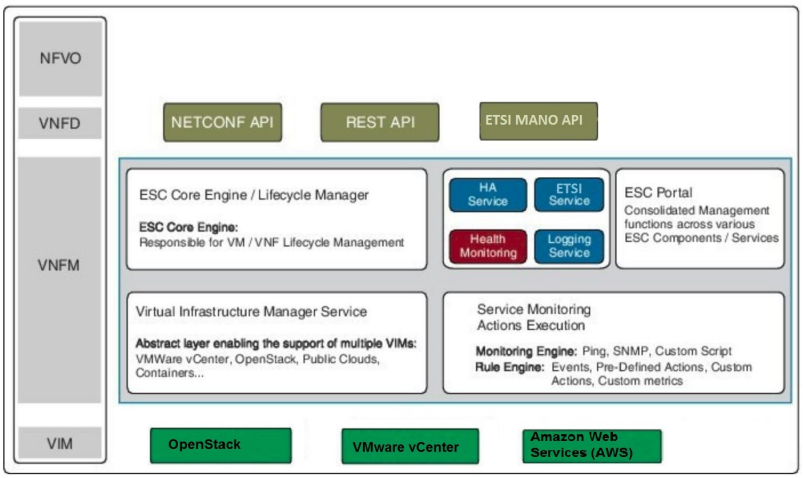
Cisco Elastic Services Controller（ESC）は、マルチベンダーサポートを可能にするオープンなモジュラアーキテクチャとして構築されています。ESC では VNF のライフサイクル管理が実行されます。つまり、VNF のオンボーディング、展開、モニタリング、および KPI 要件に基づく修復やスケーリングなどの VNF レベルのライフサイクルの決定が行われます。ESC とその管理対象 VNF は、仮想インフラストラクチャ マネージャ（VIM）内で実行される VM として

展開されます。現在サポートされている VIM は、OpenStack、VMware vCenter、および AWS です。ESC コアエンジンは、トランザクション、検証、ポリシー、ワークフロー、および VM ステートマシンを管理します。ESC のモニタリングおよびアクションサービスエンジンは、複数のモニタリング方式に基づいてモニタリングを実行します。イベントは、モニタリングアクションに基づいてトリガーされます。モニタリングエンジンは、カスタム モニタリング プラグインもサポートします。

ESC は高可用性用に設定できます。詳細については、Cisco Elastic Services Controller インストールおよびアップグレードガイド [英語] を参照してください。

ESC では、REST、NETCONF/YANG、および ETSI NFV MANO NB API（ETSI API）を使用して、トップ オーケストレーション レイヤと情報が交換されます。オーケストレーションレイヤは、Cisco NSO、サードパーティの OSS、または NFV Orchestrator にすることができます。ESC は、NETCONF/YANG ノースバウンド インターフェイス サポートを使用して NSO と統合されます。設定テンプレートである仮想ネットワーク機能記述子（VNFD）ファイルは、VNF の展開パラメータと運用動作を記述するために使用されます。VNFD ファイルは、VNF をオンボーディングし、VNF インスタンスのライフサイクルを管理するプロセスで使用されます。次の図は、Cisco Elastic Services Controller アーキテクチャを表しています。

図 *1 : Cisco Elastic Services Controller* アーキテクチャ



ESC ライフサイクルについて

Cisco Elastic Services Controller（ESC）は、動的な環境で汎用仮想ネットワーク機能（VNF）における VNF ライフサイクルのすべての側面を管理する単一の制御ポイントを提供します。また、ETSI VNF 管理およびオーケストレーション（MANO）リファレンスアーキテクチャに準拠したオープンな標準ベースのプラットフォームを通じて、高度な VNF ライフサイクル管理機能を提供します。

OpenStack または VMware vCenter のいずれかで、仮想インフラストラクチャ ドメイン内の VNF をオーケストレーションできます。VNF 展開は、サービスリクエストとして開始されます。

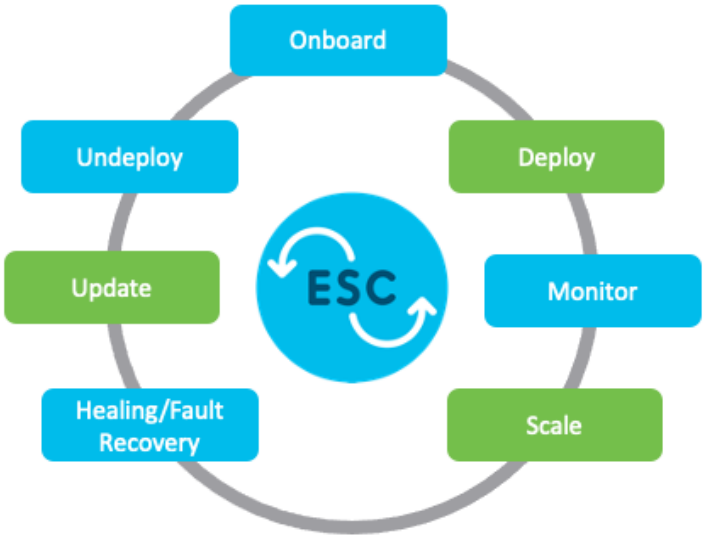
サービスリクエストは、XML ペイロードと設定パラメータから成るテンプレートで構成されます。

（注） 異なる VIM や VIM タイプに VNF を展開するハイブリッド展開がサポートされていますが、それらの VM 間のルーティングは ESC では実行されません。

ESC は、VNF のライフサイクル全体を管理します。VNF 展開は、ノースバウンド インターフェイスまたは ESC ポータルを介してサービスリクエストとして開始されます。

次の図は、ESC のライフサイクル管理を示しています。

図 *2 : ESC* の *VNF* ライフサイクル



- オンボーディング：ESC では、OpenStack および VMware vCenter でサポートするための前提条件を満たしている限り、新しい VNF タイプをすべてオンボーディングできます。たとえば、OpenStack では、Cisco ESC は raw イメージ、qcow2、および vmdk ディスク形式をサポートします。ESC は、VNF ブートストラップメカニズムのコンフィグドライブもサポートします。新しい VNF タイプの XML テンプレートを定義して、ESC で VNF をオンボードできます。

  ETSI API を使用して、VNF は NFVO にオンボードされます。詳細については、Cisco Elastic Services Controller ETSI NFV MANO ユーザーガイド [英語] の「VNF Lifecycle Operations」セクションの前提条件を参照してください。

- 展開：VNF が展開されると、ESC が新しいサービスにデイゼロ設定を適用します。一般的な設定には、新しい仮想リソースをシステムで使用可能にするためのログイン情報、ライセンス、接続情報（IP アドレス、ゲートウェイ）、およびその他の静的パラメータが含まれます。また、新しい VNF のライセンスもアクティブにします。

識別子は、ライフサイクルのこの段階で、ETSI API を使用して作成されます。詳細については、Cisco Elastic Services Controller ETSI NFV MANO ユーザーガイド [英語] の「Creating VNF Identifier」セクションを参照してください。

- モニタリング：ESC は、ICMP ping、SNMP などのさまざまな方法を使用して仮想マシンの正常性をモニタします。また、CPU 使用率、メモリ消費量、その他のコアパラメータなどの評価指標を追跡します。リクエスタは、仮想マシンの起動と管理に通常関連するすべての特性（vCPU、メモリ、ディスク、モニタリング KPI など）を XML テンプレートで指定できます。また、サービスパフォーマンス関連のメトリックおよびユーザが定義するその他の主要なパラメータをモニタするための複雑なフレームワークも提供します。
- 修復：ESC は障害が発生したときに VNF を修復します。障害シナリオは、データモデルの KPI セクションで設定されます。ESC は KPI を使用して VM をモニタします。イベントは KPI 条件に基づいてトリガーされます。トリガーされるすべてのイベントに対して実行されるアクションは、展開時にルールセクションで設定します。
- 更新：ESC では、展開が成功した後で展開を更新できます。すべての更新（つまり、vm_group の追加や削除、vm_group でのエフェメラルネットワークの追加や削除、および vm_group でのインターフェイスの追加や削除）を単一の展開で実行することも、個別に実行することもできます。
- 展開解除：ESC では、すでに展開されている VNF を展開解除できます。この操作は、ノースバウンド API を使用するか、または ESC ポータルを介して実行されます。

  ETSI API を使用して VNF を削除すると、関連する識別子も削除されます。

（注） ETSI API を使用した完全な VNF ライフサイクル操作については、Cisco Elastic Services Controller ETSI NFV MANO ユーザーガイド [英語] を参照してください。

次のセクションでは、OpenStack および VMware vCenter に VNF を展開する方法について説明します。

OpenStack での VNF の展開

ESC では、VNF の展開は、ESC ポータルまたはノースバウンド インターフェイスから発信されるサービスリクエストとして開始されます。サービスリクエストは、XML ペイロードから成るテンプレートで構成されます。これらのリソースは、OpenStack で使用できるか、ESC ポータルまたはノースバウンド インターフェイスを使用して ESC で作成できる必要があります。ESC でのリソース管理の詳細については、リソース管理の概要 （17 ページ）を参照してください。展開データモデルは、OpenStack に VNF を展開するためのリソースを参照します。

リソースの設定方法に基づいて、次のいずれかの方法で VNF を展開できます。

シナリオ	説明	リソース	利点
ESC を使用してイメージとフレーバーを作成することにより、単一の VIM に VNF を展開する	展開データモデルは、作成されたイメージとフレーバーを参照して、VNF を展開します。	イメージとフレーバーは、NETCONF/REST API を使用して ESC で作成されます。	• イメージとフレーバーは、複数の VNF 展開で使用できます。 • ESC によって作成されたリソース（イメージ、フレーバー、およびボリューム）を削除できます。
アウトオブバンドイメージ、フレーバー、ボリューム、およびポートを使用した単一 VIM への VNF の展開	展開データモデルは、OpenStack のアウトオブバンドイメージ、フレーバー、ボリューム、およびポートを参照して、VNF を展開します。	イメージ、フレーバー、ボリューム、およびポートは、ESC を使用して作成されません。	• イメージ、フレーバー、ボリューム、ポートは、複数の VNF 展開で使用できます。 • ESC を使用して作成されていないリソースは削除できません。
アウトオブバンドリソースを使用した複数の VIM への VNF の展開	展開データモデルは、アウトオブバンドイメージ、フレーバー、ネットワーク、および VIM プロジェクトを参照して、VNF を展開します。	イメージ、フレーバー、VIM プロジェクト（ロケータで指定）およびネットワークは、ESC を使用して作成されません。これらは、VIM のアウトオブバンドに存在する必要があります。	展開内の ESC で設定する必要がある（VM を展開するための）VIM を指定できます。

（注） OpenStack での VNF の展開の詳細については、OpenStack での仮想ネットワーク機能の展開（115 ページ）を参照してください。

VMware vCenter での VNF の展開

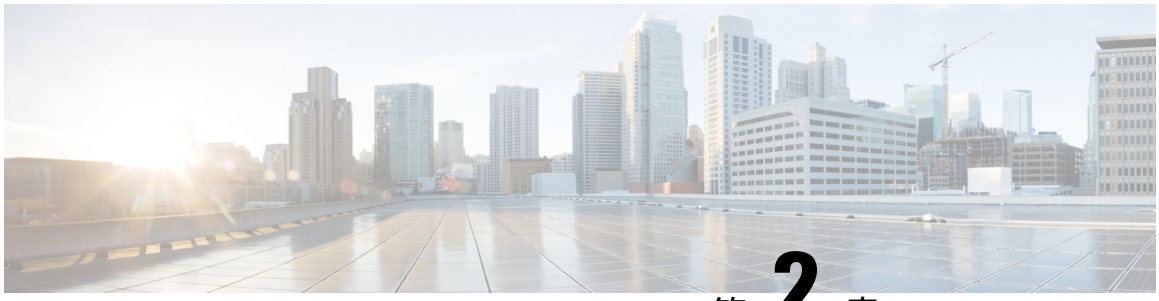
ESC では、VNF の展開は、ESC ポータルまたはノースバウンド インターフェイスから発信されるサービスリクエストとして開始されます。サービスリクエストは、ネットワーク、イメージなどの XML ペイロードから成るテンプレートで構成されます。これらのリソースは、VMware vCenter で使用できる必要があります。ESC での VM リソースの管理の詳細については、リソー

ス管理の概要 （17 ページ）を参照してください。展開データモデルは、VMware vCenter に VNF を展開するためのリソースを参照します。

VMware vCenter に VNF を展開する場合は、VMware vCenter ですでに使用可能なアウトオブバンドイメージを使用するか、ESC ポータルまたは REST API を使用してイメージを作成できます。ESC ポータルでのイメージの作成の詳細については、イメージの管理 （35 ページ）を参照してください。展開データモデルは、VNF を展開するためにこれらのイメージを参照します。

シナリオ	説明	データモデルテンプレート	画像	利点
ESC を使用したイメージの作成による VNF の展開 **重要** イメージは、VMware vCenter のテンプレートとも呼ばれます。	VNF 展開のプロセスは次のとおりです。 1. VNF 展開：展開データモデルは、作成されたイメージを参照して、VNF を展開します。	• 展開データモデル • イメージデータモデル	イメージは、REST API を使用して ESC で作成されます。	• イメージは、複数の VNF 展開で使用できます。 • ESC を使用してイメージ定義を追加または削除できます。
アウトオブバンドイメージを使用した単一 VIM への VNF の展開	1. VNF 展開：展開データモデルは、VMware vCenter のアウトオブバンドイメージを参照して、VNF を展開します。	• 展開データモデル • VMware vCenter のイメージ	ESC を使用してイメージを作成または削除することはできません。	• イメージは、複数の VNF 展開で使用できます。 • ESC ポータルからイメージを確認できます。 • アウトオブバンド展開中に、イメージを選択できます。

VMware vCenter での VNF の展開の詳細については、VMware vCenter のイメージ （147 ページ）を参照してください。



第 2 章

Elastic Services Controller インターフェイス

Elastic Services Controller インターフェイス

Cisco Elastic Services Controller（ESC）は、次のいずれかの方法で展開できます。

- Cisco Orchestration スイートの一部として展開：ESC は Cisco Network Services Orchestrator（NSO）にパッケージ化されており、Cisco Managed Services Accelerator（MSX）などのシスコのソリューション内で使用できます。
- スタンドアロン製品として展開：ESC は、VPN、vRouter、vSecurity などの Cisco VNF にバンドルされた VNFM として使用できます。

ESC が MSX、VPN、vRouter などの一部として展開されると、これらのアプリケーションはノースバウンド API を介して ESC とインターフェイスで接続します。ESC は、操作およびトランザクション用の REST および NETCONF ノースバウンド インターフェイスをサポートしています。ESC ポータルは、仮想ネットワーク機能ライフサイクル管理のタスクの一部について CRUD 操作をサポートします。

この章では、ノースバウンド API と ESC ポータルについて説明します。

Elastic Services Controller NB API

Elastic Services Controller（ESC）は、操作およびトランザクション用の REST および NETCONF ノースバウンド インターフェイスをサポートしています。

ノースバウンドインターフェイスは、NB クライアント、NSO、または任意の OSS と情報をやりとりします。REST インターフェイスの相互作用では、コールバックがトリガーされ、NETCONF/YANG インターフェイスの相互作用では、NETCONF 通知がトリガーされます。

ESC は、ETSI NFV MANO 標準に準拠する REST API もサポートしています。ETSI API を使用する場合、他の ESC インターフェイスは、データモデルの整合性を維持するために、情報の読み取り専用で使用する必要があります。ETSI API の詳細は、このドキュメントの範囲外です。詳細については、Cisco Elastic Services Controller ETSI NFV MANO ユーザーガイド [英語] を参照してください。

NETCONF/YANG ノースバウンド API

ESC は NETCONF を使用して、ネットワークとそのデバイスを設定および管理します。NETCONF は、ネットワークデバイスの設定をインストール、操作、処理、および削除するためのネットワーク管理プロトコルです。Cisco NSO は、オープンな NETCONF プロトコルと YANG ベースのデータモデルを使用して ESC と通信します。ESC は仮想ネットワーク機能をデバイスレベルで管理し、NSO はネットワーク サービス ライフサイクル全体を管理します。これらを組み合わせることで、物理インフラストラクチャと仮想インフラストラクチャの両方にまたがる完全なオーケストレーション ソリューションとなります。

（注） netconf CLI を使用した CRUD 操作の完全なパスを入力する必要はなく、*esc_nc_cli --user <username> --password <password> command <file name>* と入力するだけです。CLI の詳細については、Cisco Elastic Services Controller インストールおよびアップグレードガイド [英語] を参照してください。

NETCONF/YANG モデルは、NETCONF 通知とともに運用データも提供します。クエリを実行して、ESC のすべてのテナント、ネットワーク、および展開のリストなどの詳細を取得できます。

単一の NETCONF 要求を作成して、複数のアクションを実行できます。詳細については、「NETCONF 機能拡張要求」を参照してください。次に、2 つのテナントを同時に削除する NETCONF 要求を示します。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
<tenants>
  <tenant nc:operation="delete">
    <name>abc-mix-tenant1</name>
  </tenant>
  <tenant nc:operation="delete">
    <name>abc-mix-tenant2</name>
  </tenant>
</tenants>
</esc_datamodel>
```

次に、NETCONF/YANG API の例を示します。

テナントを作成する NETCONF 要求

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <edit-config>
    <source>
      <running />
    </source>
```

```
    <config>
      <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
        <tenants>
          <tenant>
            <name>mytenant</name>
          </tenant>
        </tenants>
      </esc_datamodel>
    </config>
  </edit-config>
</rpc>
```

設定のアクティブ化が完了すると、ステータスが SUCCESS の CREATE_TENANT タイプの escEvent が NETCONF サブスクライバに送信されます。これは、アクティベーション ワークフローが完了し、設定リソースが VIM で正常に作成されたことを示します。

テナントが正常に作成された後の NETCONF 通知：

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-05-05T19:38:27.71+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_message>Tenant successfully created</status_message>
    <tenant>mytenant</tenant>
    <vm_source />
    <vm_target />
    <event>
      <type>CREATE_TENANT</type>
    </event>
  </escEvent>
</notification>
```

テナントの運用データ（Opdata）には、名前と tenant_id が表示されます。NETCONF 要求

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get>
    <filter select="esc_datamodel/opdata/tenants/tenant[name='mytenant']" type="xpath"
/>
  </get>
</rpc>
```

NETCONF 応答

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
      <opdata>
        <tenants>
          <tenant>
            <name>mytenant</name>
            <tenant_id>dccd22a13cc64e388a4b8d39e6a8fa7f</tenant_id>
        </tenant>
      </tenants>
    </esc_datamodel>
  </data>
</rpc-reply>
```

一連の通知、イベント障害通知、および opdata の詳細については、Cisco Elastic Services Controller API ガイド [英語] を参照してください。

NETCONF APIの設定とRPCコールが検証されます。有効でない要求は拒否されます。NETCONF API は、REST とは異なり、エラーコードを NB に送信しません（たとえば、REST は 404 Not Found エラーを送信します）。

サンプルエラーメッセージ（拒否された要求）は次のとおりです。

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-path xmlns:esc="http://www.cisco.com/esc/esc"

xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">/nc:rpc/esc:filterLog</error-path>
       <error-message xml:lang="en">Exception from action callback: Error when handling
 RPC
            calls: You can only query up to 30 logs.</error-message>
        <error-info>
            <bad-element>filterLog</bad-element>
        </error-info>
    </rpc-error>
</rpc-reply>
```

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>10.0.0.0</address>
            <no_gateway>true</no_gateway>
            <!-- DISABLE GATEWAY -->
            <gateway>10.0.0.1</gateway>
            <netmask>255.255.255.0</netmask>
        </subnet>
    </network>
</networks>
```

ESC の [運用データ（Operational Data）] セクションに OpenStack と VMware vCenter のユーザ名が表示されます。

次の設定の詳細が [運用データ（Operational Data）] に表示されます。

Openstack

- active_vim：値が OpenStack として表示されます。
- os_auth_url：OpenStack 認証 URL が表示されます。
- admin_role：OpenStack ユーザが管理者であるかどうかが表示されます。
- os_tenant_name：テナントが表示されます。

- os_username：OpenStack ユーザが表示されます。
- member_role：OpenStack ユーザがメンバーであるかどうかが表示されます。

VMware vCenter

- active_vim：値が VMware として表示されます。
- vcenter_ip：vCenter IP アドレスが表示されます。
- vcenter_port：vCenter ポートであるかどうかが表示されます。
- vcenter_username：vCenter ユーザが表示されます。

複数リソースを設定するための NETCONF 要求

ユーザは単一の NETCONF 要求を作成して、複数のリソースを設定できます。

（注） 複数のリソースを設定する単一の要求は、NETCONF を使用してのみサポートされます。

単一の NETCONF 要求は、リソース間の依存関係に基づいて複数のリソースを関連付けます。たとえば、サブネットはネットワークに依存し、展開はイメージとフレーバーに依存します。

ESC には 2 種類の依存関係があります。

1. 参照型依存関係
2. 階層型依存関係

参照型依存関係

参照型依存関係では、1 つの設定に別の設定への参照があります。

次の例では、展開にイメージ（test-mix-cirros）とフレーバー（test-mix-small）への参照型依存関係があります。イメージとフレーバーは、展開設定の前に作成する必要があります。

```
  <images>
    <image>
        <name>test-mix-cirros</name>
...
</image>
</images>
<flavors>
    <flavor>
        <name>test-mix-small</name>
...
</flavor>
</flavors>
<tenants>
    <tenant>
        <name>test-mix-tenant</name>
        <deployments>
            <deployment>
```

```
                <name>dep</name>
                <vm_group>
                    <name>Group1</name>
                    <image>test-mix-cirros</image>
                    <flavor>test-mix-small</flavor>
...
</vm_group>
            </deployment>
        </deployments>
    </tenant>
</tenants>
```

階層型依存関係

階層型依存関係では、1 つの設定が別の設定の中にあります。

次の例では、サブネット（test-mix-shared-subnet1）はネットワーク（test-mix-shared-net1）の中にあります。サブネットには、ネットワークに対する階層型依存関係があります。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
<networks>
    <network>
        <name>test-mix-shared-net1</name>
        <shared>true</shared>
        <admin_state>true</admin_state>
        <subnet>
            <name>test-mix-shared-subnet1</name>
            <ipversion>ipv4</ipversion>
            <dhcp>true</dhcp>
            <address>10.193.90.0</address>
            <netmask>255.255.255.0</netmask>
            <gateway>10.193.90.1</gateway>
        </subnet>
    </network>
</networks>
</esc_datamodel>
```

階層型依存関係は、参照型依存関係のサブセットです。リソースにおけるこれらの設定の依存関係により、NETCONF は単一の要求を使用して複数の設定を実行できます。

REST ノースバウンド API

REST API は、Representational State Transfer（REST）アーキテクチャを使用する ESC へのプログラマチック インターフェイスです。API は JavaScript オブジェクトの表記（JSON）または Extensible Markup Language（XML）のマニュアルを含む HTTP または HTTPS メッセージを受け入れて返します。プログラミング言語を使用して、API メソッドまたは管理対象オブジェクト（MO）の説明を含むメッセージおよび JSON または XML ドキュメントを生成できます。

API モデルには、これらのプログラマチック エンティティが含まれます。

- クラス：管理情報ツリー（MIT）のオブジェクトのプロパティおよび状態を定義するテンプレート。
- メソッド：1 つまたは複数のオブジェクトに対して API が実行するアクションです。
- タイプ：オブジェクト ステート（たとえば、equipmentPresence）に値をマッピングするオブジェクトのプロパティ。

ESC REST API には、ヘッダーとその他のパラメータが含まれています。header パラメータには、URI があるコールバックフィールドが含まれています。クライアントコールバックではこの値があることを想定しています。URI フィールドが存在しない場合、コールバックは実行されません。

REST API ドキュメント

REST API ドキュメントには、ESC VM から直接アクセスできます。

```
http://[ESC VM IP]:8080/ESCAPI
```

詳細については、Cisco Elastic Services Controller API ガイド [英語] も参照してください。

REST API ドキュメントには、REST インターフェイスでサポートされるさまざまな操作の詳細が記載されています。

REST API の例：

REST を使用してテナントを作成するには、次の手順を実行します。

```
POST /v0/tenants/123 HTTP/1.1
Host: client.host.com
Content-Type: application/xml
Accept: application/xml
Client-Transaction-Id: 123456
Callback:/createtenantcallback
<?xml version="1.0" encoding="UTF-8"?>
<tenant xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <name>tenant1</name>
   <enabled>true</enabled>
   <description>A description...</description>
</tenant>
```

テナントが正常に作成された後の REST 応答：

```
HTTP/1.1 201 OK
Content-Type: application/xml; charset=UTF-8
Content-Length: 200
Date: Sun, 1 Jan 2011 9:00:00 GMT
ESC-Transaction-Id: 123456
ESC-Status-Code: 200
ESC-Status-Message: Success …
<?xml version="1.0" encoding="UTF-8"?>
<tenant>
   <external_tenant_id>234243490854004</external_tenant_id>
   <internal_tenant_id>434344896854965</internal_tenant_id>
   <name>tenant1</name>
   <enabled>true</enabled>
   <description>A description...</description>
</tenant>
```

REST API を使用して、同じテナント名と展開名で VNF を展開することはできません。

（注） さらに、このドキュメントでは、REST または NETCONF/YANG のいずれかを使用するシナリオの例を示しますが、両方の使用例はありません。

ETSI NFV MANO Northbound API

ETSI NFV MANO API（ETSI API）は、REST アーキテクチャを使用する ESC への、別のプログラム可能なインターフェイスです。ETSI MANO は、欧州電気通信標準化機構（ETSI）によって定義された標準、特に管理/オーケストレーション（MANO）関連に準拠しています。

詳細については、『*Cisco Elastic Services Controller ETSI NFV MANO Guide*』の「ETSI NFV MANO Northbound API Overview」を参照してください。

ETSI API ドキュメント

ETSI API ドキュメントには、ESC VM から直接アクセスできます。

```
http://[ESC VM IP]:8250/API
```

ETSI API ドキュメントには、ESTI MANO インターフェイスでサポートされるさまざまな操作の詳細が記載されています。詳細については、『Cisco ETSI API Guide』も参照してください。

Elastic Services Controller ポータル

ESC ポータルは、ESC 管理者が VNF ライフサイクル管理に関連する CRUD 操作（作成、読み取り、更新、または削除）を行うためのシンプルな Web ベースツールです。管理者は、展開、展開解除、修復、スケーリングなど、ESC のリアルタイムアクティビティを作成して表示できます。

ESC ポータルは、OpenStack で ESC VM、または KVM で VMware vCenter を作成するときにデフォルトで有効になります。ESC ポータルの有効化または無効化の詳細については、「ESC ポータルダッシュボード」を参照してください。

ESC ポータルを開始、停止、および再起動するには、次の手順を実行します。

- ESC ポータルを開始するには、sudo escadm portal start を実行します。
- ポータルを停止するには、sudo escadm portal stop を実行します。
- ポータルを再起動するには、sudo escadm portal restart を実行します。

（注） 推奨されるブラウザの画面サイズは、1920 X 1080 ピクセルです。

第 II 部

リソースの管理

リソース管理の概要

Cisco Elastic Services Controller（ESC）リソースは、イメージ、フレーバ、テナント、ボリューム、ネットワーク、およびサブネットワークで構成されます。これらのリソースは、ESC が仮想ネットワーク機能のプロビジョニングを要求するためのものです。これらのリソースは、VNF サービス要求の基本的な構成要素を構成します。たとえば、イメージは、VM インスタンスの起動に使用できるブート可能なファイルシステムです。これらのリソースを管理するには、ESC で対応するリソースを作成する必要があります。これらのリソース定義は、プロビジョニングされたインフラストラクチャに基づいて OpenStack または VMware vCenter 上に存在するか、または作成されます。

VNF 展開のタイプに応じて、必要なリソース定義が OpenStack または VMware vCenter で使用できることを確認する必要があります。OpenStack に VNF を展開する場合は、ESC でこれらの

リソース定義を作成するか、OpenStack ですでに使用可能なアウトオブバンドイメージおよびフレーバの定義を使用するオプションがあります。アウトオブバンドリソースは既存のリソースです。このリソースは、ESC 自体または別のソースによって作成されます。マルチ VIM 展開の場合、ESC はアウトオブバンドリソースを使用します。ESC は、マルチ VIM 展開用に複数の VIM コネクタをサポートします。VIM コネクタは、ESC を複数の VIM に接続します（設定されている場合）。

ESC はプロキシサーバ（使用可能な場合）を使用して OpenStack に到達します。

VMware vCenter に VNF を展開する場合は、VMware vCenter ですでに使用可能なアウトオブバンドイメージを使用するか、ESC ポータルまたは REST API を使用してイメージを作成できます。ESC ポータルを使用したイメージの作成の詳細については、イメージの管理 （35 ページ）を参照してください。展開データモデルは、VNF を展開するためにこれらのイメージを参照します。

（注） リソース定義を作成する手順は、OpenStack と VMware vCenter で異なります。

ESCから作成されたリソース（イメージ、展開など）の名前は、グローバルに一意である必要があります。

次の表に、さまざまな環境と、VNFの展開前に使用可能にする必要があるリソース定義のリストを示します。

リソースの定義	OpenStack	VMware vCenter
テナント	テナント定義の作成と削除は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル	適用なし
ネットワーク	ネットワーク定義の作成と削除は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル	分散ポートグループ定義の作成と削除は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル

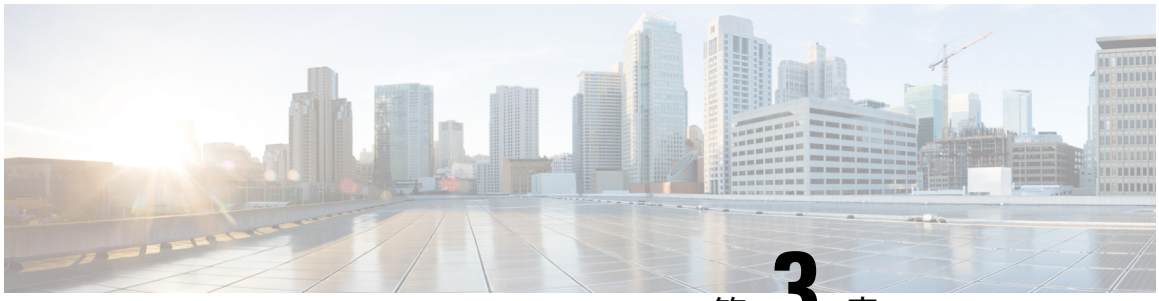
リソースの定義	**OpenStack**	**VMware vCenter**
サブネット	サブネット定義の作成と削除は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル	適用なし
フレーバ	OpenStack ですでに使用可能なアウトオブバンドフレーバ定義を使用するか、次のいずれかの方法でフレーバ定義を作成できます。 • NETCONF API • REST API • ESC ポータル	適用なし
画像	OpenStack ですでに使用可能なアウトオブバンドイメージ定義を使用するか、次のいずれかの方法でイメージ定義を作成できます。 • NETCONF API • REST API • ESC ポータル	VMware vCenter ですでに使用可能なアウトオブバンドイメージ定義を使用するか、次のいずれかの方法でイメージ定義を作成できます。 • NETCONF API • REST API • ESC ポータル
音量	OpenStack ですでに使用可能なアウトオブバンドボリュームを使用できます。詳細については、ボリュームの管理 （36 ページ）を参照してください。	適用なし

次の表に、ESC がサポートする OpenStack および VMware のバージョンを示します。

表 *2 : OpenStack* および *VMWare* のサポートされているバージョン

VIM	バージョン
OpenStack	• ニュートン • Ocata • Queens • Keystone v2 および v3
VMware	VMware および vCenter バージョン 5.5、6.0、6.5

ESC のインストールの詳細については、Cisco Elastic Services Controller インストールおよびアップグレードガイド [英語] を参照してください。



第 3 章

OpenStack のリソースの管理

OpenStack のリソースの管理

テナントの管理

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

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

ESC では、次の 3 種類のテナントを作成できます。

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

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

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

テナント名は一意である必要があります。

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

データモデルのテナントは、次の属性で管理します。

- managed_resource 属性
- vim_mapping 属性

次の表に、データモデルのテナントと属性のマッピングの詳細が示されています。

テナントタイプ	managed_resource	vim_mapping	説明

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

テナントタイプ	managed_resource	vim_mapping	説明

VIM 上の既存のテナント	false	true	既存のテナントの場合、managed_resource 属性は false に設定されます。vim_mapping 属性は true です。
			
			`<tenants>`
			`<tenant>`
			`<name>pre-existing</name>`
			
			`<managed_resource>false</managed_resource>`
			`</tenant>`
			`</tenants>`
			
			テナント UUID を使用したサンプルデータモデル
			
			`<tenants>`
			`<tenant>`
			
			`<name>76eedcae-6067-44a7-b733-fc99a2e50bdf</name>`
			
			`<managed_resource>false</managed_resource>`
			`</tenant>`
			`</tenants>`

ESC 内のテナント	-	false	ESC 内にテナントを作成するには、vim_mapping 属性を false に設定します。
			
			`<tenants>`
			`<tenant>`
			`<name>esc-tenant-A</name>`
			`<vim_mapping>false</vim_mapping>`
			`</tenant>`
			`</tenants>`

-	false	false	テナントは作成されません。要求はESCによって拒否されます。

同じタイプの複数の VIM（OpenStack VIM）に VM を展開するには、vim_mapping 属性を false に設定してテナントを作成する必要があります。このテナントは、個別に作成することも、展開の一部として作成することもできます。これにより、ESC内にテナントが作成され、マルチ VIM 展開のルートテナントとして機能します。マルチ VIM 展開の場合は、各 VM グループ内で VIM ロケータ属性を指定する必要があります。詳細については、VMware vCenter VIM での VNF の展開 （148 ページ）を参照してください。

テナントクォータ

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

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

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

コンピューティングの設定：

- metadata_items
- floating_ips
- コア
- jected_file_path_bytes
- jected_files
- jected_file_content_bytes
- インスタンス
- key_pairs
- ram
- security_groups
- security_group_rules

コンピューティングの設定：

- floatingip
- security_group_rule
- security_group
- network
- サブネット
- port
- ルータ

次の展開データモデルは、テナントのクォータ設定を示しています。

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

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

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

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

ノースバウンド API を使用したテナントの追加

次に、NETCONF を使用してテナント定義を作成する例を示します。

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
        <source>
            <running />
        </source>
        <config>
            <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
                <tenants>
                    <tenant>
                        <name>mytenant</name>
                    </tenant>
                </tenants>
            </esc_datamodel>
        </config>
    </edit-config>
</rpc>
```

（注） NETCONF API を使用したテナント定義の作成と削除の詳細については、Cisco Elastic Services Controller API ガイド [英語] を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、REST ノースバウンド API （14 ページ）を参照してください。ESC ポータルを使用したネットワークの追加と削除の詳細については、ESC ポータルを使用したリソースの管理 （381 ページ）を参照してください。

テナントのクォータの更新

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

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

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

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

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

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

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

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

次の例は、データモデルからプロパティを削除する方法を示しています。

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

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

REST API を使用したテナントクォータの更新

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

方式タイプ：

```
PUT
```

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

HTTP 要求ヘッダー：

internal_tenant_id：更新するテナント ID

callback：残りのコールバック通知を受信するアドレスとポート

Content-Type：application/xml

クォータを指定してテナントを作成する際の REST API の例。

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

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

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

ネットワークの管理

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

OpenStack ネットワーク

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

ESC は次のネットワーキング機能をサポートしています。

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

  次の属性で、プロバイダーネットワークを定義します。

  - network_type
  - physical_network
  - segmentation_id

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

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

ノースバウンド API を使用したネットワークの追加

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

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
  <tenants>
```

```
   <tenant>
      <name>quicktest4</name>
  <networks>
   <network>
    <name>proto-tenant-network34</name>
    <shared>false</shared>
    <admin_state>true</admin_state>
   </network>
  </networks>
   </tenant>
  </tenants>
</esc_datamodel>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
  <tenants>
   <tenant>
      <name>quicktest4</name>
  <networks>
   <network>
    <name>proto-tenant-network27</name>
    <subnet>
     <name>proto-tenant-subnet4</name>
     <ipversion>ipv4</ipversion>
     <dhcp>true</dhcp>
     <address>172.16.0.0</address>
     <netmask>255.255.255.0</netmask>
     <gateway>172.16.0.1</gateway>
    </subnet>
   </network>
  </networks>
   </tenant>
  </tenants>
</esc_datamodel>
```

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

```
<?xml version="1.0"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
 xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
 xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <networks>
        <network>
          <name>test-net-12</name>
          <shared>true</shared>
          <admin_state>true</admin_state>
          <provider_physical_network>vm_physnet</provider_physical_network>
          <provider_network_type>vlan</provider_network_type>
          <provider_segmentation_id>200</provider_segmentation_id>
        </network>
    </networks>
</esc_datamodel>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <networks>
        <network>
            <name>test-net-12</name>
            <subnet>
                <name>test-net-12-subnet</name>
                <ipversion>ipv4</ipversion>
                <dhcp>false</dhcp>
                <address>172.16.0.0</address>
                <gateway>172.16.0.1</gateway>
                <netmask>255.255.255.0</netmask>
            </subnet>
        </network>
    </networks>
</esc_datamodel>
```

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

```
<?xml version="1.0"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
    xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
    xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <networks>
        <network>
          <name>ProviderNetworkAttributes-vxlan-evpn</name>
          <shared>true</shared>
          <provider_network_type>vxlan-evpn</provider_network_type>
          <provider_segmentation_id>3010</provider_segmentation_id>
        </network>
    </networks>
</esc_datamodel>
```

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

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

（注） NETCONF API を使用したネットワークの作成と削除の詳細については、Cisco Elastic Services Controller API ガイド [英語] を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、REST ノースバウンド API （14 ページ）を参照してください。ESC ポータルを使用したネットワークの追加と削除の詳細については、ESC ポータルを使用したリソースの管理 （381 ページ）を参照してください。

サブネットの管理

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

（注） サブネットは OpenStack でのみサポートされます。

ノースバウンド API を使用したサブネット定義の追加

次に、NETCONF を使用してサブネット定義を作成する例を示します。

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
 <target>
 <running/>
 </target>
 <config
 <esc_datamodel xmlns="http://www.cisco.com/esc/esc"
xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <networks>
 <network>
 <name>mgmt-net</name>
 <subnet>
 <name>mgmt-net-subnet</name>
 <ipversion>ipv4</ipversion>
 <dhcp>false</dhcp>
 <address>172.16.0.0</address>
 <gateway>172.16.0.1</gateway>
 <netmask>255.255.255.0</netmask>
 </subnet>
 </network>
 </networks>
 </esc_datamodel>
 </config> </edit-config
</rpc>
```

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

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

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

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

（注） NETCONF API を使用したサブネットの作成の詳細については、Cisco Elastic Services Controller API ガイド [英語] を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、REST ノースバウンド API （14 ページ）を参照してください。ESC ポータルを使用したネットワークの追加と削除の詳細については、ESC ポータルを使用したリソースの管理 （381 ページ）を参照してください。

フレーバの管理

フレーバは、RAM とディスクのサイズ、およびコアの数を定義します。

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

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

ノースバウンド API を使用したフレーバの追加

NETCONF のフレーバ作成要求：

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <flavors>
        <flavor>
            <name>test-flavor-indep</name>
              <vcpus>1</vcpus>
              <memory_mb>512</memory_mb>
              <root_disk_mb>0</root_disk_mb>
              <ephemeral_disk_mb>0</ephemeral_disk_mb>
              <swap_disk_mb>0</swap_disk_mb>
        </flavor>
    </flavors>
</esc_datamodel>
```

フレーバの作成に成功した場合の NETCONF 通知：

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

```
  <eventTime>2015-07-13T13:33:51.805+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_message>Flavor creation completed successfully.</status_message>
    <flavor>test-flavor-indep</flavor>
    <vm_source>
  </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>CREATE_FLAVOR</type>
    </event>
  </escEvent>
</notification>
```

（注） NETCONF API を使用したフレーバの作成と削除の詳細については、Cisco Elastic Services Controller API ガイド [英語] を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、REST ノースバウンド API （14 ページ）を参照してください。ESC ポータルを使用したフレーバの追加と削除の詳細については、ESC ポータルを使用したリソースの管理 （381 ページ）を参照してください。

イメージの管理

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

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

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

サンプル XML は次のとおりです。

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

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

ノースバウンド API を使用したイメージの追加

イメージを作成するための NETCONF 要求：

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <images>
        <image>
            <name>example-cirrosimage-indep</name>

<src>http://172.16.0.0:/share/images/esc_automated_test_images/cirros-0.3.3-x86_64-disk.img</src>

              <disk_format>qcow2</disk_format>
              <container_format>bare</container_format>
              <serial_console>true</serial_console>
              <disk_bus>virtio</disk_bus>
        </image>
    </images>
</esc_datamodel>
```

イメージが正常に作成された時の NETCONF 通知：

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-07-13T13:46:50.339+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_message>Image creation completed successfully.</status_message>
    <image>example-cirrosimage-indep</image>
    <vm_source>
  </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>CREATE_IMAGE</type>
    </event>
  </escEvent>
</notification>
```

（注） NETCONF API を使用したイメージの追加の詳細については、Cisco Elastic Services Controller API ガイド [英語] を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、REST ノースバウンド API （14 ページ）を参照してください。ESC ポータルを使用したイメージの追加と削除の詳細については、ESC ポータルを使用したリソースの管理（381 ページ）を参照してください。

ボリュームの管理

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

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

ESC によって作成されたボリューム

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

次の例は、展開要求で ESC ボリュームを作成する方法を示しています。

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

（注） 展開後にボリュームが追加された場合、OpenStack API では指定されたバスタグを指定できず、OpenStack インスタンスで定義されたデフォルトを使用します。

ESC によって作成されたブート可能ボリューム

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

次の例を参考にしてください。

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

アウトオブバンドボリューム

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

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

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

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

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

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

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

アウトオブバンドブート可能ボリューム

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

次の例を参考にしてください。

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

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

アウトオブバンドのブート可能ボリュームのスワップ

アウトオブバンドのブート可能ボリュームをスワップするには、更新展開リクエストで古いボリュームを削除し、同じ volid および `boot_index` 値を持つ新しいボリュームを追加する必要があります。追加することで、OpenStack のブート可能ボリュームがスワップされます。更新後に VM を再起動する必要があります。

たとえば、

```
<volumes>
  <volume nc:operation="delete">
    <name>pre-existing</name>
    <volid>0</volid>
    <bus>ide</bus>
    <type>lvm</type>
    <boot_index>0</boot_index>
  </volume>
  <volume>
    <name>another-pre-existing</name>
    <volid>0</volid>
    <bus>ide</bus>
    <type>lvm</type>
    <boot_index>0</boot_index>
  </volume>
</volumes>
```

パラメータの説明

- [名前（Name）]：既存のボリュームの表示名を指定します。
- [Volid]：ボリュームが接続される順序を指定します。これらは、各 VM グループの 0 または 1 から始まる連続した番号です。
- [バス（Bus）]：接続するボリュームのバスタイプを指定します。
- [タイプ（Type）]：（任意）<type> を指定すると、ESC は指定されたタイプのボリュームを照合します。
- [サイズ（Size）] および [サイズ単位（Sizeunits）]：ESC によって作成されたボリュームを定義します。
- boot_index ：（任意）ブート順序を指定します。VM をイメージからブートする場合と同様に、任意のボリュームからブートするには 0 に設定します。この設定を機能させるには、OpenStack でそのボリュームの「ブート可能」プロパティを true に設定する必要があります。

マルチアタッチボリューム

ボリュームを複数のホスト/サーバーに同時にアタッチする機能は、アクティブ/アクティブまたはアクティブ/スタンバイのシナリオ（OpenStack のみ）に必要なユースケースです。ボリュームを複数のサーバーインスタンスにアタッチするには、ボリュームの詳細で multiattach フラグを True に設定する必要があります。操作を実行する前に、適切なロールとポリシー設定があることを確認してください。

multiattach=<is> True の追加仕様機能の設定を含む、この特別なタイプを作成するには、次のコマンドを使用します。

```
$ cinder type-create multiattach
$ cinder type-key multiattach set multiattach="<is> True"
```

type-key の名前は自由に指定できますが、参照するプロパティは multiattach にする必要があります。このガイドでは、タイプを multiattach として参照します。

このタイプが作成されたら、タイプを指定して、OpenStack にアウトオブバンドボリューム（ブート可能またはそれ以外）を作成します。次に例を示します。

```
$ cinder create <volume_size> --name <volume_name> --volume-type multiattach
```

このボリュームを使用するには、展開を作成するときに、このボリュームをアウトオブバンドボリュームと同じように扱います。ただし、複数の VM に対してボリューム UUID または一意の名前を指定できる点が異なります。ESC は、正しく入力されたボリュームのみを複数の VM に接続しようとします。次に例を示します。

```
<vm_group>
        <name>c1</name>
        ...
        <volumes>
         <volume>
            <name>cf-cdr0-volume</name>
            <volid>0</volid>
         </volume>
        </volumes>
        ...
  </vm_group>
  <vm_group>
       <name>c2</name>
       ...
       <volumes>
        <volume>
            <name>cf-cdr0-volume</name>
            <volid>0</volid>
       </volume>
      </volumes>
      ...
</vm_group>
```

マルチアタッチボリュームは、通常のアウトオブバンドボリュームと同様にデタッチでき、サービス更新を使用して VM 上の通常のアウトオブバンドボリュームを置き換えるためにも使用できます。このアクションの後は、新しくアタッチされたボリューム（マルチアタッチまたはその他）を認識するために VM を再起動する必要があります。

（注）

OpenStack の要件

- マルチアタッチ対応ボリュームを複数のサーバーにアタッチするために最低限必要な Compute API マイクロバージョンは 2.60 です。
- Cinder 12.0.0（Queens）または最新版（マイクロバージョン 3.50 以降）が必要です。
- nova-compute サービスは少なくとも Queens リリースレベルコード（17.0.0）で実行している必要があり、ハイパーバイザドライバは複数のゲストに対するブロックストレージデバイスの接続をサポートしている必要があります。ボリュームのマルチアタッチをサポートするコンピューティングドライバの詳細については、機能サポートマトリックスを参照してください。
- libvirt コンピューティングドライバを使用している場合、以下のネイティブ パッケージ バージョンによってマルチアタッチのサポートが決まります。
- libvirt は 3.10 以上である必要があります。
- Qemu は 2.10 未満である必要があります。
- 使用中のマルチアタッチボリュームのスワップはサポートされていません（これは、実際にはブロック ストレージ ボリュームの retype API を介して制御されます）。

テナントボリューム API

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

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

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

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

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

テナントボリューム API によって作成されたボリュームによる展開

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

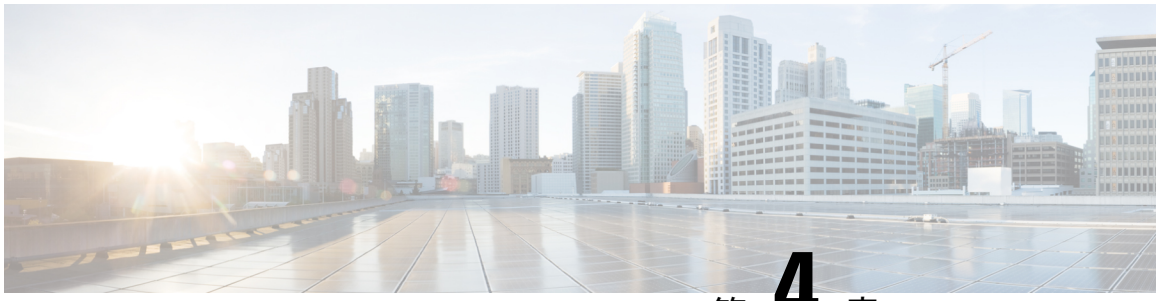
テナントボリューム API を使用して作成されたボリュームによる展開要求の例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <tenants>
        <tenant>
            <name>admin</name>
            <deployments>
                <deployment>
                    <name>admin-with-volume</name>
                    <vm_group>
                        <name>cirros</name>
                        <bootup_time>60</bootup_time>
                        <recovery_wait_time>0</recovery_wait_time>
                        <image>Automation-Cirros-Image</image>
                        <flavor>Automation-Cirros-Flavor</flavor>
                        <volumes>
                            <volume>
                                <name>some-volume</name>
                                <volid>1</volid>
                                <bus>ide</bus>
                            </volume>
                        </volumes>
                        <interfaces>
                            <interface>
                                <nicid>0</nicid>
                                <network>mynetwork</network>
                            </interface>
                        </interfaces>
                        <scaling>
                            <min_active>1</min_active>
                            <max_active>1</max_active>
```

```
                        <elastic>true</elastic>
                    </scaling>
                    <kpi_data>
                        <kpi>
                            <event_name>VM_ALIVE</event_name>
                            <metric_value>1</metric_value>
                            <metric_cond>GT</metric_cond>
                            <metric_type>UINT32</metric_type>
                            <metric_collector>
                                <type>ICMPPing</type>
                                <nicid>0</nicid>
                                <poll_frequency>3</poll_frequency>
                                <polling_unit>seconds</polling_unit>
                                <continuous_alarm>false</continuous_alarm>
                            </metric_collector>
                        </kpi>
                    </kpi_data>
                    <rules>
                        <admin_rules>
                            <rule>
                                <event_name>VM_ALIVE</event_name>
                                <action>"ALWAYS log"</action>
                                <action>"TRUE
                                  servicebooted.sh"</action>
                                <action>"FALSE recover
                                  autohealing"</action>
                            </rule>
                        </admin_rules>
                    </rules>
                    <config_data />
                </vm_group>
            </deployment>
        </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```

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

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



第 4 章

VMware vCenter のリソースの管理

ここでは、次の内容について説明します。

VMware vCenter でのイメージの追加

VMware vCenter に VNF を展開する場合は、VMware vCenter ですでに使用可能なアウトオブバンドイメージを使用するか、ESC ポータルで、あるいは REST API または NETCONF API を使用してイメージを作成できます。展開属性の詳細については、「Cisco Elastic Services Controller Deployment Attributes」を参照してください。

ノースバウンド API を使用したイメージの追加

（注） VMware vCenter に VNF を展開する場合は、VMware vCenter ですでに使用可能なアウトオブバンドイメージを使用するか、ESC ポータルで、あるいは REST API または NETCONF API を使用してイメージを作成できます。

イメージを作成するための NETCONF 要求：

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <images>
        <image>
            <name>cirrosimage-indep</name>

<src>http://172.16.0.0:/share/images/esc_automated_test_images/cirros-0.3.3-x86_64-disk.img</src>

              <disk_format>qcow2</disk_format>
              <container_format>bare</container_format>
              <serial_console>true</serial_console>
              <disk_bus>virtio</disk_bus>
        </image>
```

```
    </images>
</esc_datamodel>
```

イメージが正常に作成された時の NETCONF 通知：

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-07-13T13:46:50.339+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_message>Image creation completed successfully.</status_message>
    <image>cirrosimage-indep</image>
    <vm_source>
  </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>CREATE_IMAGE</type>
    </event>
  </escEvent>
</notification>
```

（注） NETCONF API を使用したイメージの追加の詳細については、Cisco Elastic Services Controller API ガイド [英語] を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、REST ノースバウンド API （14 ページ）を参照してください。ESC ポータルを使用したイメージの追加と削除の詳細については、ESC ポータルを使用した VMware vCenter リソースの管理 （384 ページ）を参照してください。

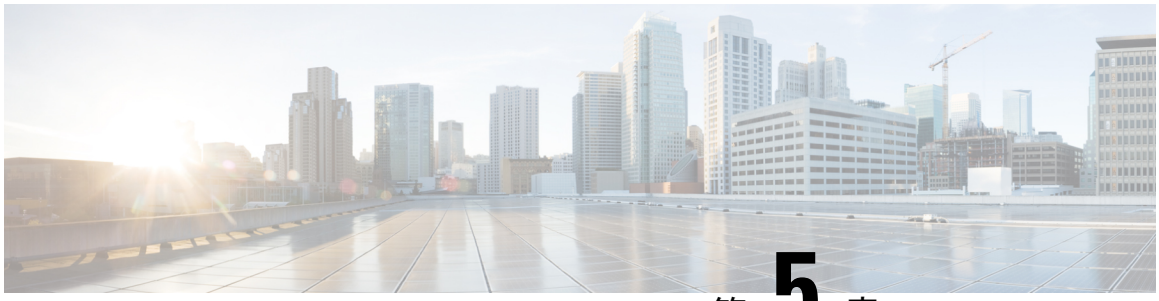
VMware vCenter での分散ポートの作成

VMware vCenter では、VM カーネルまたは仮想マシンのネットワークアダプタに接続する vSphere 分散スイッチに分散ポートを設定します。これにより、vSphere 分散スイッチの各メンバーポートのポート設定オプションを指定します。分散ポートグループは、ネットワークへの接続方法を定義します。REST インターフェイスを使用して、分散ポートグループを作成できます。

次に、REST API を使用して分散ポートグループ（VMware vCenter のみ）を作成する例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <name>network-portgroup-01</name>

    <switch_name>vdSwitch-01</switch_name>

    <vlan_id>0</vlan_id>

    <number_of_ports>8</number_of_ports>
</network>
```

（注） VMware vCenter では、ESC は vSphere 分散スイッチ（VDS）内での基本的なポートグループまたはネットワークの作成のみをサポートします。高度な VDS 設定では、アウトオブバンド設定のみが ESC でサポートされます。



第 5 章

vCloud Director のリソースの管理

vCloud Director（vCD）のリソースの管理

テンプレート、カタログ、ネットワークなど、すべての vCD リソースは、アウトオブバンド（OOB）で管理されます。vCD での VM の展開については、VMware vCloud Director（vCD）での仮想ネットワーク機能の展開 （152 ページ）を参照してください。

組織（Organization）

組織は、ユーザ、グループ、およびコンピューティングリソースのグループです。組織には、その組織が作成するvAppテンプレートと、vAppの作成に使用されるリソースが含まれます。クラウドには、1 つ以上の組織を含めることができます。

オーガニゼーション VDC

組織仮想データセンター（組織 VDC）は、仮想システムの導入環境であり、展開前に作成する必要があります。組織 VDC には、組織、およびネットワーク、ストレージ、CPU、メモリなどのリソースの割り当てメカニズムが含まれます。十分なメモリと CPU 容量、およびストレージスペース（ストレージプロファイル）が必要です。

カタログ（Catalogs）

カタログには、vAppテンプレートおよびメディアイメージへの参照が含まれます。vAppテンプレートが配置されているカタログには、展開に使用される組織ユーザの読み取りおよび書き込み権限が必要です。ESC で デイゼロ設定用の ISO ファイルを作成またはアップロードする必要がある場合は、書き込み権限が必要です。

ネットワーク（Network）

vApp の場合、ネットワークには 2 つのレベルがあります。

- vApp 内の VM 間の通信用の vApp 内のネットワーク。
- vApp 全体の VM 間の通信用の vDC 内のネットワーク。

ESC は vCenter に展開されるため、vCD の一部ではありません。ESC で VM ステータスをモニタする場合、外部ネットワークに直接または間接的に接続されている組織 VDC ネットワークまたは vApp ネットワークに接続する少なくとも 1 つのネットワーク インターフェイスが各 VM に必要です。

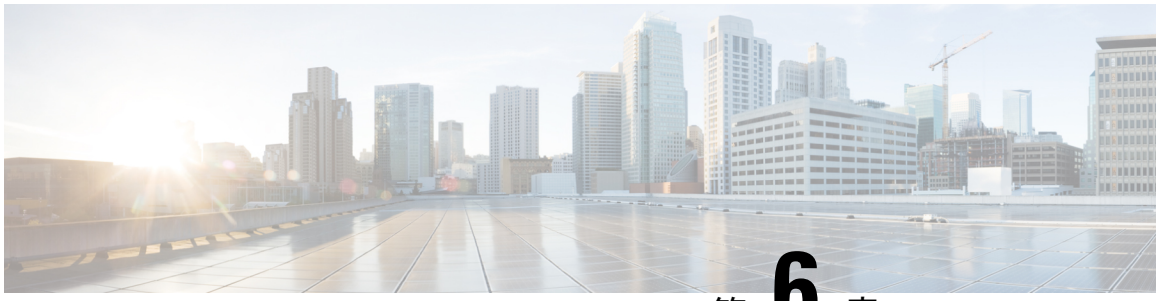
展開ストレージプロファイル

ストレージプロファイルは、展開要求で指定されます。

（注） 展開ストレージプロファイルは、VMware vSpere の下でデータストアを指定する方法です。これは、VM のボリュームまたはディスクとは異なります。

例：

```
<volumes>
    <volume>
        <name>{Storage profile name}</name>
        <volid>1</volid>
    </volume>
</volumes>
```



第 6 章

ESC リソースの管理

VIM コネクタの管理

VIM コネクタには、URL や認証クレデンシャルなどの詳細が含まれており、ESC が VIM に接続して通信できるようにします。VIM コネクタが設定されている場合、ESC は複数の VIM に接続します。VIM コネクタとそのクレデンシャルは、次の 2 つの方法で設定できます。

- bootvm.py パラメータを使用したインストール時：bootvm.py を使用して 1 つの VIM コネクタのみを設定でき、これがデフォルトのVIMコネクタになります。
- VIM コネクタ API を使用：VIM コネクタ API を使用すると、複数の VIM コネクタを追加できます。デフォルトの VIM コネクタ（bootvm.py パラメータを使用して設定されていない場合）と追加の VIM コネクタを設定できます。

デフォルトの VIM コネクタは、ESC をデフォルト VIM に接続します。マルチ VIM 展開の各 VIM は VIM コネクタで設定されます。これらの VIM はデフォルト以外の VIM です。ESC は、デフォルトの VIM でリソースを作成および管理します。デフォルト以外の VIM では、展開のみがサポートされます。

単一の VIM 展開では、単一の設定済み VIM コネクタがデフォルトの VIM コネクタになります。マルチ VIM 展開の場合は、複数のコネクタを追加し、デフォルトの VIM コネクタ API を使用して 1 つのコネクタをデフォルトとして指定する必要があります。詳細については、複数の OpenStack VIM への VNF の展開 （119 ページ）を参照してください。

（注） ESCは、次の条件が満たされた場合にのみ、リソースまたは展開を作成、更新、または削除するノースバウンド設定要求を受け入れます。

- ESC にターゲット VIM（単数/複数）があり、対応する VIM ユーザが設定されていること。
- ESC がターゲット VIM（単数/複数）に到達できること。
- ESC が VIM ユーザを認証できること。

VIM コネクタの設定

VIM コネクタは、インストール中またはインストール後に設定できます。

インストール中の VIM コネクタの設定

インストール中にVIMコネクタを設定するには、次のパラメータをbootvm.pyに指定する必要があります。

環境変数	bootvm.py 引数
OS_TENANT_NAME	--os_tenant_name
OS_USERNAME	--os_username
OS_PASSWORD	--os_password
OS_AUTH_URL	--os_auth_url

インストール後の VIM コネクタの設定

インストール後にVIMコネクタを設定するには、次のパラメータをbootvm.pyに指定する必要があります。

```
--no_vim_credentials
```

no_vim_credentials パラメータが指定されている場合、次の bootvm.py 引数は無視されます。

- os_tenant_name
- os_username
- os_password
- os_auth_url

インストールの詳細については、Cisco Elastic Services Controller インストールおよびアップグレードガイド [英語] を参照してください。インストール後に VIM コネクタ API を使用して同じ設定を行うことができます。詳細については、VIM コネクタ API を使用した VIM コネクタの管理 （54 ページ）を参照してください。

デフォルトの VIM コネクタ

デフォルトの VIM コネクタ API を使用すると、展開で複数のコネクタを使用できる場合にデフォルトの VIM コネクタを指定できます。

単一の VIM 展開の場合、ESC は単一の VIM コネクタをサポートします。この単一の VIM コネクタがデフォルトの VIM コネクタになります。ESC は、マルチ VIM 展開用に複数の VIM コネクタをサポートします。新しいロケータ属性を使用して、デフォルトの VIM コネクタを設定できます。展開およびリソースの作成に ESC リリース 2.x データモデルを使用している場合は、ESC でデフォルトの VIM コネクタを明示的に設定します。

ロケータ属性は、デフォルト以外の VIM に VM を展開するためのデータモデルに導入されています。詳細については、複数の OpenStack VIM への VNF の展開 （119 ページ）を参照してください。

展開中に VIM コネクタが使用可能であるが、デフォルトのコネクタがまだ設定されていない場合は、ロケータ属性を指定する必要があります。指定しない場合、要求は拒否されます。

デフォルトの VIM コネクタが設定されていない場合、ESC リリース 3.0 より前のデータモデルは使用できません。ESC リリース 2.x から ESC リリース 3.0 以降にアップグレードすると、既存の VIM コネクタがデフォルトの VIM コネクタとしてプロビジョニングされます。

（注） デフォルトの VIM コネクタは、設定後に別のコネクタに変更したり、削除したりできません。

デフォルトのコネクタは、データモデルの最上位（または先頭）で指定する必要があります。次に、データモデルを示します。

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

REST API を使用してデフォルトの VIM コネクタを追加するには、次のようにします。

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

インストール時に VIM コネクタを追加するには、VIM コネクタの設定 （52 ページ）の「インストール時の VIM コネクタの設定」を参照してください。VIM コネクタを使用すると、複数の VIM を ESC に接続できます。マルチ VIM 展開の詳細については、複数の OpenStack VIM への VNF の展開 （119 ページ）を参照してください。

VIM コネクタの削除

デフォルトの VIM コネクタが作成および設定されると、ESC は SystemAdminTenant を自動的に作成します。SystemAdminTenant は削除できません。VIM が接続され、VIM ユーザがシステム管理者テナントに認証されます。したがって、デフォルトの VIM は削除も更新もできません。ただし、VIM ユーザとそのプロパティは削除または更新できます。ESC から VIM 上に作成されたリソースが存在しない場合は、デフォルト以外の VIM コネクタを更新および削除できます。ESC を介して VIM にリソースが作成されている場合は、最初にリソースを削除してから、VIM ユーザを削除して VIM コネクタを削除する必要があります。

VIM コネクタ API を使用した VIM コネクタの管理

VIM ログイン情報を渡さずに ESC を展開した場合は、VIM コネクタと VIM ユーザ API（REST または NETCONF API）を使用して、ESC から VIM ログイン情報を設定できます。インストール時にデフォルトの VIM コネクタを設定している場合でも、VIM コネクタ API を使用して追加の VIM コネクタを設定できます。

NETCONF API を使用した管理

- NETCONF を使用した VIM ログイン情報の提供：

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

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

- NETCONF を使用した VIM コネクタの更新：

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <vim_connector nc:operation="replace">
      <id>example_vim</id>
      <type>OPENSTACK</type>
      <properties>
        <property>
          <name>os_auth_url</name>
          <value>{auth_url}</value>
        </property>
        <property>
          <name>os_project_name</name>
          <value>vimProject</value>
        </property>
        <!-- The project domain name is only needed for openstack v3 identity api
-->
        <property>
          <name>os_project_domain_name</name>
          <value>default</value>
        </property>
        <property>
          <name>os_identity_api_version</name>
          <value>3</value>
        </property>
      </properties>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

- Netconf を使用した VIM ユーザの更新：

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

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

- Netconf を使用した VIM コネクタの削除：

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

- NETCONF を使用した VIM ユーザの削除：

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

- コマンドを使用した VIM コネクタの削除：

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

- コマンドを使用した VIM ユーザの削除：

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

REST API を使用した管理

- REST を使用した VIM の追加：

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

<?xml version="1.0"?>
<vim_connector xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <id>example_vim</id>
```

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

- REST を使用した VIM ユーザの追加：

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

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

- REST を使用した VIM の更新：

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

<?xml version="1.0"?>
<vim_connector xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <!--unique id for each vim-->
  <id>example_vim</id>
  <type>OPENSTACK</type>
  <properties>
    <property>
      <name>os_auth_url</name>
      <value>{auth_url}</value>
    </property>
    <property>
      <name>os_project_name</name>
      <value>vimProject</value>
```

```
    </property>
    <!-- The project domain name is only needed for openstack v3 identity api -->
    <property>
      <name>os_project_domain_name</name>
      <value>default</value>
    </property>
    <property>
      <name>os_identity_api_version</name>
      <value>3</value>
    </property>
 </properties>
</vim_connector>
```

- REST を使用した VIM ユーザの更新：

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

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

- REST を使用した VIM の削除：

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

- REST を使用した VIM ユーザの削除：

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

- 各 VIM または VIM ユーザの設定が完了した後の通知の例：

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

API の詳細については、Cisco Elastic Services Controller API ガイド [英語] を参照してください。

特記事項：

- 複数の VIM コネクタを追加できますが、すべての VIM コネクタが同じ VIM タイプである必要があります。OpenStack VIM にのみ複数の VIM コネクタを追加できます。ただし、VIM コネクタごとに設定できる VIM ユーザは 1 人だけです。
- os_project_name プロパティと os_project_domain_name プロパティでは、VIM コネクタのプロパティの下で認証および承認を得るための OpenStackプロジェクトの詳細を指定します。os_tenant_name プロパティは、VIM ユーザの下に存在する場合は無視されます。
- VIM コネクタのプロパティ os_auth_url と os_project_name、および VIM ユーザプロパティ os_password は、OpenStack VIM の必須プロパティです。これらのプロパティが指定されていない場合、VIM コネクタの作成要求は拒否されます。
- VIM のユーザ名とパスワードはいつでも更新できます。ESC で作成されたリソースが存在する間は、VIM エンドポイントは更新できません。
- VIM プロパティまたは VIM ユーザログイン情報プロパティの名前は大文字と小文字が区別されません。たとえば、OS_AUTH_URL と os_auth_url は ESC にとっては同じです。

VIM コネクタのログイン情報を暗号化するには、既存の <value> フィールドを <encrypted_value> で置き換えます。

次の例を参考にしてください。

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

これにより、/opt/cisco/esc/esc_database_esc_confd.conf に含まれるキーを使用して、os_value パスワードが aes-cfb-128-encrypted-string として CFB に保存されます。

（注） 既存の値は、指定されたログイン情報内でのみ暗号化された値に置換される必要があります。

詳細については、「設定データの暗号化」を参照してください。

VIM コネクタのステータス API

次の表に、VIM コネクタのステータスと各 VIM コネクタのステータスメッセージを示します。ステータスには、VIM の ESC 接続および認証ステータスが表示されます。

VIM 到達可能性	ユーザ認証	ステータス（ESC による）	ステータスメッセージ
到達不能	-	CONNECTION_FAILED	VIM接続を確立できません
到達可能	VIMユーザが設定されていません	NO_CREDENTIALS	VIMユーザログイン情報が見つかりません
到達可能	認証に失敗しました	AUTHENTICATION_FAILED	VIM認証に失敗しました
到達可能	認証の成功	CONNECTION_SUCCESSFUL	VIMに正常に接続しました

REST API を使用したステータス

HTTP 操作：GET

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

サンプルの REST 応答は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vim_connector xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <properties>
        <property>
            <name>os_auth_url</name>
            <value>http://172.16.0.0:5000/v2.0/</value>
        </property>
    </properties>
    <id>default_openstack_vim</id>
    <status>CONNECTION_SUCCESSFUL</status>
    <status_message>Successfully connected to VIM</status_message>
    <type>OPENSTACK</type>
</vim_connector>
```

NETCONF API を使用したステータス

opdata にステータスが表示されます。VIM コネクタステータスは、VIM コネクタコンテナ内にあります。

サンプルの opdata は次のとおりです。

```
<system_config>
    <active_vim>OPENSTACK</active_vim>
    <openstack_config>
        <os_auth_url>http://172.16.0.0:5000/v2.0/</os_auth_url>
        <admin_role>admin</admin_role>
        <os_tenant_name>admin</os_tenant_name>
        <os_username>admin</os_username>
        <member_role>_member_</member_role>
    </openstack_config>
    <vim_connectors>
        <vim_connector>
            <id>my-server</id>
            <status>CONNECTION_FAILED</status>
            <status_message>Unable to establish VIM connection</status_message>
        </vim_connector>
```

```
        <vim_connector>
            <id>Openstack-Liberty</id>
            <status>NO_CREDENTIALS</status>
            <status_message>No VIM user credentials found</status_message>
        </vim_connector>
    </vim_connectors>
</system_config>
```

VIM コネクタ操作のステータス

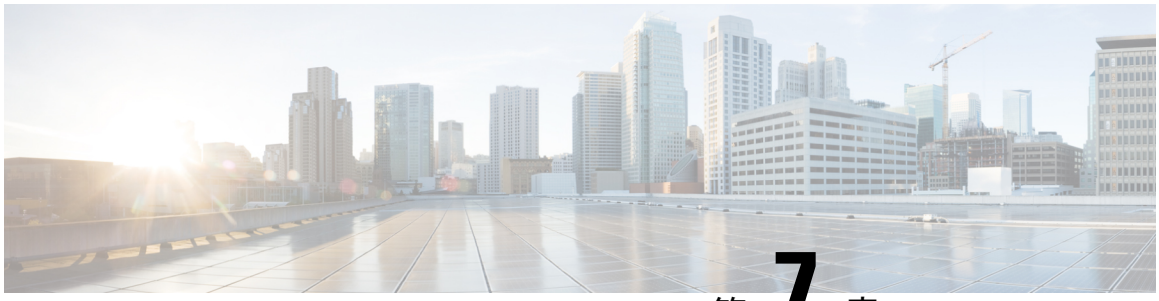
VIM _CONNECTION_STATE 通知は、REST および NETCONF を介して ESC に追加された各 VIM コネクタおよびユーザのステータスを通知します。VIM コネクタの詳細については、VIM コネクタの管理 （51 ページ）を参照してください。

通知には次の内容が表示されます。

- イベントタイプ：VIM _CONNECTION_STATE
- ステータス：成功または失敗
- ステータス メッセージ
- vim_connector_id

VIM コネクタのモニタリング、VIM ユーザの追加または削除、および VIM コネクタの更新に関する通知が送信されます。成功と失敗の通知の例を次に示します。

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


<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2017-06-27T14:51:55.862+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>0</status_code>
    <status_message>VIM Connection State Up</status_message>
    <vim_connector_id>my-server</vim_connector_id>
    <event>
      <type>VIM_CONNECTION_STATE</type>
    </event>
  </escEvent>
</notification>
```



第 7 章

VIM コネクタの設定

OpenStack の VIM コネクタの設定

OpenStack 固有の操作用に VIM コネクタを設定できます。

（注） VIM コネクタを設定するには、 VIM コネクタの設定 （52 ページ）を参照してください。

OpenStack での ESC ユーザの非管理者ロールの作成

デフォルトでは、OpenStack では ESC ユーザに管理者ロールが割り当てられます。一部のポリシーでは、特定の ESC 操作に対してデフォルトの管理者ロールの使用が制限される場合があります。ESC リリース 3.1 以降では、OpenStack で ESC ユーザの権限を制限した非管理者ロールを作成できます。

非管理者ロールを作成するには、次の手順を実行します。

1. OpenStack で非管理者ロールを作成します。
2. ESC ユーザに非管理者ロールを割り当てます。

   OpenStack Horizon（アイデンティティ）で、または OpenStack コマンド ライン インターフェイスを使用して、ESC ユーザロールを割り当てる必要があります。詳細については、OpenStack のマニュアルを参照してください。

   ロール名は OpenStack でカスタマイズできます。デフォルトでは、OpenStack のすべての非管理者ロールに同じレベルの権限が付与されます。
3. 管理者以外のロールに必要な権限を付与します。

policy.json ファイルを変更して、必要な権限を付与する必要があります。

（注） ESC ユーザロールを動作させるには、policy.json ファイルの create_port: fixed_ips および create_port: mac_address パラメータに権限を付与する必要があります。

次の表に、必要な権限を取得後に非管理者ロールで実行できる ESC 操作を示します。

表 *3 : ESC* 操作用の非管理者ロール権限

ESC VIM 操作	説明	権限	コメント
プロジェクトの作成	OpenStack プロジェクトを作成する	/etc/keystone/policy.json "identity:create_project" "identity:create_grant"	ESC 管理対象 OpenStack プロジェクトの場合は、 ロールに identity: create_grant が必要なプロジェクトにユーザを追加します。
プロジェクトの削除	OpenStack プロジェクトを削除する	/etc/keystone/policy.json "identity:delete_project"	
イメージのクエリ	すべてのイメージのリストを取得する	不要	所有者（ターゲットプロジェクトのユーザ）がクエリを実行できます。 パブリックイメージまたは共有イメージを取得できます。
イメージの作成（Create Image）	パブリックイメージを作成する	/etc/glance/policy.json "publicize_image"	デフォルトでは、管理者はパブリックイメージを作成できます。 イメージの公開はポリシーによって保護されます。
	プライベートイメージを作成する	不要	次の文を使用してプライベートイメージを作成できます。 <image> <name>mk-test-image</name> ... <disk_bus>virtio</disk_bus> <visibility>private</visibility> </image>
イメージの削除	イメージを削除する	不要	所有者はイメージを削除できます。

ESC VIM 操作	説明	権限	コメント
フレーバーのクエリ	既存のフレーバーをクエリする	不要	所有者はフレーバーをクエリできます。 パブリックフレーバーもクエリできます。
フレーバーの作成	新しいフレーバーを作成する	/etc/nova/policy.json "os_compute_api:os-flavor-manage"	フレーバーの管理は通常、 クラウドの管理者だけができます。
フレーバーの削除	フレーバーを削除する	/etc/nova/policy.json "os_compute_api:os-flavor-manage"	
ネットワークのクエリ	ネットワークのリストを取得する	/etc/neutron/policy.json "get_network"	所有者は、共有ネットワークを含むネットワークのリストを取得できます。
ネットワークの作成	通常のネットワークを作成する	不要	
	特殊なケースでネットワークを作成する	/etc/neutron/policy.json "create_network:provider:physical_network" "create_network:provider:network_type" "create_network:provider:segmentation_id" "create_network:shared"	これらのルールは、 physical_network（SR-IOV など）、network_type（SR-IOV など）、または segmentation_id（3008 など）を使用してネットワークを作成するか、あるいはネットワークを共有用に設定する場合に必要です。 <network> <name>provider-network</name> <!-- <shared>false</shared> //default is true--> <admin_state>true</admin_state> <provider_physical_network>VAR_PHYSICAL_NET</provider_physical_network> <provider_network_type>vlan</provider_network_type> <provider_segmentation_id>2330</provider_segmentation_id> ... </network>

ESC VIM 操作	説明	権限	コメント
ネットワークの削除	ネットワークを削除する	不要	所有者はネットワークを削除できます。
サブネットのクエリ	サブネットのリストを取得する	/etc/neutron/policy.json "get_subnet"	ネットワークの所有者はサブネットのリストを取得できます。 共有ネットワークからサブネットのリストを取得することもできます。 <network> <name>esc-created-network</name> <!--network must be created by ESC--> <admin_state>false</admin_state> <subnet> <name>makulandyescextnet1-subnet1</name> <ipversion>ipv4</ipversion> <dhcp>true</dhcp> <address>10.6.0.0</address> <netmask>255.255.0.0</netmask> </subnet> </network>
サブネットの作成	サブネットを作成する	不要	ネットワークの所有者はサブネットを作成できます。
サブネットの削除	サブネットを削除する	不要	ネットワークの所有者はサブネットを削除できます。
ポートのクエリ	既存のポートを取得する	不要	所有者はポートのリストを取得できます。

ESC VIM 操作	説明	権限	コメント
ポートの作成	DHCP を使用してネットワークインターフェイスを作成する	不要	
	MAC アドレスを使用してネットワークインターフェイスを作成する	/etc/neutron/policy.json "create_port:mac_address"	<interfaces> <interface> <nic id>0</nic id> <mac_address>fa:16:3e:73:19:b5</mac_address> <network>esc-net</network> </interface> </interfaces> VM リカバリにもこの権限が必要です。
	固定 IP または共有 IP を使用してネットワークインターフェイスを作成する	/etc/neutron/policy.json "create_port:fixed_ips"	<subnet> <name>IP-pool-subnet</name> <ipversion>ipv4</ipversion> <dhcp>false</dhcp> <address>172.16.0.0</address> <netmask>255.255.255.0</netmask> <gateway>172.16.0.1</gateway> </subnet><shared_ip> <nic id>0</nic id> <static>false</static> </shared_ip> VM リカバリにもこの権限が必要です。

ESC VIM 操作	説明	権限	コメント
ポートの更新	ポートデバイスの所有者の更新	不要	所有者はポートを更新できます。
	アドレスペアを許可するようにポートを更新する	/etc/neutron/policy.json "update_port:allowed_address_pairs"	<interface> <nicid>0</nicid> <network>VAR_MANAGEMENT_NETWORK_ID</network> <allowed_address_pairs> <network> <name>VAR_MANAGEMENT_NETWORK_ID</name> </network> <address> <ip_address>172.16.0.0</ip_address> <netmask>255.255.0.0 </netmask> </address> <address> <ip_address>172.16.6.1</ip_address> <ip_prefix>24 </ip_prefix> </address> </allowed_address_pairs> </interface>
ポートの削除（Delete Port）	ポートを削除する	不要	所有者はポートを削除できます。
ボリュームのクエリ	ボリュームのリストを取得する	不要	所有者はボリュームのリストを取得できます。
ボリュームの作成	ボリュームを作成する	不要	
ボリュームの削除	ボリュームを削除する	不要	所有者はボリュームを削除できます。
VM のクエリ	プロジェクト内のすべての VM を取得する	不要	所有者は、プロジェクト内のすべての VM のリストを取得できます。

ESC VIM 操作	説明	権限	コメント
VM の作成	VM を作成する	不要	
	ホストのターゲット展開で VM を作成する	/etc/nova/policy.json "os_compute_api:servers:create:forced_host"	<placement> <type>zone_host</type> <enforcement>strict</enforcement> <host>anyHOST</host> </placement>
	ゾーンのターゲット展開で VM を作成する	不要	
	同じホストに VM を作成する アフィニティ/アンチアフィニティ	不要	
	サーバグループに VM を作成する アフィニティ/アンチアフィニティ	不要	このサポートは、グループ内アンチアフィニティ専用です。
[VM の削除（Delete VM）]	VM を削除する	不要	所有者は VM を削除できます。

OpenStack でのリソース管理の詳細については、OpenStack のリソースの管理 （21 ページ）を参照してください。

OpenStack エンドポイントの上書き

デフォルトでは、ESCは認証に成功後、OpenStackが提供するエンドポイントのカタログリターンオプションを使用します。ESC はこれらのエンドポイントを使用して、OpenStack のさまざまな API と通信します。エンドポイントは正しく設定されていないこともあります。たとえば、OpenStack インスタンスは認証に KeyStone V3 を使用するように設定されているのに、OpenStack から返されるエンドポイントが KeyStone V2 用の場合があります。この問題は、OpenStack エンドポイントを上書きすることで解決できます。

VIM コネクタの設定中にOpenStackエンドポイントを上書き（設定）できます。これは、インストール時に bootvm.py パラメータおよび VIM コネクタ API を使用して実行できます。

次の OpenStack エンドポイントは、VIM コネクタの設定を使用して設定できます。

- OS_IDENTITY_OVERWRITE_ENDPOINT
- OS_COMPUTE_OVERWRITE_ENDPOINT
- OS_NETWORK_OVERWRITE_ENDPOINT
- OS_IMAGE_OVERWRITE_ENDPOINT
- OS_VOLUME_OVERWRITE_ENDPOINT

インストール時にOpenStackエンドポイントを上書きする場合、ユーザはESC設定パラメータファイルを作成し、ESC VM の展開中にこのファイルを引数として bootvm.py に渡します。

次に、param.conf ファイルの例を示します。

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

インストール時の VIM コネクタの設定の詳細については、VIM コネクタの設定 （52ページ）を参照してください。

VIM コネクタ API（REST と NETCONF の両方）を使用して、デフォルト以外の VIM コネクタの OpenStack エンドポイントを上書き（設定）するには、新しい VIM コネクタの作成時または既存の VIM コネクタの更新時に、上書きするエンドポイントを VIM コネクタのプロパティとして追加します。

各 VIM コネクタには、独自の上書きエンドポイントを設定できます。デフォルトの上書きエンドポイントはありません。

次の例では、os_identity_overwrite_endpoint および os_network_overwrite_endpoint プロパティを追加して、エンドポイントを上書きします。

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <!--represents a vim-->
    <vim_connector>
      <id>default_openstack_vim</id>
      <type>OPENSTACK</type>
      <properties>
        <property>
          <name>os_auth_url</name>
          <value>http://172.16.0.0:35357/v3</value>
        </property>
        <property>
```

```
            <name>os_project_domain_name</name>
            <value>default</value>
          </property>
          <property>
            <name>os_project_name</name>
            <value>admin</value>
          </property>
          <property>
            <name>os_identity_overwrite_endpoint</name>
            <value>http://some_server:some_port/</value>
          </property>
          <property>
            <name>os_network_overwrite_endpoint</name>
            <value>http://some_other_server:some_other_port/</value>
          </property>
        </properties>
      </vim_connector>
    </vim_connectors>
</esc_system_config>
```

AWS の VIM コネクタ設定

VIM コネクタと VIM ユーザ API を使用して、AWS 展開の VIM クレデンシャルを設定できます。

（注） AWS 展開では、デフォルトの VIM コネクタはサポートされていません。

VIM コネクタの **aws_default_region** 値は認証を提供し、VIM ステータスを更新します。認証後にデフォルトリージョンを変更することはできません。

VIM コネクタの設定

AWS 展開用の VIM コネクタを設定するには、AWS クレデンシャルから AWS_ACCESS_ID、AWS_SECRET_KEY を指定します。

```
[admin@localhost ~]# esc_nc_cli --user <username> --password <password> edit-config
aws-vim-connector-example.xml
```

（注） 既存の VIM コネクタ設定を編集するには、必要な変更を加えた後、同じコマンドを使用します。

AWS VIM コネクタの例は次のとおりです。

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

```
            </property>
         </properties>
         <users>
            <user>
               <id>AWS_ACCESS_ID</id>
               <credentials>
                  <properties>
                     <property>
                        <name>aws_secret_key</name>
                        <encrypted_value>AWS_SECRET_KEY</encrypted_value>
                     </property>
                  </properties>
               </credentials>
            </user>
         </users>
      </vim_connector>
   </vim_connectors>
</esc_system_config>
```

VIM コネクタの削除

既存の VIM コネクタを削除するには、最初に展開、VIM ユーザ、次に VIM コネクタを削除する必要があります。

```
[admin@localhost ~]# esc_nc_cli --user <username> --password <password> delete-vimuser
AWS_EAST_2 AWS_ACCESS_ID
```

```
[admin@localhost ~]# esc_nc_cli --user <username> --password <password> delete-vimconnector
AWS_EAST_2
```

（注） 同じ VIM タイプに対して複数の VIM コネクタを設定できます。

AWS 展開用の VIM コネクタは、VIM コネクタの API を使用して設定する必要があります。

ESC は、VIM コネクタごとに 1 人の VIM ユーザをサポートします。

VIM コネクタとそのプロパティは、展開後に更新できません。

AWS での VNF の展開については、単一または複数の AWS リージョンでの VNF の展開 （158 ページ）を参照してください。

VMware vCloud Director（vCD）の VIM コネクタの設定

vCD 組織に接続するには、VIM コネクタを設定する必要があります。組織および組織ユーザは、VMware vCD で事前設定する必要があります。展開データモデルについては、「VMware vCloud Director（vCD）での仮想ネットワーク機能の展開」を参照してください。

VIM コネクタの詳細は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
   <vim_connectors>
```

```
      <vim_connector>
         <id>vcd_vim</id>
         <type>VMWARE_VCD</type>
         <properties>
            <property>
               <name>authUrl</name>
                <!-- vCD is the vCD server IP or host name -->
               <value>https://vCD</value>
            </property>
         </properties>
         <users>
            <user>
              <!-- the user id here represents {org username}@{org name} -->
               <id>user@organization</id>
               <credentials>
                  <properties>
                     <property>
                        <name>password</name>
                          <!—the organization user’s password-->
                        <value>put user’s password here</value>
                     </property>
                  </properties>
               </credentials>
            </user>
         </users>
      </vim_connector>
   </vim_connectors>
</esc_system_config>
```

VMware vSphere の VIM コネクタの設定

vSphere 組織に接続するには、VIM コネクタを設定する必要があります。組織および組織ユーザは、VMware vSphere で事前設定する必要があります。展開データモデルについては、「VMware vSphere での仮想ネットワーク機能の展開」を参照してください。

VIM コネクタの詳細は次のとおりです。

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <vim_connector>
      <id>vimc-vc-lab</id>
      <type>VMWARE_VSPHERE</type>
      <properties>
        <property>
          <name>vcenter_ip</name>
          <value>IP_ADDRESS</value>
        </property>
        <property>
          <name>vcenter_port</name>
          <value>PORT</value>
        </property>
      </properties>
      <users>
        <user>
          <id>esc@vsphere.local</id>
          <credentials>
            <properties>
              <property>
                <name>vcenter_password</name>
                <value>PASS</value>
```

```
            </property>
          </properties>
        </credentials>
      </user>
    </users>
  </vim_connector>
 </vim_connectors>
</esc_system_config>
```

CSP クラスタへの VIM コネクタの追加

ESC は、既存の VIM コネクタペイロードの cluster_name プロパティを使用した CSP クラスタへの VIM コネクタの追加をサポートしています。

新しい VIM コネクタの作成

VIM コネクタが cluster_name プロパティで追加されると、ESC は csp_host_ip がクラスタの一部であるかどうかを検証して確認します。

次に、VIM コネクタをクラスタに追加する方法の例を示します。

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
   <vim_connector>
      <id>CSP-3</id>
      <type>CSP</type>
      <properties>
        <property>
          <name>csp_host_ip</name>
          <value> 168.20.117.16</value>
        </property>
        <property>
          <name>csp_host_port</name>
          <value>2022</value>
        </property>
         <property>
          <name>cluster_name</name>
          <value>Cluster_Test</value>
        </property>
        </properties>
      <users>
        <user>
          <id>admin</id>
          <credentials>
            <properties>
              <property>
                <name>csp_password</name>
                <value>password1</value>
              </property>
            </properties>
          </credentials>
        </user>
      </users>
```

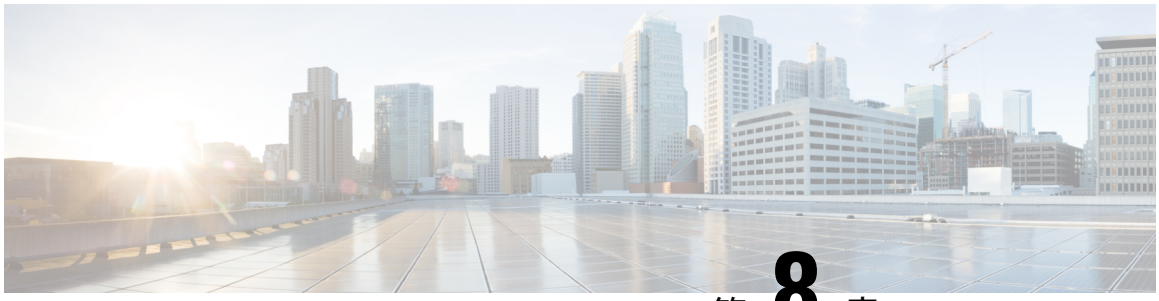
ESC で次のコマンドを実行して、クラスタに VIM コネクタを追加します。

```
esc_nc_cli --user <username> --password <password> edit-config  add_vim_connector.xml
```

csp_host_ip がクラスタの一部ではない場合、ESC は次のエラーを表示します。

```
Cluster [Cluster_Test] is not available or csp_host_ip is not valid.
```

CSP クラスタでの ESC を使用した VNF の展開の詳細については、「CSP クラスタでの ESC を使用した VNF の展開」の章を参照してください。



第 8 章

異なる VIM の VIM コネクタのプロパティ

VIM コネクタのプロパティ

VIM コネクタの設定により、ESC は VIM に接続できます。設定内のプロパティは、VIM とそのクレデンシャルに固有の詳細を提供します。次の表に、さまざまな VIM の VIM コネクタプロパティを示します。詳細については、VIM コネクタの管理 （51 ページ）を参照してください。

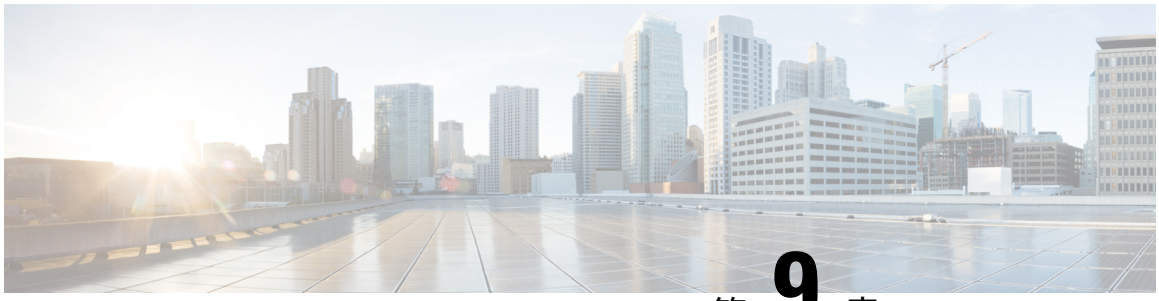
表 *4 : VIM* コネクタのプロパティ

VIM	プロパティ	参考資料
OpenStack	<properties> <property> <name>os_auth_url</name> <value>http://172.16.103.153:35357/v3</value> </property> <property> <name>os_project_domain_name</name> <value>default</value> </property> <property> <name>os_project_name</name> <value>admin</value> </property> <property> <name>os_identity_overwrite_endpoint</name> <value>http://some_server:some_port/</value> </property> <property> <name>os_network_overwrite_endpoint</name> <value>http://some_other_server:some_other_port/</value> </property>	詳細については、OpenStack の VIM コネクタの設定 （63 ページ）を参照してください。

VIM	プロパティ	参考資料
AWS	<properties> <property> <name>aws_default_region</name> <value>us-east-2</value> </property> </properties> <users> <user> <id>AWS_ACCESS_ID</id> <credentials> <properties> <property> <name>aws_secret_key</name> <encrypted_value>AWS_SECRET_KEY</encrypted_value> </property> </properties>	詳細については、AWS の VIM コネクタ設定 （71 ページ）を参照してください。
VMware vCD	<properties> <property> <name>authUrl</name> <!-- vCD is the vCD server IP or host name --> <value>https://vCD</value> </property> </properties> <users> <user> <!-- the user id here represents {org username}@{org name} --> <id>user@organization</id> <credentials> <properties> <property> <name>password</name> <!—the organization user’s password--> <value>put user’s password here</value> </property> </properties> </credentials> </user>	詳細については、VMware vCloud Director（vCD）の VIM コネクタの設定 （72 ページ）を参照してください。

VIM	プロパティ	参考資料
VMware vSphere	<esc_system_config xmlns="http://www.cisco.com/esc/esc"> <vim_connectors> <vim_connector> <id>vimc-vc-lab</id> <type>VMWARE_VSPHERE</type> <properties> <property> <name>vcenter_ip</name> <value>IP_ADDRESS</value> </property> <property> <name>vcenter_port</name> <value>PORT</value> </property> </properties> <users> <user> <id>esc@vsphere.local</id> <credentials> <properties> <property> <name>vcenter_password</name> <value>PASS</value> </property> </properties> </credentials> </user> </users> </vim_connector> </vim_connectors> </esc_system_config>	詳細については、VMware vSphere の VIM コネクタの設定 （73 ページ）を参照してください。

VIM	プロパティ	参考資料
Cisco Cloud Services Provider（CSP）2100	<properties> <property> <name>csp_host_ip</name> <value>172.16.89.100</value> </property> <property> <name>csp_host_port</name> <value>2022</value> </property> </properties> <users> <user> <id>admin</id> <credentials> <properties> <property> <name>csp_password</name> <value>*******</value> </property> </properties>	CSP の拡張については、クラウドサービスプロバイダーの拡張機能（399ページ）を参照してください。



第 9 章

外部設定ファイルの認証

外部設定ファイルの認証

Cisco ESC リリース 4.0 以前では、ESC は、デイゼロ設定、モニタリング、展開、および LCS アクションの一部として、いくつかの外部設定ファイルとスクリプトをサポートしています。ESC は、展開の一部として、認証の有無にかかわらず、リモートサーバからのこれらのファイルの取得をサポートしています。

ESC リリース 4.0 以降、ファイルロケータ属性は展開レベル、つまり展開コンテナの直下で定義されます。これにより、複数の VM グループとそのデイゼロ設定および LCS アクションが、展開内の必要な場所で同じファイルロケータを参照できるようになります。

展開データモデルの例は次のとおりです。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>sample-tenant</name>
      <deployments>
        <deployment>
          <name>sample-deployment</name>
          <file_locators>
            <file_locator>
              <name>post_deploy_alive_script</name>
              <remote_file>
                <file_server_id>http-my-server</file_server_id>
              <remote_path>/share/qatest/vnfupgrade/lcspostdeployalive.sh</remote_path>

                <local_target>vnfupgrade/lcspostdepalive.sh</local_target>
                <persistence>FETCH_ALWAYS</persistence>
                <properties/>
              </remote_file>
            </file_locator>
            <file_locator>
              <name>asa-day0-config</name>
              <remote_file>
```

```
            <file_server_id>http-my-server</file_server_id>
            <remote_path>/share/qatest/day0/asa_config.sh</remote_path>
            <local_target>day0.1/asa_config.sh</local_target>
            <persistence>FETCH_ALWAYS</persistence>
          </remote_file>
        </file_locator>
        <file_locator>
          <name>scriptlocator</name>
          <remote_file>
            <file_server_id>dev_test_server</file_server_id>
            <remote_path>/share/users/gomoore/actionScript.sh</remote_path>
            <local_target>action/actionScript.sh</local_target>
            <persistence>FETCH_MISSING</persistence>
            <properties/>
          </remote_file>
        </file_locator>
      </file_locators>
      <policies>
        <policy>
          <name>VNFUPGRADE_POST_DEPLOY_ALIVE</name>
          <conditions>
            <condition>
              <name>LCS::POST_DEPLOY_ALIVE</name>
            </condition>
          </conditions>
          <actions>
            <action>
              <name>post_deploy_alive_action</name>
              <type>SCRIPT</type>
              <properties>
                <property>
                  <name>file_locator_name</name>
                  <value>post_deploy_alive_script</value>
                </property>
              </properties>
            </action>
          </actions>
        </policy>
      </policies>
    <vm_group>
        <name>ASA-group</name>
        <image>ASAImage</image>
        <flavor>m1.large</flavor>
        <recovery_policy>
          <max_retries>1</max_retries>
        </recovery_policy>
        <scaling>
          <min_active>1</min_active>
          <max_active>1</max_active>
          <elastic>true</elastic>
        </scaling>
        <placement>
          <type>affinity</type>
          <enforcement>strict</enforcement>
        </placement>
        <bootup_time>120</bootup_time>
        <recovery_wait_time>60</recovery_wait_time>
        <interfaces>
          <interface>
            <nicid>0</nicid>
            <network>my-net</network>
          </interface>
        </interfaces>
        <kpi_data>
```

```
                    <kpi>
                      <event_name>VM_ALIVE</event_name>
                      <metric_value>1</metric_value>
                      <metric_cond>GT</metric_cond>
                      <metric_type>UINT32</metric_type>
                      <metric_occurrences_true>1</metric_occurrences_true>
                      <metric_occurrences_false>5</metric_occurrences_false>
                      <metric_collector>
                        <nicid>0</nicid>
                        <type>ICMPPing</type>
                        <poll_frequency>5</poll_frequency>
                        <polling_unit>seconds</polling_unit>
                        <continuous_alarm>false</continuous_alarm>
                      </metric_collector>
                    </kpi>
                  </kpi_data>
                  <rules>
                <admin_rules>
                        <rule>
                          <event_name>VM_ALIVE</event_name>
                          <action>ALWAYS log</action>
                          <action>TRUE servicebooted.sh</action>
                          <action>FALSE recover autohealing</action>
                        </rule>
                      </admin_rules>
                  </rules>
                  <config_data>
                    <configuration>
                      <dst>ASA.static.txt</dst>
                      <file_locator_name>asa-day0-config</file_locator_name>
                    </configuration>
                  </config_data>
                  <policies>
                    <policy>
                      <name>SVU1</name>
                      <conditions>

<condition><name>LCS::DEPLOY_UPDATE::PRE_VM_VOLUME_DETACH</name></condition>
                      </conditions>
                      <actions>
                        <action>
                          <name>LOG</name><type>pre_defined</type>
                        </action>
                        <action>
                          <name>pre_vol_detach</name>
                          <type>SCRIPT</type>
                          <properties>
                            <property>
                              <name>file_locator_name</name>
                              <value>scriptlocator</value>
                            </property>
                            <property>
                              <name>exit_val</name>
                              <value>0</value>
                            </property>
                          </properties>
                        </action>
                      </actions>
                    </policy>
                  </policies>
                </vm_group>
              </deployment>
            </deployments>
          </tenant>
```

```
  </tenants>
</esc_datamodel>
```

展開を実行する前に、APIを使用してリモートサーバ（ファイルサーバ）を個別に設定する必要があります。REST API と NETCONF API の両方がサポートされます。

- URL、ユーザ名を含む認証の詳細、およびパスワードを含むリモートサーバ。設定には REST または NETCONF を使用できます。

（注） ユーザ名とパスワードはオプションです。パスワードはESC内で暗号化されます。

展開前にリモートファイルサーバを設定する必要があります。クレデンシャルは、展開中にいつでも更新できます。

- ファイルロケータが展開データモデルに追加されます。ファイルサーバへの参照と、ダウンロードするファイルへの相対パスが含まれます。

認証を使用してリモートでファイルを取得するには、以下を行う必要があります。

1. リモートサーバを追加します。
2. ファイルロケータでリモートサーバを参照します。ファイルロケータは、デイゼロおよびLCS アクションブロックの設定データの一部です。
3. 展開の一部として、ファイルロケータに基づいてデイゼロおよびライフサイクルステージ（LCS）スクリプトが取得されます。

ファイルサーバのパラメータは次のとおりです。

- id：ファイルサーバのキーと識別子として使用されます。
- base_url：サーバのアドレス。（例：http://www.cisco.com または https://192.168.10.23）
- file_server_user：サーバへの認証時に使用するユーザ名。
- file_server_password：サーバへの認証用のパスワードを含む文字列。最初に、ユーザは内部で暗号化されたクリアテキスト文字列を指定します。
- properties：将来の拡張性のための名前と値のペア。

ファイルロケータのパラメータは次のとおりです。

- name：ファイルロケータのキーおよび識別子として使用されます。
- local_file または remote_file：ファイルの場所を選択します。ローカルファイルは、ESC VM ファイルシステムにすでに存在するファイルを指定するために使用されます。remote_file は、リモートサーバから取得するファイルを指定するために使用されます。
  - file_server_id：ファイルを取得するファイルサーバオブジェクトの ID。

- remote_path：ファイルサーバオブジェクトで定義された base_url からのファイルのパス。
- local_target：ファイルを保存するためのオプションのローカル相対ディレクトリ。
- properties：必要な情報の名前と値のペア。
- persistence：ファイルストレージのオプション。値には、CACHE、FETCH_ALWAYS、および FETCH_MISSING（デフォルト）が含まれます。

- checksum：転送されるファイルの有効性を検証するために使用する、オプションの BSD スタイルのチェックサム値。

サーバ接続、ファイルの存在、チェックサムなどのファイルサーバ値の有効性が検証されます。

file_server_password フィールドとプロパティの encrypted_data フィールドの encrypted_data 値は、伝送用 AES / 128 ビットを使用して CFB モードで暗号化されます。データは、サーバへのアクセスに必要になるまで暗号化されたままになります。暗号化された値の詳細については、「設定データの暗号化」を参照してください。

ファイルサーバの例

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <file_servers>
    <file_server>
      <id>server-1</id> <!-- unique name for server -->
      <base_url>https://www.some.server.com</base_url>
      <file_server_user>user1</file_server_user>
      <file_server_password>sample_password</file_server_password>
      <!-- encrypted value -->
      <!-- properties list containing additional items in the future -->
      <properties>
        <property>
          <name>server_timeout</name>
          <value>60</value>
      <!-- timeout value in seconds, can be over-ridden in a file_locator -->
        </property>
      </properties>
    </file_server>
    <file_server>
      <id>server-2</id>
      <base_url>https://www.some.other.server.com</base_url>
          <properties>
        <property>
          <name>option1</name>
          <encrypted_value>$8$EADFAQE</encrypted_value>
        </property>
    </file_server>
  </file_servers>
</esc_datamodel>
```

デイゼロ設定の例

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants><tenant>
      <name>sample-tenant</name>
      <deployments><deployment>
          <name>sample-deployment</name>
```

```
            <vm_group>
              <name>sample-vm-group</name>
              <config_data>
                <!-- exisiting configuration example - remains valid -->
                <configuration>
                  <file>file:///cisco/config.sh</file>
                  <dst>config.sh</dst>
                </configuration>
                <!-- new configuration including use of file locators -->
                <configuration>
                  <dst>something</dst>
                  <file_locators>
                    <file_locator>
                      <name>configlocator-1</name> <!-- unique name -->
                      <remote_file>
                        <file_server_id>server-1</file_server_id>
                        <remote_path>/share/users/configureScript.sh</remote_path>
                        <!-- optional user specified local silo directory -->
                        <local_target>day0/configureScript.sh</local_target>
                        <!-- persistence is an optional parameter -->
                        <persistence>FETCH_ALWAYS</persistence>
                        <!-- properties in the file_locator are only used for
                             fetching the file not for running scripts -->
                        <properties>
                          <property>
                            <!-- the property name "configuration_file" with value "true"
 indictates this is the
                                 script to be used just as using the <file> member case
of the configuration -->
                            <name>configuration_file</name>
                            <value>true</value>
                          </property>
                          <property>
                            <name>server_timeout</name>
                            <value>120</value> <!-- timeout value in seconds, overrides
the file_server property -->
                          </property>
                          </properties>
                        </remote_file>
                        <!-- checksum is an optional parameter.
                           The following algorithms are supported: SHA-1, SHA-224, SHA-256,
 SHA-384, SHA-512 -->
                        <checksum>SHA256 (configureScript.sh) =
dd526bb2c0711238ec2649c4b91598fb9a6cf1d2cb8559c337c5f3dd5ea1769e</checksum>
                      </file_locator>
                      <file_locator>
                        <name>configlocator-2</name>
                        <remote_file>
                          <file_server_id>server-2</file_server_id>
                          <remote_path>/secure/requiredData.txt</remote_path>
                          <local_target>day0/requiredData.txt</local_target>
                          <persistence>FETCH_ALWAYS</persistence>
                          <properties/>
                        </remote_file>
                      </file_locator>
                    </file_locators>
                  </configuration>
                </config_data>
              </vm_group>
            </deployment></deployments>
        </tenant></tenants>
</esc_datamodel>
```

デイゼロ設定およびLCSアクションの詳細については、「デイゼロ設定」および「再展開ポリシー」の項を参照してください。

設定データの暗号化

秘密キーと秘密情報を使用して設定データを暗号化できます。ESCでは、デイゼロ設定、デイゼロ設定変数、VIMコネクタとVIMユーザ、およびLCSアクションに秘密キーが含まれています。

ConfDは、暗号化された文字列タイプを提供します。組み込みの文字列タイプを使用すると、暗号化された値がConfDに保存されます。値の暗号化に使用されるキーは、confd.confに保存されます。

データの暗号化はオプションです。必要に応じて、encrypt_data値を使用してデータを保存できます。

次の例では、デイゼロ設定データに暗号化された値が含まれています。encrypted_dataは組み込みの文字列タイプ *tailf:aes-cfb-128-encrypted-string* を使用します。

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

Advanced Encryption Standard（AES）キーの生成

AESキーの長さは16バイトで、32文字の16進数文字列が含まれています。

暗号化を機能させるには、confd.confでAESキーを設定する必要があります。

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

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

デフォルトのAESキーはconfDで使用できます。

```
0123456789abcdef0123456789abcdef
```

confDキーはハードコードされています。escadm.pyはランダムなAESキーを生成し、confDが開始する前にデフォルトのconfD AESキーを置き換えます。

変数の暗号化

encrypted_val を使用して、デイゼロ設定内でパスワードやシャーシ ID などの変数を暗号化できます。ESC では、展開データモデル内の変数として *val* または *encrypted_val* を選択できます。

encrypted_val 内のテキストは暗号化されて、confD データベース（CDB）と PostgreSQL DB に格納されます。テキストは使用時にのみ復号化されます（データが保存されているときは復号化されません）。ESC ログでは、*encrypted_val* のテキストがマスクされます。

次の例では、ノースバウンドクライアント（Netconf または REST）で *encrypted_val* にプレーンテキストが入力されています。展開要求が ESC ConfD によって処理されると、プレーンテキストは暗号化されて ESC データベースに格納されます。

```
<config_data>
  <configuration>
    <dst>vnf_day0.cfg</dst>
    <data>file://opt/cisco/esc/esc_database/vnf_day0.cfg</file>
    <variable>
      <name>user</name>
      <val>admin</val>
    </variable>
    <variable>
      <name>password</name>
      <encrypted_val>ADMIN-PASSWORD</encrypted_val>
    </variable>
```

encrypted_val が netconf または CLI を介して ConfD 設定から取得されると、暗号化形式でプレーンテキストが表示されます。

```
<config_data>
  <configuration>
    <dst>vnf_day0.cfg</dst>
    <data>file://opt/cisco/esc/esc_database/vnf_day0.cfg</file>
    <variable>
      <name>user</name>
      <val>admin</val>
    </variable>
    <variable>
      <name>password</name>

<encrypted_val>$8$cVl6r9aR7W3wmHLYUrAOQHnjJGH0XltJjiCBTXANJFV0sJfb/NF+lEJiUA0j/JxA</encrypted_val>

    </variable>
```

（注） 単一の値が *encrypted_val* に格納されます。同じ変数値が置き換えられて、スケールグループ内にあるすべての VM のデイゼロ設定テンプレートに入力されます。

デイゼロ設定では、*encrypted_val* を使用してシャーシ ID を保護できます。シャーシ ID の値は、VNF のアップグレードを実行するノースバウンドクライアントまたはオペレータによって提供されます（VNF 展開中にスクリプトによって生成されたシャーシ ID はサポートされません）。

```
<config_data>
  <configuration>
    <dst>staros_param.cfg</dst>
    <file>file://opt/cisco/esc/images/staros_param_upf.cfg</file>
    <variable>
      <name>CHASSIS_ID</name>
      <encrypted_val>VALUE-PROVIDED-BY-NORTHBOUND-OPERATOR</encrypted_val>
    </variable>
```

デイゼロ設定の詳細については、デイゼロ設定 （173 ページ）を参照してください。

ConfD AES 暗号化文字列をエンコードするための Cisco Elastic Controller サービススクリプト

この機能は、dep.xml など、設定要求で使用できる AES 暗号化文字列をエンコードするスクリプトを提供します。次に、同じ機能を提供する 2 つのスクリプト（代替）を示します。

- `esc_nc_cli encrypt`
- `esc_confd_encrypt` ：ESC VM または ESC VM への接続が可能なリモート Linux サーバで使用するスタンドアロンスクリプトです。

次のコマンドは、プレーンテキストを AES 暗号化文字列に暗号化するのに役立ちます。

```
esc_nc_cli encrypt
```

次に例を示します。

```
admin@esc-01$ esc_nc_cli encrypt
Enter plain text (input is not echoed to terminal) >  *********
admin@127.0.0.1's password:
$8$aaCBcnVmZ+6lEV1FvhhitzQMLisLc3pxk1uUh+7DL4A=


admin@esc-01$ esc_nc_cli encrypt input.txt
admin@127.0.0.1's password:
$8$SLwFZuA0m0Rgf69fPNOeiq4ispm5H1SZIVGzzDd5R2g=
```

次のコマンドは、独立したスタンドアロンスクリプトとして実装される `esc_nc_cli` と同等です。

次に例を示します。

```
admin@esc-01$ esc_confd_encrypt
Enter plain text (input is not echoed to terminal) > ********
admin@localhost's password:
$8$QL5vFU1vt3KEs3kKIrC0+Faq8cF83WdptPO45GTIBGA=

admin@esc-01$ esc_confd_encrypt --file input.txt
admin@localhost's password:
$8$uzN7+kMgCf4RLxB5R0qMnLIbixO6EUpliUuHJRwR944=
```

次のコマンドは、ConfD CLI ssh（ポート 2024）に接続します。

次に例を示します。

```
admin@esc-01$ esc_nc_cli cli
ssh -o StrictHostKeyChecking=no -p 2024 admin@127.0.0.1
admin@127.0.0.1's password: *****

admin connected from 127.0.0.1 using ssh on esc-01
admin@esc-01>
```

リモートホストからのスクリプトの使用

両方のスクリプトを使用して、リモートESCで暗号化を実行できます。たとえば、ESC VM、ノースバウンドクライアント、または管理「ジャンプホスト」に接続できるLinuxサーバなどです。

次に例を示します。

```
abc@my-server-39:~$ esc_confd_encrypt --host 172.25.0.89 --user admin
Enter plain text (input is not echoed to terminal) >
admin@172.25.0.89's password:
$8$VUnQkT30fKqAWWCiyDPkqUjS+jDd0/sNIyGNd4bVppE=

abc@my-server-39:~$ esc_nc_cli encrypt --host 172.25.0.89 --user admin
Enter plain text (input is not echoed to terminal) >
admin@172.25.0.89's password:
$8$uRBKqpZZ9rcUIrfBam0WfCXq3tirTD+FRcafBqAArRs=

abc@my-server-39:~$ esc_nc_cli encrypt --host 172.25.0.89 --user admin --password
'REDACTED'
Enter plain text (input is not echoed to terminal) >
$8$iG9vvLAqk69wUSMVMVf5XDpwkdDi/P1V9ucJlXKn2NQ=
```

公開キー認証によるスクリプトへのパスワードレスアクセスの有効化

`esc_nc_cli`や`esc_confd_encrypt`など、ラッパーユーティリティを介して直接使用するために、ConfD（netconfおよびssh cli）に対してパスワードレスアクセス（公開キー認証）を有効にする方法は2つあります。

次に、ConfDで秘密キーペアと設定公開キー認証を作成する例を示します（推奨）。

```
admin@esc-01$ ssh-keygen -t rsa -b 2048 -C "admin" -N "" -f ~/.ssh/test_confd_rsa
Generating public/private rsa key pair.
Your identification has been saved in /home/admin/.ssh/test_confd_rsa.
Your public key has been saved in /home/admin/.ssh/test_confd_rsa.pub.
The key fingerprint is:
SHA256:u3/dpc4iY6/60fiGjGeJjMcigUKlSrxCptZWYo8JQ6o admin
The key's randomart image is:
+---[RSA 2048]----+
|                 |
| . .             |
|+ o              |
|.X o .           |
|O *.*   S        |
|Eo.=..   . o    .|
|o..  . +.+oo.. o.|
|    . o *.Xo+.o .|
|     . ooB+Booo  |
+----[SHA256]-----+
admin@esc-01$ sudo mkdir --mode=700 -p /var/confd/homes/admin/.ssh
admin@esc-01$ sudo cp ~/.ssh/test_confd_rsa.pub /var/confd/homes/admin/.ssh/authorized_keys
admin@esc-01$ sudo chown -R esc-user:esc-user /var/confd/homes/admin/.ssh

admin@esc-01$ printf "value-of-encrypted_val" | esc_nc_cli encrypt --privKeyFile
```

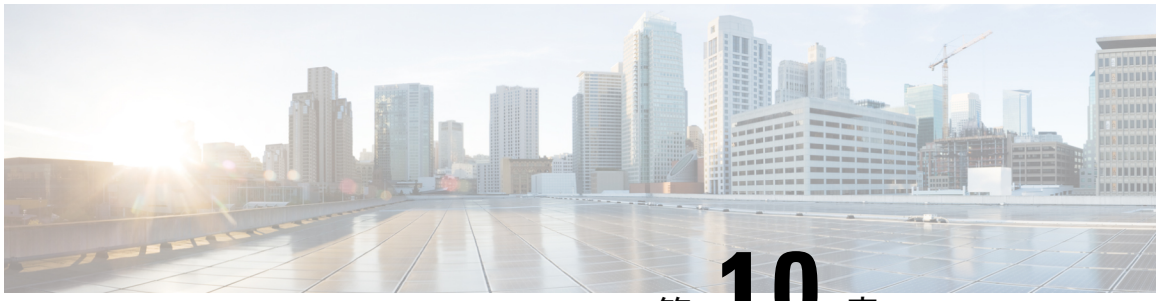
```
~/.ssh/test_confd_rsa
$8$VmDBKYupSGUCaILw8g2VYykVD9D16jA44sQNglFUUAv+uQtO0BmEtSC85vfuRJu0

admin@esc-01$ printf "value-of-encrypted_val" | esc_confd_encrypt --privKeyFile
~/.ssh/test_confd_rsa
$8$oFXwX1jeIHVxmBuMdPe6Vz6usaSahPVh0gZEGHm0uoAvK+twC0kUK5w7/QY0goUM

admin@esc-01$ cat .ssh/config
Host localhost 127.0.0.1
   Port 2024
   IdentityFile ~/.ssh/test_confd_rsa

admin@esc-01$ printf "value-of-encrypted_val" | esc_nc_cli encrypt
$8$GZ4+2nSo/YklKVk8RTdNR9oDJjWe89VsUiUR2FnIwtW4WPSXLivOXbmZnHR2YpfP

admin@esc-01$ printf "value-of-encrypted_val" | esc_confd_encrypt
$8$ggQaMq3QEIhS+1P8gmtr47LwdPyrCFoHHC2jzv2vKnxBFvIPNQapHurj+bcHfpEe
```

次に、組み込みの esc-nc-admin アカウントで ConfD にアクセスするために ConfD キーを有効にする例を示します（下位互換性のために提供）。

```
admin@esc-01$ sudo escadm confd keygen --user admin
Generated SSH key pair for user admin and authorized them for user esc-nc-admin

admin@esc-01$ printf "value-of-encrypted_val" | esc_nc_cli encrypt
$8$4c5m8cqK21VNyb1gCfc77p41LKxA9Ar8n6CApQwNst8yk/ilDphiDXetmHPmKuvP

admin@esc-01$ printf "value-of-encrypted_val" | esc_confd_encrypt
$8$yY8sG6leUkrnY+fBUrYVmnwPSBY9aIrUKXmpaHVGfvNWggLuSPkqZcRCjejPej+y
```

第 III 部

仮想ネットワーク機能のオンボーディング



第 10 章

仮想ネットワーク機能のオンボーディング

OpenStack および VMware vCenter で新しい VNF をオンボードできます。VNF をオンボードするには、前提条件を満たし、展開データモデルを準備する必要があります。この章では、OpenStack および VMware vCenter で展開データモデルを準備するための前提条件と手順について説明します。

OpenStack での仮想ネットワーク機能のオンボーディング

OpenStack で VNF をオンボーディングする前に、次の前提条件を満たす必要があります。

- VNF イメージ形式は、OpenStack と互換性がある必要があります（例：qcow2 形式）。イメージは、OpenStack Glance クライアント、あるいは NETCONF API または REST API を使用した ESC によって、OpenStack にオンボードできます。
- VM に渡されるデイゼロ設定ファイルは、OpenStack の設定ドライブまたはユーザデータのいずれかと互換性があるため、VM はデイゼロ設定の詳細をブートストラップメカニズムに使用できます。
- デイゼロ変数はプレーンテキスト形式で、事前定義された Day-0 変数を使用する必要があります。これにより、VM はデイゼロファイルで使用可能な静的 IP 情報を使用できます。

展開データモデルの準備

VNF オンボーディングの一環として、展開データモデルを準備する必要があります。展開データモデルとは、リソース要件、ネットワーキング、KPI のモニタリング、配置ポリシー、ライフサイクルステージ（LCS）、スケーリングルールなどの運用上の動作を記述する XML ファイル（テンプレート）です。

OpenStack 展開のためのデータモデルの準備

VNF 展開データモデルは、リソース要件、ネットワーキング、デイゼロ設定、および KPI のモニタリング、配置ポリシー、ライフサイクルステージ、スケーリングルールなど、他のサービスの運用動作を記述する XML ファイルまたはテンプレートです。

VNF をオンボードし、展開データモデルで VNF サービスを定義するには、次の手順を実行する必要があります。

1. VM リソースの準備
2. VNF ネットワーキングの説明
3. デイゼロ設定の準備
4. 展開データモデルでのメトリックや KPI などの運用動作の定義

VM リソースの準備

展開データモデルは、VNF を展開するためにテナント、イメージ、フレーバー、ボリュームなどのリソースを参照します。ESC を使用してこれらのリソースを作成するか、OpenStack ですでに使用可能な既存のリソースを使用できます。詳細については、リソース管理の概要 （17 ページ）を参照してください。

リソースを含むサンプルデータモデルは次のとおりです。

```
<?xml version="1.0" encoding="ASCII"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
     <name>vnf_tenant</name>
     <deployments>
      <deployment>
      ...
    <name>vnf-dep</name>
      <vm_group>
       <name>Grp1</name>
       <flavor>vnf_flavor</flavor>
       <image>vnf_image</image>
      ...
     </vm_group>
     </deployment>
     </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```

VNF ネットワークの説明

VNF に展開された VM は、さまざまな目的のために特定のネットワークに接続する必要があります。それらのネットワークは、管理ネットワーク、VM 内の内部ネットワークなどです。各ネットワークが OpenStack で使用可能であるか、または ESC によって作成されていることを確認します。ネットワークは、展開データモデルで定義して、展開時に作成する必要があります。詳細については、ネットワークの管理 （30 ページ）を参照してください。

ネットワークとサブネットワークを作成し、VMインターフェイスのネットワーク接続を指定する方法を示すサンプルの展開データモデルは次のとおりです。

```
<deployment>
         <name>vnf-dep</name>
         ...
         <networks>
           <network>
            <name>vnf_net</name>
            <shared>false</shared>
            <admin_state>true</admin_state>
            <subnet>
             <name>vnf_subnet</name>
             <ipversion>ipv4</ipversion>
             <dhcp>true</dhcp>
             <address>172.16.0.0</address>
             <netmask>255.255.255.0</netmask>
             <gateway>172.16.0.1</gateway>
            </subnet>
           </network>
         </networks>
        ...

      </deployment>

    </deployments>

        <vm_group>
         <name>Grp1</name>
         ...
          <interfaces>
          <interface>
           <nicid>0</nicid>
            <network>vnf_management</network>
           </interface>
            <interface>
             <nicid>1</nicid>
             <network>vnf_net</network>
            </interface>
           </interfaces>
          ...
         </vm_group>
```

デイゼロ設定の準備

デイゼロ設定の一環として、ブートストラップのため、インストール時にデイゼロファイルがVNFに渡されます。デイゼロファイルは、展開データモデルに記述されています。詳細については、デイゼロ設定 （173 ページ）を参照してください。

デイゼロファイルをコンフィグドライブおよびユーザデータとして記述するサンプルは次のとおりです。

```
<config_data>
          <configuration>
           <dst>--user-data</dst>
            <file>file://var/test/test-script.sh</file>
             </configuration>
             <configuration>
           <dst>/etc/configure-networking.sh</dst>
            <file>file://var/test/configure-networking.sh</file>
          </configuration>
         </config_data>
```

運用動作の定義

複合 VNF をオンボードするには、ネットワーク接続、KPI のモニタリング、配置ポリシー、ライフサイクルステージ、スケーリングルールなど、いくつかの運用動作を設定する必要があります。これらの動作は、展開データモデルで記述できます。詳細については、導入パラメータ （169 ページ）を参照してください。

これらの詳細を使用して展開データモデルを準備すると、VNF をオンボーディングし、OpenStack で VNF サービスをインスタンス化したことになります。これで、VNF を展開できます。VNF が展開されると、ESC が新しいサービスのデイゼロ設定を適用します。詳細については、OpenStack での仮想ネットワーク機能の展開 （115 ページ）を参照してください。

VMware vCenter における VNF の準備の詳細については、VMware vCenter 展開のためのデータモデルの準備 （98 ページ）を参照してください。

VMware vCenter での仮想ネットワーク機能のオンボーディング

VMware vCenter で VNF をオンボーディングする前に、次の前提条件を満たす必要があります。

- VNF イメージ形式は、ova などの VMware vCenter と互換性がある必要があります。
- VM に渡されるデイゼロ設定ファイルは、OVF プロパティまたは CDROM ドライブからの設定の読み取りと互換性がある必要があります。
- CDROM ドライブでは、デイゼロ変数はプレーンテキスト形式である必要があります。

VMware vCenter 展開のためのデータモデルの準備

VNF 展開データモデルは、リソース要件、ネットワーキング、デイゼロ設定、および KPI のモニタリング、配置ポリシー、ライフサイクルステージ、スケーリングルールなどのその他の動作を記述する XML ファイルまたはテンプレートです。

VNF をオンボードし、展開データモデルで VNF サービスを定義するには、次の手順を実行する必要があります。

1. VM リソースの準備
2. VNF ネットワーキングの説明
3. リソースプールとフォルダ仕様のサポート
4. デイゼロ設定の準備
5. 展開データモデルでのメトリックや KPI などの運用動作の定義

VM リソースの準備

展開データモデルは、VNF を展開するためのリソースを指します。イメージ（テンプレート）は、VMware 展開で参照される唯一のリソースです。イメージは既存のイメージでも、ESC で作成されたイメージでもかまいません。

（注） テナントは VMware vCenter 展開には存在しませんが、展開データモデルにはデフォルトの管理テナントが必要です。

イメージの詳細を含むサンプルデータモデルは次のとおりです。

```
<?xml version="1.0" encoding="ASCII"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
     <name>admin</name>
     <deployments>
     <deployment>
     ...
     <name>vnf-dep</name>
      <vm_group>
        <image>vnf_image</image>
          ...
     </vm_group>
      </deployment>
     </deployments>
    </tenant>
  </tenants>
 </esc_datamodel>
```

VMware vCenter では、各 vm_group に配置ポリシーとボリュームの詳細が必要です。zone_host タイプの配置は、展開のターゲットコンピューティングホストまたはクラスタを定義します。ボリュームは、展開のターゲットデータストアを定義します。次の展開データモデルは、コンピューティングクラスタ cluster1 への展開ターゲットを定義し、ESC がデータストアを自動的に選択できるようにします。

```
<?xml version="1.0" encoding="ASCII"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>admin</name>
      <deployments>
        <deployment>
  ...
        <name>vnf-dep</name>
          <vm_group>
            ...
 <placement>
   <type>zone_host</type>
  <zone>cluster1</zone>
   </placement>
     <volumes>
  <volume>
  <name>auto-select</name>
  <volid>1</volid>
  </volume>
  </volumes>
    </vm_group>
```

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

次の展開データモデルは、コンピューティングホスト host1 およびデータストア datastore1 への展開ターゲットを定義します。

```
<?xml version="1.0" encoding="ASCII"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>admin</name>
      <deployments>
        <deployment>
  ...
        <name>vnf-dep</name>
          <vm_group>
            ...
   <placement>
  <type>zone_host</type>
  <host>host1</host>
  </placement>
   <volumes>
     <volume>
      <name>datastore1</name>
      <volid>1</volid>
  </volume>
  </volumes>
   </vm_group>
    </deployment>
     </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```

VNF ネットワークの説明

VNF に展開された VM は、さまざまな目的のために特定のネットワークに接続する必要があります。これらのネットワークには、管理ネットワーク、VM 間の内部ネットワーク、およびさまざまな目的のその他のネットワークなどがあります。VMware では、ネットワークとは vDS ポートグループを指し、サブネットは vCenter の IP プールを指します。ESC は、VMware 展開の静的 IP のみをサポートします。これらのネットワークが VMware vCenter で使用可能であるか、ESC によって作成されていることを確認します。展開中にネットワークを作成するには、展開データモデルでネットワークを定義します。展開データモデルは次のとおりです。

```
        <deployment>
          <name>vnf-dep</name>
          ...
         <networks>
            <network>
              <name>vnf_management</name>
              <admin_state>true</admin_state>
              <number_of_ports>8</number_of_ports>
              <shared>false</shared>
              <switch_name>vdswitch1</switch_name>
              <vlan_id>0</vlan_id>
              <subnet>
                <name>vnf_management-subnet</name>
                <ipversion>ipv4</ipversion>
```

```
            <dhcp>false</dhcp>
            <address>172.16.0.0</address>
            <netmask>255.255.255.0</netmask>
            <gateway>172.16.0.1</gateway>
          </subnet>
        </network>
      </networks>
      ...

    </deployment>
  </deployments>
```

（注） VMware Vcenter では、nicid 値は 1 から始まります。OpenStack では、nicid 値は 0 から始まります。

```
<vm_group>
          <name>Grp1</name>
          ...
          <interfaces>
            <interface>
             <nicid>1</nicid>
             <network>vnf_management</network>
            </interface>
            <interface>
             <nicid>2</nicid>
             <network>vnf_net</network>
            </interface>
          </interfaces>
          ...
        </vm_group>
```

デフォルト以外のリソースプールとフォルダをサポートするための ESC（vCenter の場合）

vCenter または vSphere 展開の一部として、ユーザーは必要に応じて展開 XML 記述子に注釈を付けて、展開の対象となるリソースプールやフォルダを記述できます。

リソースプールとフォルダはVMグループレベルの属性であり、次に示すように、オプションの拡張セクションで名前または値のペアを使用して指定されます。

```
<deployment>
         <name>Deployment-Name</name>
         <vm_group>
           <name>VM-Group-Name</name>
           <bootup_time>300</bootup_time>
           . . .
           <interfaces>
             <interface>
               . . .
             </interface>
           </interfaces>
           <extensions>
             <extension>
               <name>vmware_vsphere_placement</name>
               <properties>
                 <property>
                   <name>folder</name>
                   <value>My_Folder</value>
                 </property>
```

```
            <property>
              <name>resource_pool</name>
              <value>My_Resource_Pool</value>
            </property>
          </properties>
        </extension>
      </extensions>
```

（注）

1. 拡張機能名は vmware_vsphere_placement にする必要があります。違う名前の場合、ESC はその下のプロパティを無視します。
2. 「My_Folder」というフォルダは、このフォルダが存在しない場合に作成されます。それ以外の場合、フォルダはそのまま使用されます。
3. リソースプール「My_Resource_Pool」が存在する必要があります。存在しない場合、エラーが返されます。リソースプールの作成には、展開属性でサポートされていないCPUおよびメモリ使用量に関連する多くのパラメータ値が必要なため、エラーが返されます。
4. 既存の動作では、拡張機能が指定されていない場合、デフォルトのクラスタリソースプールとフォルダが使用されます。
5. 展開 XML で単一のリソースプールまたは単一のフォルダを指定します。単一のリソースプールと単一のフォルダはどちらも独立しており、展開 XML で使用するために相互に依存していません。

デイゼロ設定の準備

デイゼロ設定の一環として、ブートストラップのため、インストール時にデイゼロファイルがVNFに渡されます。デイゼロファイルは、展開データモデルに記述する必要があります。詳細については、デイゼロ設定（173ページ）を参照してください。サンプルのデイゼロファイルは、展開された VM に接続された CDROM コンテンツのファイルとして渡されたデイゼロ設定を示しています。

```
<config_data>
          <configuration>
            <dst>day0-config</dst>
            <file>http://somehost:80/day0.txt</file>
          </configuration>
          <configuration>
            <dst>idtoken</dst>
            <file>http://somehost:80/idtoken.txt</file>
          </configuration>
        </config_data>
```

次の例は、OFV 設定を介して渡されるデイゼロ設定を示しています。

```
<config_data>
         <configuration>
          <dst>ovfProperty:mgmt-ipv4-addr</dst>
          <data>$NICID_1_IP_ADDRESS/16</data>
          </configuration>
         <configuration>
          <dst>ovfProperty:com.cisco.csr1000v:hostname</dst>
          <data>$HOSTNAME</data>
```

```
    <variable>
     <name>HOSTNAME</name>
     <val>csrhost1</val>
     <val>csrhost2</val>
    </variable>
   </configuration>
  </config_data>
```

運用動作の定義

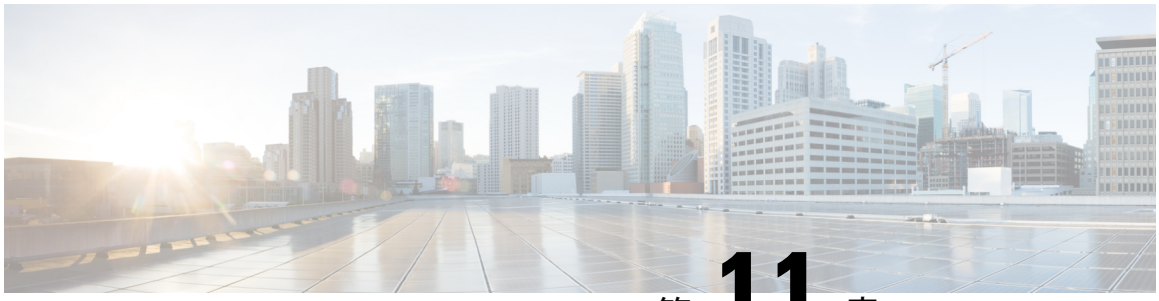
複合 VNF をオンボードするには、ネットワーク接続、KPI のモニタリング、配置ポリシー、ライフサイクルステージ、スケーリングルールなど、いくつかの運用動作を設定する必要があります。これらの動作は、展開データモデルで記述できます。詳細については、導入パラメータ （169 ページ）を参照してください。

これらの詳細を使用して展開データモデルを準備すると、VNF をオンボーディングし、OpenStack で VNF サービスをインスタンス化したことになります。これで、VNF を展開できます。VNF が展開されると、ESC が新しいサービスのデイゼロ設定を適用します。詳細については、VMware vCenter のイメージ （147 ページ）を参照してください。

OpenStack での VNF の準備については、OpenStack 展開のためのデータモデルの準備 （96 ページ）を参照してください。

第 IV 部

仮想ネットワーク機能の展開と設定



第 11 章

ESC トランクおよび VLAN 機能

ESC トランクおよび VLAN 機能

VM には、ドメイン内のネットワークにアクセスするように設定された 1 つ以上のインターフェイスがある場合があります。たとえば、データネットワーク用の eth0 と管理ネットワーク用の eth1。

VM が複数のネットワークに接続する必要がある場合は、OpenStack トランキングを使用して構成を簡素化します。

（注） トランクと VLAN のサポートは、ESC 5.8 で実装されました。

ESC 展開では、VM グループで次のようにトランクを定義します。

```
<vm_group>
    <name>...</name>
    <image>...</image>
    <flavor>...</flavor>
    <interfaces>
        <interface>
            <nicid>0</nicid>
            <network>parent-net</network>
        </interface>
    </interfaces>
    <trunks>
        <trunk>
            <name>trunk-name</name>
            <parent_nicid>0</parent_nicid>
            <subports>
                <subport>
                    <name>child-port</name>
                    <network>child-net</network>
                    <segmentation_type>vlan</segmentation_type>
                    <segmentation_id>500</segmentation_id>
                </subport>
            </subports>
```

```
        </trunk>
    </trunks>
</vm_group>
```

<trunks> の下にある要素の説明

要素	必須	説明
トランク	y	トランクの下に複数のトランクを定義できます。
trunk	y	トランク定義をラップします。
> name	y	トランクの名前。これはOpenStackのトランクの名前になります。
> parent_nicid	y	これは、<interfaces> の下で定義されている NIC ID を指定します。この例では 0。この NIC 用に作成されたポートは、トランクの親ポートとして使用されます。
> subports	y	1 つ以上のサブポート要素のラッパー。
>> subport	y	サブポート定義をラップします。少なくとも 1 つのサブポートを定義する必要があります。
>>> name	y	サブポートの名前。これは、OpenStack のサブポートの名前になります。
>>> network	y	このポートに関連付ける OpenStack ネットワークの名前または ID。展開用に定義された外部またはエフェメラルネットワークである可能性があります。
>>> segmentation_type	n	デフォルトは vlan です。他の可能な値については、Openstack API のドキュメントを参照してください。
>>> segmentation_id	y	このサブポートに割り当てる VLAN ID。

トランクの作成：

REST または NETCONF インターフェイスを使用して、展開 XML を送信します。トランクは、ESC が VM を展開する前に作成されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2022-06-08T13:39:14.609+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Trunk trunk-D120-vm1: CREATE_TRUNK completed
successfully</status_message>
    <event>
      <type>CREATE_TRUNK</type>
    </event>
  </escEvent>
</notification>
```

HTTP コールバックメッセージ：

```
::ffff:127.0.0.1 - - [08/Jun/2022 13:39:09] "POST / HTTP/1.1" 200 2
------------------------------
```

```
REQUEST_METHOD:          POST
SERVER_PORT:             9009
PATH_INFO:               /
CONTENT_TYPE:            application/json
HTTP_ESC_TRANSACTION_ID  b9cce743-afd8-4eda-9303-5aaeccf4d400
HTTP_ESC_STATUS_MESSAGE *  Trunk trunk-D120-vm1: CREATE_TRUNK completed successfully
HTTP_ESC_STATUS_CODE     200
DATA:
------------------------------
* <JSON config data>
---------------END DATA---------------
```

デイゼロ設定：

/etc/network/interfaces ファイルでのサブインターフェイスの作成や、テスト用の IP リンクコマンドの実行など、VM でトランクを使用するにはネットワーク構成が必要です。ESC が VM を展開すると、トランクサブポート情報がデイゼロスクリプトで利用可能になります。通常のテンプレート変数に加えて、次の変数を使用できます。インデックスは、トランクの下で定義されているサブポートに基づきゼロから始まります。

- SUBPORT_<index>_VLAN_ID
- SUBPORT_<index>_MAC_ADDRESS
- SUBPORT_<index>_ID
- SUBPORT_<index>_NETWORK
- SUBPORT_<index>_NAME
- SUBPORT_<index> _NETWORK_ID
- SUBPORT_<index>_SEGMENTATION_TYPE

構成データの例：

```
<config_data>
    <configuration>
        <dst>--user-data</dst>
        <data>
#cloud-config
hostname: D120-vm2
password: secret
chpasswd: { expire: False }
ssh_pwauth: True
runcmd:
  - [ sh, -xc, "ip link add link ens3 name ens3.$SUBPORT_0_VLAN_ID address
$SUBPORT_0_MAC_ADDRESS type vlan id $SUBPORT_0_VLAN_ID" ]
  - [ sh, -xc, "ip link set dev ens3.$SUBPORT_0_VLAN_ID up" ]
  - [ sh, -xc, "dhclient -v ens3.$SUBPORT_0_VLAN_ID" ]
        </data>
    </configuration>
</config_data>
```

トランクのクエリ：

REST または NETCONF API を使用して、トランクとサブポートの展開データを利用できます。

```
<trunks>
    <trunk>
        <port_id>2f1dc1e6-a90a-4568-8c9e-965fda6c0cfb</port_id>
        <parent_nicid>0</parent_nicid>
```

```
        <id>13485eec-c0e0-41d0-b32e-b95ecd23ecef</id>
        <name>trunk-D120-vm1</name>
        <subports>
            <name>child-port-D120-vm1</name>
            <network>child-D120-net</network>
            <mac_address>fa:16:3e:72:cd:8d</mac_address>
            <segmentation_type>vlan</segmentation_type>
            <segmentation_id>120</segmentation_id>
            <id>1b6a0601-3281-4950-baca-f485470f74e3</id>
        </subports>
    </trunk>
</trunks>
```

トランクの削除：

VM が展開解除されると、トランクとサブポートが削除されます。NETCONF メッセージは、トランクごとに投稿されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2022-06-08T13:24:47.834+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Trunk trunk-D120-vm2: DELETE_TRUNK completed
successfully</status_message>
    <event>
      <type>DELETE_TRUNK</type>
    </event>
  </escEvent>
</notification>
```

HTTP コールバックイベント：

```
::ffff:127.0.0.1 - - [08/Jun/2022 13:24:47] "POST / HTTP/1.1" 200 2
------------------------------
REQUEST_METHOD:          POST
SERVER_PORT:             9009
PATH_INFO:               /
CONTENT_TYPE:            application/xml
HTTP_ESC_TRANSACTION_ID  be42f1d2-a792-4516-91dc-583bdcd28c55
HTTP_ESC_STATUS_MESSAGE *  Trunk trunk-D120-vm2: DELETE_TRUNK completed successfully
HTTP_ESC_STATUS_CODE     200
DATA:
------------------------------
* <?xml version="1.0" encoding="UTF-8"?><!-- trunk details -->
---------------END DATA---------------
```

トランクの変更：

トランク自体は変更されませんが、サブポートは追加、削除、および変更できます。

HTTP PUT REST を送信すると、必要に応じてトランクサブポートが変更されます。サブポートを PUT リクエストのサブポートに置き換えます。

- リクエストにない既存のサブポートは VM から削除されます。
- リクエスト内の既存のサブポートは残り、ID、MAC アドレスは保持されます。
- リクエストに新しいサブポートを追加します。

```
curl -X PUT "http://localhost:8080/ESCManager/v0/deployments/D120" \
    -H "Callback: http://localhost:9009" \
```

```
    -H "Callback-ESC-Events: http://localhost:9009" \
    -H "Content-Type: application/xml" \
    -d "<esc_datamodel xmlns=\"http://www.cisco.com/esc/esc\"> ..."
```

NETCONF `edit-config` の使用

NETCONF リクエストでは、ルールが少し異なります。

- リクエストにない既存のサブポートは無視し、変更せずに続行します。
- リクエストに新しいサブポートを追加します。
- サブポートを削除し、nc:operation='delete' で注釈を付けます。次に例を示します。

```
<subport nc:operation='delete'>
    <name>child-port-D120-vm1</name>
    <network>child-D120-net</network>
    <segmentation_type>vlan</segmentation_type>
    <segmentation_id>120</segmentation_id>
</subport>
```

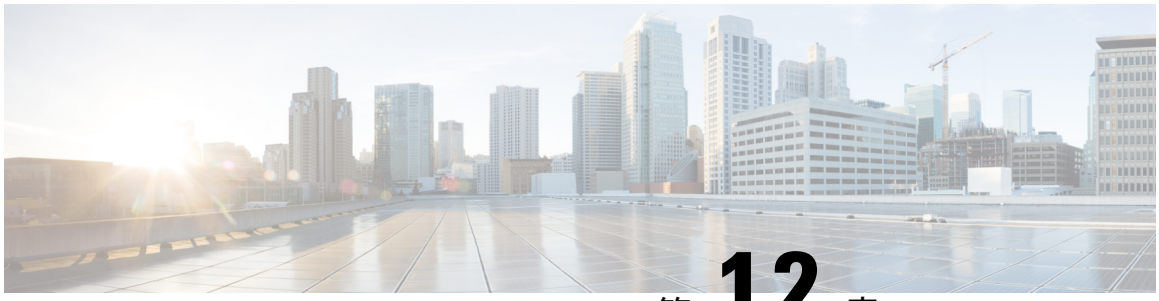
`edit-config`（REST API）の使用

内部 REST API を使用して、NETCONF ペイロードを送信します。

```
curl -X POST --location "http://localhost:8080/ESCManager/internal/conf/edit-config" \
    -H "Callback: http://localhost:9009" \
    -H "Callback-ESC-Events: http://localhost:9009" \
    -H "Content-Type: application/xml" \
    -d "<esc_datamodel
    xmlns=\"http://www.cisco.com/esc/esc\"
    xmlns:nc=\"urn:ietf:params:xml:ns:netconf:base:1.0\">
    ..."
```

制限事項

- OpenStack では、展開された VM にトランクを追加できません。セカンダリインターフェイスを接続し、ポートをトランクの親として使用できます。
- VM グループのスケーリングはサポートされていません。
- サブポートの MAC アドレスの指定はできません。



第 12 章

仮想ネットワーク機能の展開

仮想ネットワーク機能の展開

OpenStack、VMware vCenter または AWS のいずれかで、仮想インフラストラクチャ ドメイン内の VNF をオーケストレーションできます。VNF 展開は、ノースバウンド インターフェイスまたは ESC ポータルを介してサービスリクエストとして開始されます。サービスリクエストは、XML ペイロードと展開パラメータから成るテンプレートで構成されます。この章では、VNF（OpenStack または VMware vCenter）を展開する手順と、展開中に実行できる操作について説明します。展開パラメータの詳細については、「展開パラメータの設定」を参照してください。

重要 静的 IP アドレスを割り当てて、ネットワークを VNF に接続できます。展開データモデルでは、静的 IP アドレスを指定する新しい ip_address 属性が導入されています。詳細については、「Cisco Elastic Services Controller Deployment Attributes」を参照してください。

基本的なインターフェイス設定の詳細については、『*Cisco Elastic Services Controller Administration Guide*』を参照してください。

第 13 章

OpenStack での仮想ネットワーク機能の展開

OpenStack での仮想ネットワーク機能の展開

ここでは、Elastic Services Controller（ESC）のいくつかの展開シナリオと VNF の展開手順について説明します。次の表に、さまざまな展開シナリオを示します。

シナリオ	説明	リソース	利点
ESC を使用してイメージとフレーバーを作成することにより、単一の VIM に VNF を展開する	展開データモデルは、作成されたイメージとフレーバーを参照して、VNF を展開します。	イメージとフレーバーは、NETCONF/REST API を使用して ESC で作成されます。	• イメージとフレーバーは、複数の VNF 展開で使用できます。 • ESC によって作成されたリソース（イメージ、フレーバー、およびボリューム）を削除できます。

シナリオ	説明	リソース	利点
アウトオブバンドイメージ、フレーバー、ボリューム、およびポートを使用した単一 VIM への VNF の展開	展開データモデルは、OpenStack のアウトオブバンドイメージ、フレーバー、ボリューム、およびポートを参照して、VNF を展開します。	イメージ、フレーバー、ボリューム、およびポートは、ESC を使用して作成されません。	• イメージ、フレーバー、ボリューム、ポートは、複数の VNF 展開で使用できます。 • ESC を使用して作成されていないリソースは削除できません。
アウトオブバンドリソースを使用した複数の VIM への VNF の展開	展開データモデルは、アウトオブバンドイメージ、フレーバー、ネットワーク、および VIM プロジェクトを参照して、VNF を展開します。	イメージ、フレーバー、VIM プロジェクト（ロケータで指定）およびネットワークは、ESC を使用して作成されません。これらは、VIM のアウトオブバンドに存在する必要があります。	展開内の ESC で設定する必要がある（VM を展開するための）VIM を指定できます。

複数の OpenStack VIM に VNF を展開するには、「複数の OpenStack VIM への VNF の展開」を参照してください。

単一の OpenStack VIM での VNF の展開

VNF の展開は、ESC ポータルまたはノースバウンド インターフェイスから発信されるサービス要求として開始されます。サービス要求は XML ペイロードで構成されます。ESC は、次の展開シナリオをサポートします。

- ESC を使用したイメージおよびフレーバの作成による VNF の展開
- アウトオブバンドイメージ、フレーバ、ボリューム、およびポートを使用した VNF の展開

VNF を展開する前に、OpenStack でイメージ、フレーバ、ボリューム、およびポートが使用可能であることを確認するか、これらのリソースを作成する必要があります。イメージ、フレーバ、およびボリュームの作成の詳細については、リソース管理の概要 （17 ページ）を参照してください。

展開では、展開と同じテナントによってアウトオブバンドポートを作成する必要があります。ポートの設定の詳細については、『*Cisco Elastic Services Controller Administration Guide*』の「Interface Configurations」を参照してください。

複数の VIM に VM を展開するには、「複数の OpenStack VIM への VNF の展開」を参照してください。

展開中、ESC は展開データモデルで展開の詳細を検索します。展開データモデルの詳細については、「Cisco Elastic Services Controller Deployment Attributes」を参照してください。ESC が特定のサービスに対する展開の詳細を見つけることができない場合は、*vm_group* の既存のフレーバとイメージを使用して展開を続行します。ESC がイメージとフレーバの詳細を検出できない場合、展開は失敗します。

重要 ネットワークに使用するサブネットを指定することもできます。展開データモデルでは、サブネットを指定する新しい `subnet` 属性が導入されています。詳細については、「Cisco Elastic Services Controller Deployment Attributes」を参照してください。

（注） SERVICE_UPDATE 設定が失敗すると、VM の最小数と最大数が変化し、スケールインまたはスケールアウトが発生します。OpenStack で発生したエラーのため、ESC は設定内の VM の最小数または最大数をロールバックできません。CDB（ESC DB）が同期していません。この場合、手動ロールバックを実行するには、別の SERVICE_UPDATE 設定を実行する必要があります。

OpenStack での展開では、UUID または名前を使用してイメージとフレーバを参照できます。名前は VIM で一意である必要があります。同じ名前の複数のイメージがある場合、展開は正しいイメージを識別できず、展開は失敗します。

すべての展開および ESC イベント通知にテナント UUID が表示されます。次に例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-01-22T15:14:52.484+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>VIM Driver: VM successfully created,
      VM Name:
[SystemAdminxyz_abc_NwDepMod1_0_5e6b7957-20e7-4df9-9113-e5fc8c047e91]</status_message>
    <depname>test_NwDepModVmGrp1</depname>
    <tenant>admin</tenant>
    <tenant_id>62cd11f560b44bf5815eaad41fc94c80</tenant_id>
 </event>
```

再起動時間パラメータ

再起動時間パラメータが展開要求に導入されます。これにより、展開におけるリカバリの再起動待機時間をよりきめ細かく制御できます。展開では、VM が再起動すると、モニタに再起動時間が設定されます。VM ALIVE イベントの前に再起動時間が経過すると、VM_RECOVERY_COMPLETE や undeploy などの次のアクションが実行されます。

（注） 再起動時間が指定されていない場合は、ブートアップ時間が使用されます。

データモデルの変更は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
   <tenants>
      <tenant>
         <name>tenant</name>
         <deployments>
            <deployment>
               <name>depz</name>
               <vm_group>
                  <name>g1</name>
                  <image>Automation-Cirros-Image</image>
                  <flavor>Automation-Cirros-Flavor</flavor>
                  <reboot_time>30</reboot_time>
                  <recovery_wait_time>10</recovery_wait_time>
                  <interfaces>
                     <interface>
                        <nicid>0</nicid>
                        <port>pre-assigned_IPV4_1</port>
                        <network>my-network</network>
                     </interface>
                  </interfaces>
                  <kpi_data>
                     <kpi>
                        <event_name>VM_ALIVE</event_name>
                        <metric_value>1</metric_value>
                        <metric_cond>GT</metric_cond>
                        <metric_type>UINT32</metric_type>
                        <metric_collector>
                           <nicid>0</nicid>
                           <type>ICMPPing</type>
                           <poll_frequency>3</poll_frequency>
                           <polling_unit>seconds</polling_unit>
                           <continuous_alarm>false</continuous_alarm>
                        </metric_collector>
                     </kpi>
                  </kpi_data>
                  <rules>
                     <admin_rules>
                        <rule>
                           <event_name>VM_ALIVE</event_name>
                           <action>ALWAYS log</action>
                           <action>TRUE servicebooted.sh</action>
                           <action>FALSE recover autohealing</action>
                        </rule>
                     </admin_rules>
                  </rules>
                  <config_data />
                  <scaling>
                     <min_active>1</min_active>
                     <max_active>2</max_active>
                     <elastic>true</elastic>
                  </scaling>
                  <recovery_policy>
                     <recovery_type>AUTO</recovery_type>
                     <action_on_recovery>REBOOT_ONLY</action_on_recovery>
                     <max_retries>1</max_retries>
```

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

通知の例は次のとおりです。

```
20:43:48,133 11-Oct-2016 WARN  ===== SEND NOTIFICATION STARTS =====
20:43:48,133 11-Oct-2016 WARN  Type: VM_RECOVERY_INIT
20:43:48,134 11-Oct-2016 WARN  Status: SUCCESS
20:43:48,134 11-Oct-2016 WARN  Status Code: 200
20:43:48,134 11-Oct-2016 WARN  Status Msg: Recovery event for
VM [dep-12_CSR1_c_0_37827511-be08-4702-b0bd-1918cb995118] triggered.
20:43:48,134 11-Oct-2016 WARN  Tenant: gilan-test-5
20:43:48,134 11-Oct-2016 WARN  Service ID: NULL
20:43:48,134 11-Oct-2016 WARN  Deployment ID: f6ff8164-fe6d-4589-84fa-f39d676e9231
20:43:48,134 11-Oct-2016 WARN  Deployment name: dep-12
20:43:48,134 11-Oct-2016 WARN  VM group name: CSR1_cirros
20:43:48,134 11-Oct-2016 WARN  VM Source:
20:43:48,134 11-Oct-2016 WARN  VM ID: 90d2066c-9a07-485b-8f72-b51026a62922
20:43:48,134 11-Oct-2016 WARN  Host ID:
69c3fba0a5b5ffff211bd05b9da7e2130d98d005a9bef71ace7d09ff
20:43:48,134 11-Oct-2016 WARN  Host Name: my-server
20:43:48,134 11-Oct-2016 WARN  [DEBUG-ONLY] VM IP: 192.168.0.75;
20:43:48,135 11-Oct-2016 WARN  =====  SEND NOTIFICATION ENDS  =====
20:43:56,149 11-Oct-2016 WARN
20:43:56,149 11-Oct-2016 WARN  ===== SEND NOTIFICATION STARTS =====
20:43:56,149 11-Oct-2016 WARN  Type: VM_RECOVERY_REBOOT
20:43:56,149 11-Oct-2016 WARN  Status: SUCCESS
20:43:56,149 11-Oct-2016 WARN  Status Code: 200
20:43:56,150 11-Oct-2016 WARN  Status Msg: VM
[dep-12_CSR1_c_0_37827511-be08-4702-b0bd-1918cb995118] is rebooted.
20:43:56,150 11-Oct-2016 WARN  Tenant: gilan-test-5
20:43:56,150 11-Oct-2016 WARN  Service ID: NULL
20:43:56,150 11-Oct-2016 WARN  Deployment ID: f6ff8164-fe6d-4589-84fa-f39d676e9231
20:43:56,150 11-Oct-2016 WARN  Deployment name: dep-12
20:43:56,150 11-Oct-2016 WARN  VM group name: CSR1_cirros
20:43:56,150 11-Oct-2016 WARN  VM Source:
20:43:56,151 11-Oct-2016 WARN    VM ID: 90d2066c-9a07-485b-8f72-b51026a62922
20:43:56,151 11-Oct-2016 WARN    Host ID:
69c3fba0a5b5ffff211bd05b9da7e2130d98d005a9bef71ace7d09ff
20:43:56,151 11-Oct-2016 WARN    Host Name: my-server
20:43:56,152 11-Oct-2016 WARN    [DEBUG-ONLY] VM IP: 192.168.0.75;
20:43:56,152 11-Oct-2016 WARN  =====  SEND NOTIFICATION ENDS  =====
20:44:26,481 11-Oct-2016 WARN
20:44:26,481 11-Oct-2016 WARN  ===== SEND NOTIFICATION STARTS =====
20:44:26,481 11-Oct-2016 WARN  Type: VM_RECOVERY_COMPLETE
20:44:26,481 11-Oct-2016 WARN  Status: FAILURE
20:44:26,481 11-Oct-2016 WARN  Status Code: 500
20:44:26,481 11-Oct-2016 WARN  Status Msg: Recovery: Recovery completed with errors
```

複数の OpenStack VIM への VNF の展開

ESC を使用して、同じタイプの複数の VIM に VNF を展開できます。ESC は、複数の OpenStack VIM での VNF の展開をサポートします。OpenStack の単一インスタンスに VM を展開するには、OpenStack での仮想ネットワーク機能の展開 （115 ページ）を参照してください。

VNF を複数の VIM に展開するには、次の手順を実行する必要があります。

- VIM コネクタとそのクレデンシャルを設定します。
- ESC 内にテナントを作成する

VIM コネクタは VIM を ESC に登録します。VNF を複数の VIM に展開するには、VIM の各インスタンスに VIM コネクタとそのクレデンシャルを設定する必要があります。VIM コネクタは、インストール時に bootvm.py パラメータを使用するか、VIM コネクタ API を使用して設定できます。デフォルトの VIM コネクタは、単一の VIM 展開に使用されます。マルチ VIM 展開では、VIM コネクタを指定するためにロケータ属性が使用されます。

通常、マルチ VIM 展開をサポートする ESC は以下を備えています。

- ESC がリソースを作成および管理するデフォルトの VIM
- 展開のみがサポートされているデフォルト以外の VIM

詳細については、VIM コネクタの管理 （51 ページ）を参照してください。

（vim_mapping 属性が false に設定されている）ESC 内のテナントであるデータモデル階層内のルートテナントと、ロケータ属性内に配置されたアウトオブバンド VIM テナントが、複数の VIM に VNF を展開するために使用できる必要があります。ルートテナントが存在しない場合、ESC はマルチ VIM 展開中にテナントを作成できます。複数の ESC テナントを作成できます。ユーザは、複数の VIM に複数のテナントを使用できます。詳細については、テナントの管理 （21 ページ）を参照してください。

マルチ VIM 展開では、VM グループごとにターゲット VIM を指定できます。各 VM グループを異なる VIM に展開できますが、VM グループ内の VM は同じ VIM に展開されます。

マルチ VIM 展開を有効にするには、データモデルの VM グループにロケータ属性を追加する必要があります。ロケータノードは、次の属性で構成されます。

（注） ロケータ属性が展開に存在する場合、VM はロケータで指定された VIM に展開されます。ロケータ属性が展開に存在しない場合、VM はデフォルトの VIM に展開されます。デフォルトの VIM も存在しない場合、要求は拒否されます。

- vim_id：ターゲット VIM の VIM ID。ESC は vim_id を定義し、vim_connector ID にマッピングします。VIM コネクタは、vim_id で指定された VIM に展開する前に存在している必要があります。
- vim_project：ターゲット VIM で作成されたテナント名。これは、OpenStack に存在するアウトオブバンドテナントまたはプロジェクトです。

（注） ESC は、マルチ VIM 展開でポート、イメージ、フレーバ、ボリュームなどのアウトオブバンドリソース（既存のリソース）のみをサポートします。アウトオブバンドポートは、展開と同じテナントによって作成する必要があります。

ただし、マルチ VIM 展開では、デフォルト以外の VIM でロケータ属性を使用してエフェメラルボリュームのみを作成できます。その他のリソースは、デフォルト以外の VIM では作成できません。

VM のリカバリ、VM のスケールインとスケールアウトは、VM が展開されている同じ VIM 内でサポートされます。異なる VIM で VM を拡張またはリカバリすることはできません。

次の例では、esc-tenant は ESC 内のテナントです。VIM テナントへのマッピングはなく、VM はこの esc-tenant に展開されません。アウトオブバンドで作成される vim_project、project-test-tenant（ロケータ属性内）は、VM が展開されているテナントです。

```
<tenants>
   <tenant>
      <name>esc-tenant</name>
      <deployments>
         <deployment>
            <name>dep-1</name>
            <vm_group>
               <name>group-1</name>
               <locator>
                  <vim_id>vim-1</vim_id>
                  <vim_project>project-test-tenant</vim_project>
               </locator>
            </vm_group>
         </deployment>
      </deployments>
   </tenant>
</tenants>
```

ロケータ属性を使用して、単一の VIM に VNF を導入することもできます。つまり、ロケータ属性を持つデータモデルは、単一の OpenStack VIM で VM を導入するためにも使用できます。ロケータ属性（ESC リリース 2.x データモデル）なしで展開するには、単一の OpenStack VIM での VNF の展開 （116 ページ）を参照してください。

展開データモデルは次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"
xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications">
   <tenants>
      <tenant>
         <name>test-esc-tenant1</name>
         <deployments>
            <deployment>
               <name>dep-1</name>
               <vm_group>
                  <name>g1</name>
                  <locator>
                     <vim_id>vim1</vim_id>
```

```
                    <vim_project>project-test</vim_project>
                 </locator>
                 <bootup_time>150</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>my-network</network>
                    </interface>
                 </interfaces>
                 <scaling>
                    <min_active>1</min_active>
                    <max_active>1</max_active>
                    <elastic>true</elastic>
                 </scaling>
                 <kpi_data>
                    <kpi>
                       <event_name>VM_ALIVE</event_name>
                       <metric_value>1</metric_value>
                       <metric_cond>GT</metric_cond>
                       <metric_type>UINT32</metric_type>
                       <metric_collector>
                          <type>ICMPPing</type>
                          <nicid>0</nicid>
                          <poll_frequency>3</poll_frequency>
                          <polling_unit>seconds</polling_unit>
                          <continuous_alarm>false</continuous_alarm>
                       </metric_collector>
                    </kpi>
                 </kpi_data>
                 <rules>
                    <admin_rules>
                       <rule>
                          <event_name>VM_ALIVE</event_name>
                          <action>ALWAYS log</action>
                          <action>TRUE servicebooted.sh</action>
                          <action>FALSE recover autohealing</action>
                       </rule>
                    </admin_rules>
                 </rules>
                 <config_data />
              </vm_group>
           </deployment>
        </deployments>
     </tenant>
  </tenants>
</esc_datamodel>
```

アウトオブバンドリソースを使用し、展開の一部としてルートテナントを作成するサンプルのマルチ VIM 展開データモデル。

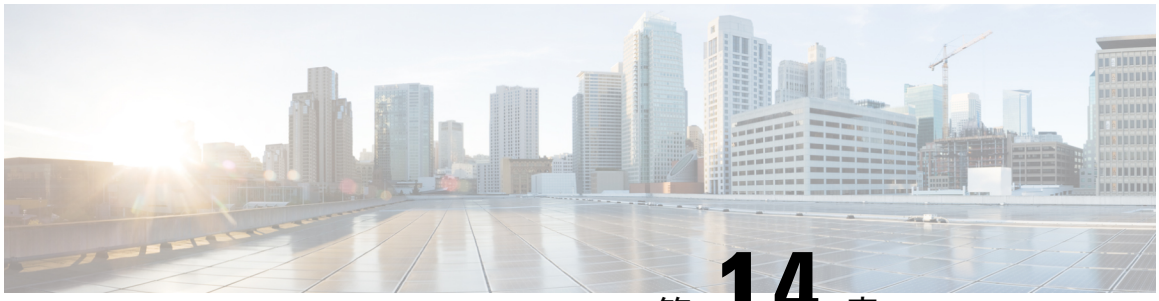
```
<esc_datamodel>
    <tenants>
        <tenant>
            <!-- This root level tenant is an ESC tenant either previously created or
created here marked by vim_mapping atrribute. -->
            <name>esc-tenant-A</name>
            <vim_mapping>false</vim_mapping>
            <deployments>
                <deployment>
                    <name>dep-1</name>
                    <vm_group>
```

```
                        <name>Grp-1</name>
                        <locator>
                            <vim_id>SiteA</vim_id>
                            <!-- vim_project: OOB project/tenant that should already
exist in the target VIM -->
                            <vim_project>Project-X</vim_project>
                        </locator>
                        <!-- All other details in vm group remain the same. -->
                        <flavor>Flavor-1</flavor>
                        <image>Image-1</image>
            ...
            ...
          </vm_group>
                </deployment>
            </deployments>
        </tenant>
    </tenants>
</esc_datamodel>
```

ESC が要求を受け入れるには、マルチ VIM 展開で指定されたすべての VIM が設定され、CONNECTION_SUCCESSFUL ステータスである必要があります。展開で指定された VIM が到達不能またはその他のステータスである場合、要求は拒否されます。

マルチ VIM 展開の VM にはアフィニティルールとアンチアフィニティルールを適用できます。詳細については、OpenStack のアフィニティルールとアンチアフィニティルール （207 ページ）を参照してください。

マルチ VIM 展開は、ライフサイクルステージ（LCS）を使用したリカバリをサポートします。サポートされている LCS の詳細については、リカバリポリシー（ポリシーフレームワークを使用） （351 ページ）を参照してください。既存のマルチ VIM 展開を更新できます。ただし、VM グループ内のロケータ属性は更新できません。既存の展開の更新に関する詳細については、既存の展開の更新 （225 ページ）を参照してください。



第 14 章

複数の VIM への仮想ネットワーク機能の展開

複数の VIM への仮想ネットワーク機能の展開

ここでは、Elastic Services Controller（ESC）の展開シナリオと、OpenStack、Cisco Cloud Services Platform（CSP）、vCloud Director（vCD）などのさまざまなタイプの VIM を展開する手順について説明します。

（注）
- ESC テナントは、マルチ VIM タイプの展開に必要です。
- 同じ展開間アンチアフィニティからの展開は、すべての vm_group で同じ VIM に展開する必要があります。

次の表に、ESC VM および VNF 展開の VIM タイプでサポートされるマトリックスを示します。

表 *5 : ESC VM* および *VNF* 展開の *VIM* タイプのサポートマトリックス

ESC VM のインストール先	OpenStack	vCloud Director	Cisco Cloud Services Platform
OpenStack	サポート対象	サポート対象	サポート対象
VMware vCenter	サポート対象	サポート対象	サポート対象

サンプルの展開モデル

```
<?xml version="1.0"?>
```

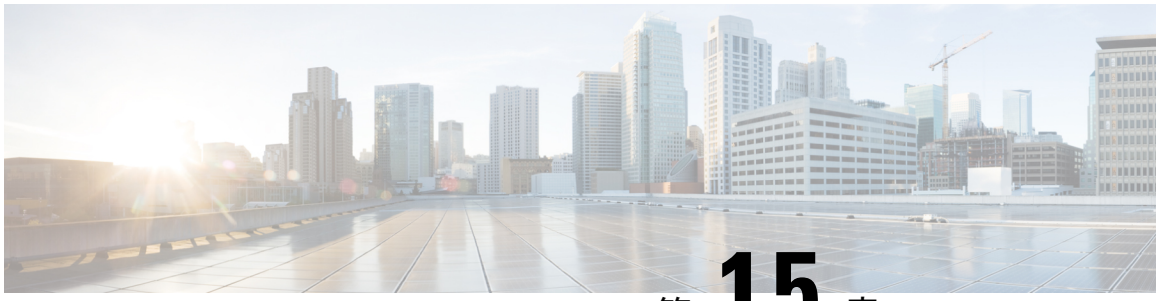
```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>VCDNCTestMVTypeDeployment-Tenant</name>
      <vim_mapping>false</vim_mapping>
      <deployments>
        <deployment>
          <name>VCDNCTestMVTypeDeployment-Dep</name>
          <vm_group>
            <name>VCDNCTestMVTypeDeployment-VCD-Group</name>
            <vim_vm_name>jenkins-VCDNCTestMVTypeDeployment-VCD-VM</vim_vm_name>
            <locator>
              <!-- vCD vim connector id -->
              <vim_id>VCD1</vim_id>
              <!-- vCD orgnization -->
              <vim_project>VAR{CFG{TARGET_LAB_0}:VCD_ORG1}</vim_project>
              <!-- vDC name -->
              <vim_vdc>VAR{CFG{TARGET_LAB_0}:VCD_ORG1_VDC1}</vim_vdc>
            </locator>
            <interfaces>
              <interface>
                <nicid>0</nicid>
                <network>VAR{CFG{TARGET_LAB_0}:VCD_MGT_NETWORK}</network>

<ip_address>VAR{CFG{TARGET_LAB_0}:VCD_MGT_NETWORK_IP_BASE}.VAR{CFG{TARGET_LAB_0}:STATIC_IP_RANGE}.0.2</ip_address>

              </interface>
            </interfaces>
          </vm_group>
          <vm_group>
            <name>VCDNCTestMVTypeDeployment-OS-Group</name>
            <vim_vm_name>jenkins-VCDNCTestMVTypeDeployment-OS-VM</vim_vm_name>
            <locator>
              <vim_id>Openstack1</vim_id>
              <!-- VIM Project = OOB Tenant -->

<vim_project>REPLACE_WITH_GENERATED_OOB_PROJECT_NAME_FOR_CFG{TARGET_LAB_1}</vim_project>

            </locator>
            <interfaces>
              <interface>
                <nicid>0</nicid>
                <network>VAR{CFG{TARGET_LAB_1}:MANAGEMENT_NETWORK}</network>
              </interface>
              <interface>
                <nicid>1</nicid>
                <network>VCDNCTestMVTypeDeployment-Net-2</network>
              </interface>
            </interfaces>
          </vm_group>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```

マルチ VIM 展開でサポートされる機能

次の表に、マルチ VIM 展開環境でサポートされるすべての機能を示します。

表 *6 :* マルチ *VIM* 展開でサポートされる機能

機能	OpenStack	Cisco Cloud Services Platform	vCloud Director
複数の VM グループによる展開	サポート対象	サポート対象	サポート対象
単一のVMグループによるマルチ展開	サポート対象	サポート対象	サポート対象
展開の拡大縮小	サポート対象	サポート対象	サポート対象
展開の更新	サポート対象	サポート対象	サポート対象
エフェメラルネットワーク用の VIM ロケータ	サポート対象	サポート対象	サポート対象
リカバリ	サポート対象	サポート対象	サポート対象
LCS 通知	サポート対象	サポート対象	サポート対象
VM 操作 start/stop/reset/disable_monitor/enable_monitor	サポート対象	サポート対象	サポート対象
OpenStack での ESC（デフォルト VIM あり/なし）	サポート対象	サポート対象	サポート対象



第 15 章

既存環境への導入

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

ブラウンフィールド展開：

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

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

ブラウンフィールド展開のクイックスタート：

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

ブラウンフィールドデータの作成：

データを作成し、エラーを修正します（エラーが返された場合）。

ペイロードの例

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

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

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

VNF の展開

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

ペイロードの例：

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

展開の完了：

この状態では、ESC VM が VNF を管理します。

ペイロードの例：

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

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

インポートデータの削除：

（注） 次の手順は省略可能です。

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

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

必要なファイル：

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

- 展開 XML：

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

- インポート XML：

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

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

chassisID と user_key：

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

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

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

サポートされている API：

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

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

ConfD：ブラウンフィールド CREATE：

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

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

使用例：

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

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

検証：

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

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

ConfD または ESCManager REST API：DEPLOY

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

使用例：

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

検証：

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

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

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

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

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

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

ConfD：ブラウンフィールド FINALIZE

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

テナント名と展開名の 2 つの必須引数が必要です。

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

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

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

検証

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

使用例：

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

ConfD：ブラウンフィールド DELETE インポートデータ

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

テナント名と展開名の 2 つの必須引数が必要です。

使用例

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

検証

「ファイナライズ済み」のブラウンフィールド展開のみデータを削除できます。

インポート XML の例：

以下は、インポート XML ファイルまたはデータスニペットの例の一部です。

基本 VM、およびエフェメラルボリュームとポート：

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

基本 VM

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

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

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

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

関連のない構造を除いた、関連する展開 XML は次のように表示されます。

元の展開 XML

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

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

基本 VM とエフェメラルネットワーク：

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

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

基本 VM

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

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

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

関連のない構造を除いた、関連する展開 *XML* を以下に示します。

元の展開 XML

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

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

スケールアウトされた展開：

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

ペイロードの例：

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

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

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

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

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

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

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

ChassisID とユーザーキーの仕様：

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

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

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

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

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

ペイロードの例：

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

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

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

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

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

関連のない構造を除いた、関連する展開 XML を以下に示します。

元の展開 XML

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

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

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

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

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

ライフサイクル管理スクリプトの実行：

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

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

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

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

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

制限事項：

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

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

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

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

ブラウンフィールドデータの生成：

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

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

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

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

- **展開 XML**

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

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

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

- **インポート XML**

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

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

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

- **スクリプトを使用したインポートおよび展開 XML の生成**：

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

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

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

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

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

[使用状況（Usage）] ページの出力を次に示します。

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

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

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

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

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

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

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

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

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

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

                          Incorrectly specified policies will be ignored.  Examples:

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

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

                          Specify the path using an absolute path.  Example:

                          -c /tmp/target/data
```

使用例

export_brownfield_data.py：使用例

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

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

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

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

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

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

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

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

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

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

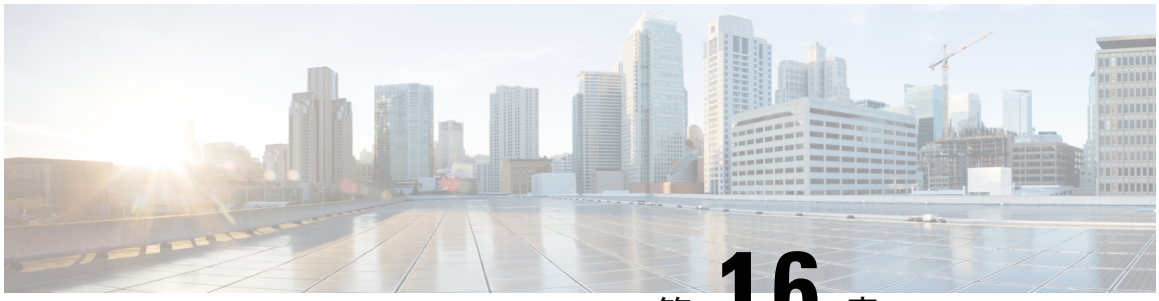
ソース ESC VM からの管理の削除：

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

考えられる 4 つの手法は次のとおりです。

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

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



第 16 章

VMwareでの仮想ネットワーク機能の展開

VMware vCenter のイメージ

アウトオブバンドイメージの定義を使用して VNF を展開できます。次の表に展開シナリオを示します。

シナリオ	説明	データモデルテンプレート	画像	利点
ESC を使用したイメージの作成による VNF の展開 **重要** イメージは、VMware vCenter のテンプレートとも呼ばれます。	VNF 展開のプロセスは次のとおりです。 1. VNF 展開：展開データモデルは、作成されたイメージを参照して、VNF を展開します。	• 展開データモデル • イメージデータモデル	イメージは、REST API を使用して ESC で作成されます。	• イメージは、複数の VNF 展開で使用できます。 • ESC を使用してイメージ定義を追加または削除できます。

シナリオ	説明	データモデルテンプレート	画像	利点
アウトオブバンドイメージを使用した単一 VIM への VNF の展開	1. VNF 展開：展開データモデルは、VMware vCenter のアウトオブバンドイメージを参照して、VNF を展開します。	• 展開データモデル • VMware vCenter のイメージ	ESC を使用してイメージを作成または削除することはできません。	• イメージは、複数の VNF 展開で使用できます。 • ESC ポータルからイメージを確認できます。 • アウトオブバンド展開中に、イメージを選択できます。

（注） ESC は、ESC 5.8 リリース以降の VIM タイプの VMware vSphere に対する IPv6 展開をサポートしますが、デュアルスタック ネットワークの作成はサポートされないという制限があります。つまり、IPv4 または IPv6 サブネットのいずれかは作成されますが、両方は**作成されません**。

VMware vCenter VIM での VNF の展開

ここでは、Cisco Elastic Services Controller の展開シナリオと、VMware に VNF を展開する手順について説明します。

VNF の展開は、ESC ポータルまたはノースバウンド インターフェイスから発信されるサービス要求として開始されます。サービス要求は XML ペイロードで構成されます。ESC は、次の展開シナリオをサポートします。

- ESC を介したリソースの作成による VNF の展開
- アウトオブバンドリソースを使用した VNF の展開

VNF を展開する前に、リソースが VMware vCenter で使用可能であることを確認するか、これらのリソースを作成する必要があります。リソース管理の概要 （17 ページ）を参照してください。展開中、ESC は展開データモデルで展開の詳細を検索します。展開データモデルの詳細については、「Cisco Elastic Services Controller Deployment Attributes」を参照してください。

（注） 単一の ESC インスタンスは、 VIM ロケータ ごとに 1 つの vCenter Distributed Switch（vDS）のみをサポートします。

- vDS には、クラスタ化された 1 つ以上の ESXi ホストが含まれます。
- ESXi ホストが 1 つのコンピューティングクラスタの下にある場合、DRS がオンの場合は [自動化レベル（Automation Level）] を [手動（Manual）] に設定する必要があります。
- クラスタ化されたデータストアはサポートされていません。
- ホストがクラスタ化されている場合は、クラスタまたはデータセンターの下のフラットなデータストアのみがサポートされます。

ESCはデフォルトのリソースプールのみをサポートします。リソースプールを追加または作成することはできません。「ネットワーキングの設定操作がロールバックされ、ホストがvCenterサーバから切断されています」という内容のエラーメッセージが表示された場合は、vCenterの制限が原因です。データストアの自動選択は次のように機能します。

- ESCは最初にホストを選択します。展開がクラスタを対象としている場合、ホストはコンピューティングホストの容量に対するVMの数の比率に基づいて選択されます。それ以外の場合は、ホストを対象とする展開で要求されるとおりにホストが選択されます。
- データストアはその空き領域に基づいてホストから選択されます。

VMware vCenterのリカバリの一環として再展開が行われるたびに、VMのインターフェイスに異なる MAC アドレスが割り当てられます。

VM への OVF プロパティの受け渡し

VMware vCenter での VNF の展開の一環として、名前と値のペアを OVF プロパティとして VM に渡すことができます。VNFの展開中にこれらの設定を渡すには、展開データモデルのテンプレートに追加の引数を含める必要があります。

サンプル設定を次に示します。

```
   <esc_datamodel ...>
    ...
    <config_data>
   <configuration>
      <dst>ovfProperty:mgmt-ipv4-addr</dst>
      <data>$NICID_1_IP_ADDRESS/24</data>
   </configuration>
   <configuration>
      <dst>ovfProperty:com.cisco.csr1000v:hostname</dst>
      <data>$HOSTNAME</data>
      <variable>
         <name>HOSTNAME</name>
         <val>csrhost1</val>
         <val>csrhost2</val>
      </variable>
   </configuration>
</config_data>
```

```
  ...
</esc_datamodel>
```

複数の仮想データセンター（マルチ VDC）での VNF の展開

仮想データセンター（VDC）は、仮想リソース、動作の詳細、ルール、およびポリシーを組み合わせて特定のグループの要件を管理します。グループは、複数の VDC、イメージ、テンプレート、およびポリシーを管理できます。このグループは個々のグループに VDC レベルでクォータを割り当て、リソース制限を割り当てることができます。

ESC ポータルで使用可能な VDC のリストを表示するには、[データセンター（Datacenters）] を選択します。

はじめる前に

複数の VDC に VNF を展開する前に、次の条件が満たされていることを確認します。

- 両方の VDC にまたがる標準外部ネットワークを使用して、ESC が展開された VM に ping を実行できることを確認します。
- VM の少なくとも 1 つの管理インターフェイスが外部ネットワークに接続されていることを確認します。
- VDC が vCenter に存在することを確認します。

（注）
- ESC は、VDC で作成する必要があるすべてのリソースが帯域外であり、VDC 内に存在することを前提としています。
- 現在、ESC は vCenter に存在する任意の VDC に展開できます。ESC が展開できる VDC には範囲や制限はありません。

VNF を展開する場合は、VNF をプロビジョニングする必要がある仮想データセンターのロケータ名を指定する必要があります。

配置要求では、リソースを作成および削除するためのロケータ要素が導入されます。

ロケータ要素には次のものが含まれます。

- データセンター名のタグ：リソース（展開、イメージ、ネットワーク、およびサブネット）のターゲット VDC を指定します。
- switch_name：ネットワークを関連付けるターゲット VDS を指定します。

ロケータ要素を使用すると、以下を実行できます。

- ロケータ内でデータセンター属性を指定することで、別の VDC でイメージまたはテンプレートを作成できます。次の例を参考にしてください。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
    <images>
        <image>
            <name>automated-uLinux</name>
```

```
            <src>http://VAR_FILE_SERVER_IP/share/images/uLinux/uLinux.ovf</src>
            <locators>
                <datacenter>VAR_VDC2</datacenter>
            </locators>
        </image>
    </images>
</esc_datamodel>
```

- VDC からネットワークを作成および削除できます。

（注） ネットワークが統合型の展開の一部である場合、データセンター属性は展開要求の展開属性から取得されます。

```
<network>
    <locators>
        <datacenter>DC-03</datacenter>
        <switch_name>dvSwitch</switch_name>
    </locators>
    <name>test-yesc-net-u</name>
    <shared>false</shared>
    <admin_state>true</admin_state>
</network>
```

Cisco Elastic Services Controller ポータルでは、VM をプロビジョニングする VDC を選択できます。サービス要求を作成するとき、この VM をプロビジョニングする VDC を選択できます。

ESC 運用データの *default_locators* コンテナは、ESC で設定されたデフォルトのロケータを示しますが、複数の Center VIM を設定することができます。

（注） ロケータが設定されていない場合、*default_locators* コンテナは表示されません。

運用データの例は次のとおりです。

```
Operational Data
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=172.16.0.1 --user=admin
--privKeyFile=/var/confd/homes/admin/.ssh/confd_id_dsa --privKeyType=dsa --get -x
"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">
    <data>
        <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
            <opdata>
                <status>OPER_UP</status>
                <stats>
                    <hostname>test-ESC-host</hostname>
                    <os_name>Linux</os_name>
                    <os_release>2.6.32-573.22.1.el6.x86_64</os_release>
                    <arch>amd64</arch>
                    <uptime>9481</uptime>
                    <cpu>
                        <cpu_num>4</cpu_num>
                    </cpu>
                </stats>
                <system_config>
```

```
                    <active_vim>VMWARE</active_vim>
                    <vmware_config>
                        <vcenter_ip>172.16.1.0</vcenter_ip>
                        <vcenter_port>80</vcenter_port>
                        <vcenter_username>root</vcenter_username>
                    </vmware_config>
                </system_config>
                <default_locators>
                    <datacenter>DC-4</datacenter>
                </default_locators>
                <tenants>
                    <tenant>
                        <name>admin</name>
                        <tenant_id>SystemAdminTenantId</tenant_id>
                    </tenant>
                </tenants>
            </opdata>
        </esc_datamodel>
    </data>
</rpc-reply>
[admin@test-ESC-host esc-cli]$
```

VMware vCloud Director（vCD）での仮想ネットワーク機能の展開

ここでは、ESC の展開シナリオと、VMware vCloud Director（vCD）に VNF を展開する手順について説明します。vCD に ESC をインストールする場合は、Cisco Elastic Services Controller インストールおよびアップグレードガイド [英語] を参照してください。

組織や組織 VDC などのリソースは、展開前に vCD で作成する必要があります。詳細については、vCloud Director（vCD）のリソースの管理 （49 ページ）を参照してください。

VNF を展開するには、次の手順を実行する必要があります。

1. VMware vCD で事前設定済みの組織および組織ユーザの詳細とともに、VIM コネクタを追加します。「VMware vCloud Director（vCD）の VIM コネクタの設定」を参照してください。

   ロケータの下にある vim_vdc リーフは、展開のターゲットとなる vDC を参照します。

2. VMware vCD で事前設定済みの組織 VDC、カタログ、および vApp テンプレートパラメータを使用して VNF を展開します。

これらのリソースを作成する場合は、VMware vCloud Director のマニュアルを参照してください。

VNF を vCD に展開する前に、次の主要なパラメータを設定する必要があります。

- VMWARE_VCD_PARAMS：各展開セクションのデータモデルの拡張セクションに VMWARE_VCD_PARAMS パラメータを指定します。VMWARE_VCD_PARAMS パラメータには、CATALOG_NAME と VAPP_TEMPLATE_NAME が含まれます。

- CATALOG_NAME：vApp テンプレートおよびメディアイメージへの参照を含む事前設定済みのカタログの名前を指定します。
- VAPP_TEMPLATE_NAME：オペレーティングシステム、アプリケーション、およびデータとともにロードされる仮想マシンイメージを含む事前設定済みの vApp テンプレートの名前を指定します。これにより、仮想マシンが組織全体で一貫して設定されます。

展開例は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"
xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications">
   <tenants>
      <tenant>
         <!-- ESC scope tenant -->
         <name>esc-tenant</name>
         <vim_mapping>false</vim_mapping>
         <deployments>
            <deployment>
               <!-- vApp instance name -->
               <name>vapp-inst1</name>
               <policies>
                  <placement_group>
                     <name>placement-anti-affinity</name>
                     <type>anti_affinity</type>
                     <enforcement>strict</enforcement>
                     <vm_group>g1</vm_group>
                     <vm_group>g2</vm_group>
                  </placement_group>
               </policies>
               <extensions>
                  <extension>
                     <name>VMWARE_VCD_PARAMS</name>
                     <properties>
                        <property>
                           <name>CATALOG_NAME</name>
                           <value>catalog-1</value>
                        </property>
                        <property>
                           <name>VAPP_TEMPLATE_NAME</name>
                           <value>uLinux_vApp_Template</value>
                        </property>
                     </properties>
                  </extension>
               </extensions>
               <vm_group>
                  <name>g1</name>
                  <locator>
                     <!-- vCD vim connector id -->
                     <vim_id>vcd_vim</vim_id>
                     <!-- vCD organization corresponding to the vim connector -->
                     <vim_project>organization</vim_project>
                     <!-- vDC pre-preconfigured in organization -->
                     <vim_vdc>VDC-1</vim_vdc>
                  </locator>
                  <!-- VM name in vAppTemplate -->
                  <image>vm-001</image>
                  <bootup_time>150</bootup_time>
```

```
    <recovery_wait_time>30</recovery_wait_time>
    <interfaces>
       <interface>
          <nicid>0</nicid>
          <network>MgtNetwork</network>
          <ip_address>172.16.0.0</ip_address>
       </interface>
    </interfaces>
    <scaling>
       <min_active>1</min_active>
       <max_active>1</max_active>
       <elastic>true</elastic>
       <static_ip_address_pool>
          <network>MgtNetwork</network>
          <ip_address>172.16.0.0</ip_address>
       </static_ip_address_pool>
    </scaling>
    <kpi_data>
       <kpi>
          <event_name>VM_ALIVE</event_name>
          <metric_value>1</metric_value>
          <metric_cond>GT</metric_cond>
          <metric_type>UINT32</metric_type>
          <metric_collector>
             <type>ICMPPing</type>
             <nicid>0</nicid>
             <poll_frequency>3</poll_frequency>
             <polling_unit>seconds</polling_unit>
             <continuous_alarm>false</continuous_alarm>
          </metric_collector>
       </kpi>
    </kpi_data>
    <rules>
       <admin_rules>
          <rule>
             <event_name>VM_ALIVE</event_name>
             <action>"ALWAYS log"</action>
             <action>"TRUE servicebooted.sh"</action>
             <action>"FALSE recover autohealing"</action>
          </rule>
       </admin_rules>
    </rules>
    <config_data>
       <configuration>
          <dst>ovfProperty:mgmt-ipv4-addr</dst>
          <data>$NICID_0_IP_ADDRESS/24</data>
       </configuration>
    </config_data>
 </vm_group>
 <vm_group>
    <name>g2</name>
    <locator>
       <!-- vCD vim connector id -->
       <vim_id>vcd_vim</vim_id>
       <!-- vCD organization corresponding to the vim connector -->
       <vim_project>organization</vim_project>
       <!-- vDC pre-preconfigured in organization -->
       <vim_vdc>VDC-1</vim_vdc>
  <locator>
     <vim_id>vcenter-22</vim_id>
     <vim_vdc>OTT-ESC-10</vim_vdc>
     </locator>
    </locator>
    <!-- VM name in vAppTemplate -->
```

```
                        <image>vm-002</image>
                        <bootup_time>150</bootup_time>
                        <recovery_wait_time>30</recovery_wait_time>
                        <interfaces>
                           <interface>
                              <nicid>0</nicid>
                              <network>MgtNetwork</network>
                              <ip_address>172.16.0.1</ip_address>
                           </interface>
                        </interfaces>
                        <scaling>
                           <min_active>1</min_active>
                           <max_active>1</max_active>
                           <elastic>true</elastic>
                           <static_ip_address_pool>
                              <network>MgtNetwork</network>
                              <ip_address>172.16.0.1</ip_address>
                           </static_ip_address_pool>
                        </scaling>
                        <kpi_data>
                           <kpi>
                              <event_name>VM_ALIVE</event_name>
                              <metric_value>1</metric_value>
                              <metric_cond>GT</metric_cond>
                              <metric_type>UINT32</metric_type>
                              <metric_collector>
                                 <type>ICMPPing</type>
                                 <nicid>0</nicid>
                                 <poll_frequency>3</poll_frequency>
                                 <polling_unit>seconds</polling_unit>
                                 <continuous_alarm>false</continuous_alarm>
                              </metric_collector>
                           </kpi>
                        </kpi_data>
                        <rules>
                           <admin_rules>
                              <rule>
                                 <event_name>VM_ALIVE</event_name>
                                 <action>"ALWAYS log"</action>
                                 <action>"TRUE servicebooted.sh"</action>
                                 <action>"FALSE recover autohealing"</action>
                              </rule>
                           </admin_rules>
                        </rules>
                        <config_data>
                           <configuration>
                              <dst>ovfProperty:mgmt-ipv4-addr</dst>
                              <data>$NICID_0_IP_ADDRESS/24</data>
                           </configuration>
                        </config_data>
                     </vm_group>
                  </deployment>
               </deployments>
            </tenant>
         </tenants>
      </esc_datamodel>
```

vCD で設定された VM 配置ポリシーを利用するには、vAppTemplate でポリシーを *Mobile* に設定する必要があります。次の配置データモデルを使用します。

```
<vm_group>
  <name>vm_grp1</name>
  …
  <placement>
```

```
        <type>vm_policy</type>
        <enforcement>strict</enforcement>
        <policy>Test-VM-Placement-Policy-2</policy>
      </placement>
      …
    </vm_group>
```

一意のポリシー名を指定すると、そのポリシーを使用してVMのターゲットホストを決定するようにvCDに通知します。

第 17 章

Amazon Web Services での仮想ネットワーク機能の展開

Amazon Web Services での仮想ネットワーク機能の展開

ここでは、Elastic Services Controller（ESC）の展開シナリオと、Amazon Web Services（AWS）に VNF を展開する手順について説明します。AWS に ESC をインストールする場合は、*Cisco Elastic Services Controller* インストールおよびアップグレードガイド [英語] を参照してください。

展開前に、次の AWS リソースを AWS で作成する必要があります。

- Amazon マシンイメージ（AMI）
- キーペア
- Elastic IP
- セキュリティ グループ
- ネットワーク要素（VPC、サブネット、ACL、ゲートウェイ、ルートなど）

これらのリソースを作成するには、AWS のマニュアルを参照してください。

AWS 展開前の VIM コネクタ設定の詳細については、「AWS の VIM コネクタ設定」を参照してください。

シナリオ	説明	リソース	利点
ESC を使用した Amazon マシンイメージ（AMI）およびリージョンの作成による単一 VIM への VNF の展開	展開データモデルは、Amazon マシンイメージ（AMI）、フレーバ、AWS リージョン、キーペア、セキュリティグループ、ネットワークインターフェイス、および作成された VIM プロジェクトを参照し、VNFを展開します。	Amazon マシンイメージ（AMI）、フレーバ、AWS リージョン、キーペア、セキュリティグループ、ネットワークインターフェイス、VIMプロジェクト（ロケータで指定）、および ESC によって作成されたネットワーク。	• 展開内の ESC で設定する必要がある（VM を展開するための）VIMを指定できます。 • イメージとフレーバは、複数の VNF 展開で使用できます。 • ESCによって作成されたリソースを削除できます。
ESC を使用した AMI およびリージョンの作成による、複数のVIM への VNF の展開	展開データモデルは、Amazon マシンイメージ（AMI）、フレーバ、AWS リージョン、キーペア、セキュリティグループ、ネットワークインターフェイス、および作成された VIM プロジェクトを参照し、VNF を展開します。	イメージ、フレーバ、VIMプロジェクト（ロケータで指定）および ESC を使用して作成されたネットワーク。	展開内の ESC で設定する必要がある（VM を展開するための）VIM を指定できます。

詳細については、単一または複数の AWS リージョンでの VNF の展開 （158 ページ）を参照してください。

単一または複数の AWS リージョンでの VNF の展開

ESC を使用して、単一または複数の AWS リージョンまたは同じタイプの VIM に VNF を展開できます。

（注） AWS は ESC 用の仮想インフラストラクチャ マネージャ（VIM）です。このドキュメントでは、AWS リージョンと AWS VIM という用語は同じ意味で使用されています。

単一または複数の VIM に VNF を展開するには、次の手順を実行する必要があります。

- VIM コネクタ API を使用して VIM コネクタとそのログイン情報を設定する
- ESC 内にテナントを作成する

VIM コネクタは VIM を ESC に登録します。単一または複数の AWS VIM に VNF を展開するには、VIM のリージョンごとに VIM コネクタとそのログイン情報を設定する必要があります。VIM コネクタ API を使用して VIM コネクタを設定できます。詳細については、AWS の VIM コネクタ設定 （71 ページ）を参照してください。

（注） デフォルトの VIM コネクタは、AWS 展開ではサポートされていません。

ESC は、vim_mapping 属性が false に設定されている ESC 内にテナントを作成します。このテナントは、VIM から独立しています。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
   <tenants>
      <tenant>
         <name>aws-sample-tenant</name>
         <vim_mapping>false</vim_mapping>
      </tenant>
   </tenants>
</esc_datamodel>
```

単一または複数の AWS VIM 展開の場合、各 VM グループのターゲットリージョンを指定する必要があります。

AWS VIM 展開を有効にするには、データモデルの VM グループにロケータ属性を追加する必要があります。ロケータノードは、次の属性で構成されます。

- vim_id：ターゲット VIM の VIM ID。ESC は vim_id を定義し、vim_connector ID にマッピングします。VIM コネクタは、vim_id で指定された VIM に展開する前に存在している必要があります。
- vim_project：ターゲット VIM で作成されたテナント名。これは、OpenStack に存在するアウトオブバンドテナントまたはプロジェクトです。
- vim_region：VM グループが展開されている AWS リージョン。これはオプションです。VIM リージョンが指定されていない場合、VM は VIM コネクタで指定された aws_default_region に展開されます。

```
<locator>
   <vim_id>AWS_EAST_2</vim_id>
   <vim_region>us-east-1</vim_region>
   <!-- the deployment is going into
North Virginia -->
</locator>
```

VIM リージョンが指定されていない場合

```
<locator>
   <vim_id>AWS_EAST_2</vim_id>
   <!-- the deployment is going into the default region Ohio (us-east-2)
as defined in the VIM Connector example above -->
```

```
    </locator>
```

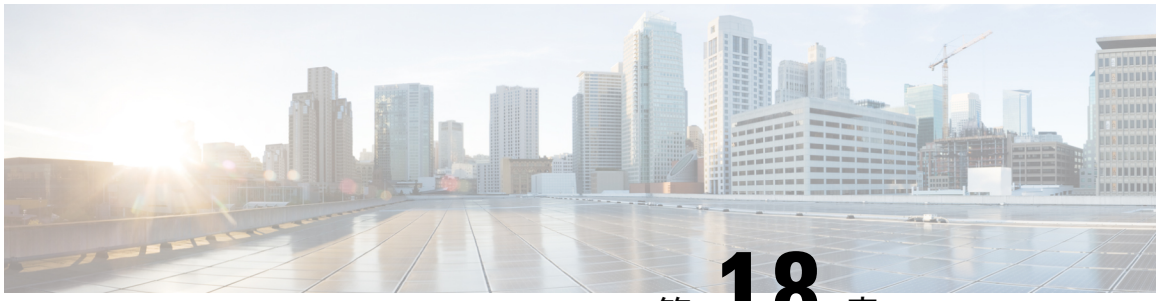
VIM コネクタとロケータを設定したら、特定のリソースを拡張機能として展開に渡す必要があります。次の例では、Elastic IP、キーペア、および送信元の宛先が拡張機能として AWS 展開に渡されます。

```
<extensions>
   <extension>
      <name>AWS_PARAMS</name>
      <properties>
         <property>
            <name>elastic_ip</name>
            <value>13.56.148.25</value>
         </property>
         <property>
            <name>source_dest_check</name>
            <value>true</value>
         </property>
         <property>
            <name>key_pair_name</name>
            <value>esc-us-east-1</value>
         </property>
      </properties>
   </extension>
</extensions>
```

AWS の展開例は次のとおりです。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
   <tenants>
      <tenant>
         <name>aws-east-1-tenant</name>
         <vim_mapping>false</vim_mapping>
         <deployments>
            <deployment>
               <name>aws-east-1-dep</name>
               <vm_group>
                  <name>aws-vm-east-1</name>
                  <locator>
                     <vim_id>AWS_US_EAST_1</vim_id>
                  </locator>
                  <bootup_time>600</bootup_time>
                  <recovery_wait_time>33</recovery_wait_time>
                  <flavor>t2.micro</flavor>
                  <image>ami-c7bfa6bd</image>
                  <extensions>
                     <extension>
                        <name>AWS_PARAMS</name>
                        <properties>
                           <property>
                              <name>key_pair_name</name>
                              <value>esc-us-east-1</value>
                           </property>
                        </properties>
                     </extension>
                  </extensions>
                  <interfaces>
                     <interface>
                        <nicid>0</nicid>
```

```
                        <network>vpc-d7ee1bac</network>
                        <security_groups>
                           <security_group>esc-sg-us-east-1</security_group>
                        </security_groups>
                     </interface>
                  </interfaces>
                  <kpi_data>
                     <kpi>
                        <event_name>VM_ALIVE</event_name>
                        <metric_value>1</metric_value>
                        <metric_cond>GT</metric_cond>
                        <metric_type>UINT32</metric_type>
                        <metric_collector>
                           <type>ICMPPing</type>
                           <nicid>0</nicid>
                           <poll_frequency>3</poll_frequency>
                           <polling_unit>seconds</polling_unit>
                           <continuous_alarm>false</continuous_alarm>
                           <monitoring_public_ip>true</monitoring_public_ip>
                        </metric_collector>
                     </kpi>
                  </kpi_data>
                  <rules>
                     <admin_rules>
                        <rule>
                           <event_name>VM_ALIVE</event_name>
                           <action>ALWAYS log</action>
                           <action>FALSE recover autohealing</action>
                           <action>TRUE servicebooted.sh</action>
                        </rule>
                     </admin_rules>
                  </rules>
                  <config_data />
                  <scaling>
                     <min_active>1</min_active>
                     <max_active>1</max_active>
                     <elastic>true</elastic>
                  </scaling>
               </vm_group>
            </deployment>
         </deployments>
      </tenant>
   </tenants>
</esc_datamodel>
```



第 18 章

CSP クラスタでの ESC を使用した VNF の展開

CSP クラスタでの ESC を使用した VNF の展開

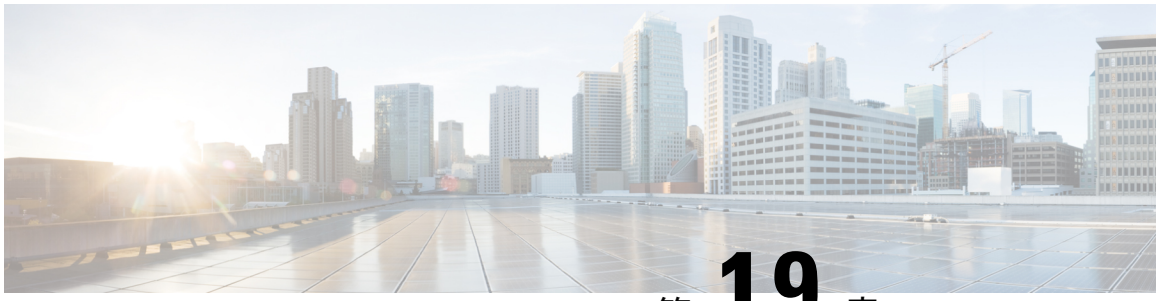
VNF の展開は、ESC ポータルまたはノースバウンド インターフェイスから発信されるサービス要求として開始されます。サービス要求は XML ペイロードで構成されます。

VNF を CSP に展開するには、ディスクストレージ名を（glusterFS）にします。デフォルトでは、ディスクストレージはローカルです。

イメージ拡張プロパティの下に、Gluster として disk_storage_name が必要です。クラスタ VIM コネクタを使用して初期展開を実行します。

次の例は、イメージ拡張プロパティの下に disk_storage_name として Gluster を追加する方法を示しています。

```
deploy_csp_1.xml
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
              <extension>
                <name>image</name>
                <properties>
                  <property>
                    <name>disk-resize</name>
                    <value>true</value>
                  </property>
                  <property>
                    <name>disk_type</name>
                    <value>virtio</value>
                  </property>
                  <property>
                    <name>disk_storage_name</name>
                    <value>gluster</value>
                  </property>
                </properties>
              </extension>
```



第 19 章

統合型の展開

統合型の展開

ESC は、VNF を展開する前に、テナント、ネットワーク、サブネットワークなどの OpenStack リソースを作成します。

統合型の展開中に、OpenStack リソースを作成または削除する単一の結合要求を送信し、VNF を展開します。複数のネットワークとサブネットワークを作成できますが、統合型の展開を使用して作成できる VNF とテナントは 1 つだけです。

統合型の展開要求は、新しい展開要求と、展開定義内に直接配置された任意の数のネットワークおよびサブネットワークとして定義されます。テナント内に直接配置されているネットワークおよびサブネットは、統合型の展開要求の一部とは見なされず、後続の展開解除要求時に削除されません。

サービスおよび展開 ID、テナント、ネットワーク、サブネットワーク ID などの必要な情報を使用して展開データモデルとファイルを更新します。NETCONF API または REST API を使用できます。たとえば、POST REST コールと DELETE REST コールを送信します。

（注） 単一の NETCONF 要求を使用して、ネットワークやサブネットワークの作成、イメージやフレーバーの作成、VNF の展開など、複数のアクションを実行できます。

展開属性のリストについては、Elastic Services Controller の展開属性 [英語] を参照してください。

- 単一の展開要求で展開データモデルを作成するには、POST REST コールを次の宛先に送信します。

  ```
  http://[ESC_IP]:8080/v0/deployments/[internal_dep_id]
  ```

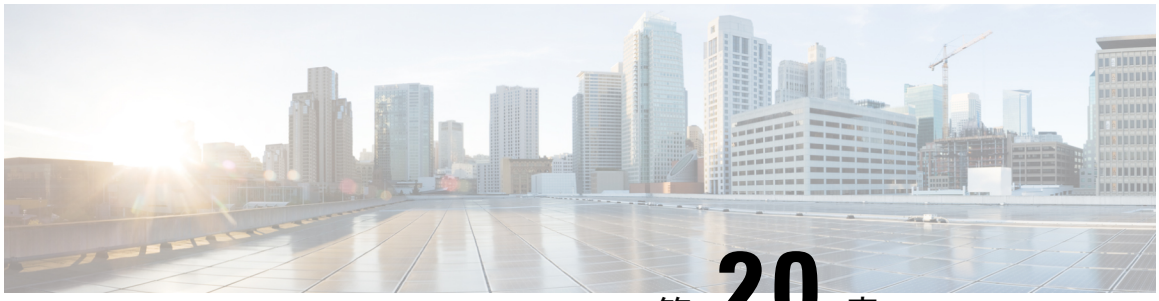
- 単一の展開要求を削除するには、DELETE REST コールを次の宛先に送信します。

  ```
  http://[ESC_IP]:8080/v0/deployments/[internal_dep_id]
  ```

VNF が展開解除され、ネットワークとサブネットが指定された順序で削除されます。

（注） 統合型の展開要求の一環としてテナントの作成が失敗した場合は、手動でロールバックして ESC をクリーンアップする必要があります。手動ロールバックの一環として、まず展開をクリーンアップするために展開解除要求が必要です。その後、失敗したテナント作成をクリーンアップするためのテナントの削除要求が行われます。

展開解除要求時に、統合型の展開要求の一環として作成されたネットワークとサブネットワークは、VNF とともに削除されます。ただし、統合型の展開要求によって作成されたテナントは削除されません。



第 20 章

仮想ネットワーク機能の展開解除

仮想ネットワーク機能の展開解除

すでに展開されている VNF の展開を解除できます。REST API またはNETCONF API/YANG API を使用して VNF を展開解除します。

重要 ESC ポータルを使用して VNF を展開解除することもできます。詳細については、「ESC ポータルダッシュボード」を参照してください。

展開解除要求のサンプルは次のとおりです。

```
DELETE /v0/deployments/567 HTTP/1.1
Host: client.host.com
Content-Type: application/xml
Accept: application/xml
Client-Transaction-Id: 123456
Callback:/undeployservicecallback
```

詳細については、『Cisco Elastic Services Controller API Guides』を参照してください。

再起動パラメータ

再起動時間パラメータが展開要求に導入されます。これにより、展開の動作時間がより柔軟になります。展開では、VMが再起動すると、モニタに再起動時間が設定されます。VM Aliveイベントの前に再起動時間が経過すると、vm_recovery_complete や undeploy などの次のアクションが実行されます。

第 21 章

展開パラメータの設定

導入パラメータ

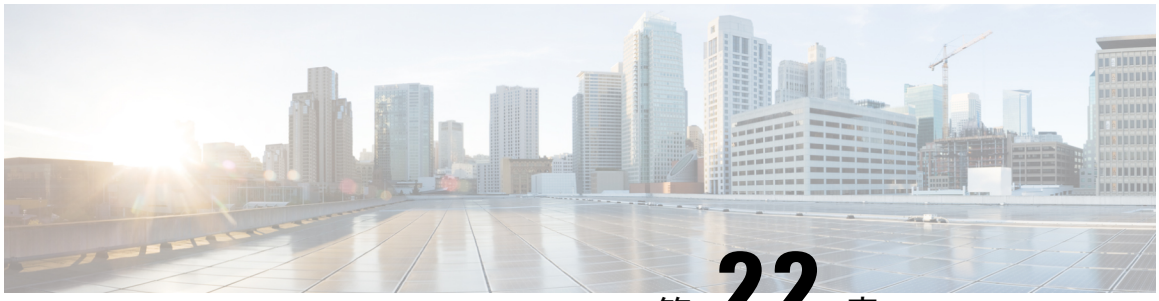
VNF 展開は、ノースバウンドインターフェイスまたは ESC ポータルを介してサービスリクエストとして開始されます。サービスリクエストは、XML ペイロードと展開パラメータから成るテンプレートで構成されます。展開パラメータは、VNF とそのライフサイクルのプロパティを決定するルール、ポリシー、またはデイゼロ設定です。次の表に、展開パラメータの完全なリストと、OpenStack または VMware vCenter での相互運用方法を示します。

導入パラメータ	OpenStack	VMware vCenter	VMware vCloud Director
デイゼロ設定	デイゼロ設定は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル	デイゼロ設定は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル	• NETCONF API • REST API • ETSI API
VNF の展開	個別および複合 VNF の設定は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル（展開テンプレートを使用して展開できます）。	個別および複合 VNF の設定は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル（VNF 設定は、展開フォームまたは展開テンプレートを使用して設定できます）。	個別および複合 VNF の設定は、次のいずれかの方法で行います。 • NETCONF API • REST API • ETSI API

導入パラメータ	OpenStack	VMware vCenter	VMware vCloud Director
仮想ネットワーク機能の展開解除	展開解除は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル	VNF の展開解除は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル	VNF の展開解除は、次のいずれかの方法で行います。 • NETCONF API • REST API • ETSI API
アフィニティルールとアンチアフィニティルール	アフィニティルールとアンチアフィニティルール定義の作成と削除は、次のいずれかの方法で行います。 • NETCONF API • REST API	アフィニティルール定義の作成と削除は、次のいずれかの方法で行います。 • NETCONF API • REST API • ESC ポータル（展開フォームを使用してアフィニティとアンチアフィニティを設定できます）。	アフィニティルールとアンチアフィニティルール定義の作成と削除は、次のいずれかの方法で行います。 • NETCONF API • REST API • ETSI API
VNF 操作	VNF 操作は、次のいずれかの方法で行います。 • REST API • NETCONF API • ESC ポータル	VNF 操作は、次のいずれかの方法で行います。 • REST API • NETCONF API • ESC ポータル 詳細については、Elastic Services Controller ポータル （16 ページ）を参照してください。	VNF操作は、次のいずれかの方法で行います。 • REST API • NETCONF API • ETSI API

導入パラメータ	OpenStack	VMware vCenter	VMware vCloud Director
マルチクラスタ	該当なし	マルチクラスタ設定は、次のいずれかの方法で行います。 • REST API • ESC ポータル 詳細については、「ESC ポータルを使用した VMware vCenter での VNF の展開」を参照してください。	該当なし
複数の仮想データセンター（マルチ VDC）	該当なし	複数の仮想データセンターの選択は、次のいずれかの方法で行います。 • REST API • ESC ポータル	該当なし
ハードウェア アクセラレーション	ハードウェア アクセラレーションは、次のいずれかの方法でサポートされます。 • NETCONF API • REST API 詳細については、Cisco Elastic Services Controller アドミニストレーション ガイド [英語] の「Hardware Acceleration Support (OpenStack Only)」を参照してください。	N/A	N/A
単一のルート I/O 仮想化	シングルルート I/O 仮想化の設定は、次のいずれかの方法で行います。 • NETCONF API • REST API	シングルルート I/O 仮想化の設定は、次のいずれかの方法で行います。 • NETCONF API • REST API	該当なし

この章では、展開のカスタマイズの設定手順について説明します。VNF の展開の詳細については、OpenStack での仮想ネットワーク機能の展開 （115 ページ）を参照してください。



第 22 章

デイゼロ設定

デイゼロ設定

VNF の初期設定またはデイゼロ設定は、VM タイプに基づいています。VNF 管理者は、VNF の展開時に各 VM タイプの初期テンプレートを設定します。同じ設定テンプレートが、その VM タイプのすべての展開済み VM と新しい VM に適用されます。テンプレートは、個々の VM の展開時に処理されます。デイゼロ設定は引き続き維持されるため、VM のすべての初期展開、修復、およびスケーリングには同じデイゼロテンプレートが使用されます。

デイゼロ設定タスクには、インターフェイスの起動、ネットワークの管理、静的または動的IP（DHCP、IPAM）のサポート、SSH キー、VNF での NetConf 対応設定のサポートなどがあります。

（注） ESC は、サービスの更新中に追加されたインターフェイスのデイゼロ設定をサポートしません。デイゼロ設定のリカバリの場合、ネットワーク インターフェイス カード ID を持つすべてのインターフェイスが設定されます。

データモデルの設定のデイゼロ

データモデルでは、デイゼロ設定ファイルをさまざまな方法で指定できますが、一度に使用できるオプションは 1 つだけです。

- <file> URL </file> : URL は、ESC VM ファイルシステム上のファイル、またはレポート HTTP サーバでホストされるファイルを指定します。ESC は、URL で指定されたファイルをダウンロードします。このファイルは、このテンプレートで指定されたトークンを変数セクションで指定された値に置き換えるためのテンプレートとして使用されます。このテンプレートは、デイゼロ設定を生成するために使用されます。

- <data> インライン設定コンテンツ </data>：テンプレートの URL を指定します。これにより、インラインテキストをテンプレートとして使用できます。
- <encrypted_data>インライン設定コンテンツ </encrypted_data>：インライン設定の内容は、データに基づいて暗号化されます。
- <file_locators> ファイルロケータのリスト </file_locators> ：ファイルと同様に、file_locator は基本認証を使用してリモートサーバからダウンロードするファイルを定義します（必要な場合）。

（注） <file_locators> は ESC リリース 4.0 で廃止されました。

- <file_locator_name> 展開で定義された file_locator </file_locator_name> ：ファイルと同様に、file_locator_name は基本認証を使用してリモートサーバからファイルをダウンロードするために使用されます（必要な場合）。

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。各ユーザデータと設定ドライブファイルは、設定タグで定義されます。内容はテンプレートの形式です。ESC は、テンプレートを Apache Velocity Template Engine を介して処理した後で、VM に渡す前します。

config_data タグは、vm_group ごとに定義されます。同じ設定テンプレートが vm_group 内のすべての VM に適用されます。テンプレートファイルは、展開の初期化時に取得され、保存されます。テンプレートの処理は、VM の展開時に適用されます。設定ファイルの内容は、ファイルまたはデータから取得できます。

```
<file> url </file>
<data> inline config content </data>
```

宛先名は、<dst> によって設定に割り当てられます。ユーザデータは、<dst>--user-data</dst> を使用して特殊なケースとして扱われます。

サンプル設定データモデル

```
<config_data>
    <configuration>
        <file>file://cisco/userdata_file.txt</file>
        <dst>--user-data</dst>
        <variable>
            <name>CUSTOM_VARIABLE_FOR_USERDATA</name>
            <val>SOME_VALUE_XXX</val>
        </variable>
    </configuration>
    <configuration>
        <file>file://cisco/config.sh</file>
        <dst>config.sh</dst>
        <variable>
            <name>CUSTOM_VARIABLE_FOR_CONFIG</name>
            <val>SOME_VALUE_XXX</val>
        </variable>
    </configuration>
</config_data>
```

カスタム変数は、設定内の変数タグで指定できます。各設定には、ゼロ個以上の変数を含めることができます。各変数は複数の値を持つことができます。複数の値は、vm_group ごとに複

数のVMを作成する場合にのみ役立ちます。また、スケールインとスケールアウトを実行する場合、VM グループに VM を追加したり削除したりできます。

（注） 変数タグに複数の値を指定する場合は、次の点に注意してください。

- 最初に展開されたVMに割り当てられた変数値は一意であり、プールから取得されます。プールから値を割り当てる際に従うべき順序はありません。つまり、最初のVMはプールの 2 番目の値を使用できます。
- スケールアウトされた VM には、プールからの一意の変数値が必要です。
- （展開解除または再展開後に）復元されたVMは、以前と同じ値を保持する必要があります。

<file> の内容は、Velocity Template Engine によって処理されるテンプレートです。ESC は、設定テンプレートを処理する前に、インターフェイスごとに一連の変数を入力します。

NICID_n_a_IP_ALLOCATION_TYPE	FIXED | DHCP を含む文字列
NICID_n_a_NETWORK_ID	neutron network uuid を含む文字列
NICID_n_a_IP_ADDRESS	IPv4 または IPv6 アドレス
NICID_n_a_MAC_ADDRESS	文字列
NICID_n_a_GATEWAY	IPv4 または IPv6 ゲートウェイアドレス
NICID_n_a_CIDR_ADDRESS	IPv4 または IPv6 CIDR プレフィックスアドレス
NICID_n_a_CIDR_PREFIX	プレフィックス長を持つ整数
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。
NICID_n_a_ANYCAST_ADDRESS	IPv4 または IPv6 を含む文字列
NICID_n_a_IPV4_OCTETS	CloudVPN に固有の、IP アドレスの最後の 2 オクテット（16.66 など）を含む文字列

n は、データモデルのインターフェイス番号（0、1、2、3 など）です。

（注） インターフェイス番号 n は、OpenStack の場合は 0、VMware の場合は 1 から始まります。

例

```
NICID_0_NETWORK_ID=0affdc19-60fd-4a4f-a02b-f062d7a66c27
NICID_0_MAC_ADDRESS=fa:16:3e:4d:c5:f8

NICID_0_0_IP_ALLOCATION_TYPE=DHCP
NICID_0_0_IP_ADDRESS=1.1.22.133
NICID_0_0_GATEWAY=1.1.0.1
NICID_0_0_CIDR_ADDRESS=1.1.0.0
NICID_0_0_CIDR_PREFIX=16
NICID_0_0_NETMASK=255.255.0.0

NICID_0_1_IP_ALLOCATION_TYPE=DHCP
NICID_0_1_IP_ADDRESS=fd04:1::a03
NICID_0_1_GATEWAY=fd04:1::1
NICID_0_1_CIDR_ADDRESS=fd04:1::/64
NICID_0_1_CIDR_PREFIX=64
```

デフォルトでは、ESCは展開中にデイゼロ設定ファイルの$変数を実際の値に置き換えます。設定ファイルごとに$変数の置換を有効または無効にできます。

設定データモデルに次のフィールドを追加します。

`<template_engine>VELOCITY | NONE</template_engine>` の設定フィールド

値は次のとおりです。

- VELOCITY は変数の置換を可能にします。
- NONE は変数の置換を無効にします。

値が設定されていない場合、デフォルトのオプションはVELOCITYで、$変数の置換が行われます。NONE に設定すると、$ 変数の置換は行われません。

速度テンプレートエンジンを使用してテンプレートを処理する際は、次のヒントに従う必要があります。

- テンプレートでドル記号をエスケープするには、以下を挿入します。

  ```
  #set ( $DS = "$" )
  ```

  その後、変数を以下の値で置換します。

  ```
  passwd: ${DS}1${DS}h1VxC40U${DS}uf2qLUwGTjHgZplkP78xA
  ```

- テンプレート内のブロックをエスケープするには、#[[および#]] を挿入します。次に例を示します。

  ```
  #[[ passwd: $1$h1VxC40U$uf2qLUwGTjHgZplkP78xA ]]#
  ```

ファイルロケータ

外部設定ファイルを取得するため、デイゼロ設定にファイルロケータが追加されます。ファイルロケータには、ファイルサーバへの参照と、ダウンロードするファイルへの相対パスが含まれています。

（注） ファイルロケータ属性は展開レベルで定義されます。つまり、ポリシーアクションやデイゼロ設定セクションではなく、展開コンテナの直下で定義されます。更新されたデータモデルについては、「リモートサーバからのファイルの取得」を参照してください。

ファイルロケータを使用したデイゼロ設定の例：

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>sample-tenant</name>
      <deployments>
        <deployment>
          <name>sample-deployment</name>
          <vm_group>
            <name>sample-vm-group</name>
            <config_data>
              <!-- exisiting configuration example - remains valid -->
              <configuration>
                <file>file:///cisco/config.sh</file>
                <dst>config.sh</dst>
              </configuration>
              <!-- new configuration including use of file locators -->
              <configuration>
                <dst>ASA_config_0</dst>
                <file_locators>
                  <file_locator>
                    <name>configlocator-1</name>
                    <!-- unique name -->
                    <remote_file>
                      <file_server_id>server-1</file_server_id>
                      <remote_path>/share/users/configureScript.sh</remote_path>
                      <!-- optional user specified local silo directory -->
                      <local_target>day0/configureScript.sh</local_target>
                      <!-- persistence is an optional parameter -->
                      <persistence>FETCH_ALWAYS</persistence>
                      <!-- properties in the file_locator are only used for
                           fetching the file not for running scripts -->
                      <properties>
                        <property>
                          <!-- the property name "configuration_file" with value "true"
 indictates this is the
                               script to be used just as using the <file> member case
of the configuration -->
                          <name>configuration_file</name>
                          <value>true</value>
                        </property>
                        <property>
                          <name>server_timeout</name>
                          <value>120</value>
                          <!-- timeout value in seconds, overrides the file_server
property -->
                        </property>
                      </properties>
                    </remote_file>
                    <!-- checksum is an optional parameter.
                       The following algorithms are supported: SHA-1, SHA-224, SHA-256,
 SHA-384, SHA-512 -->
                    <checksum>SHA256 (configureScript.sh) =
```

```
dd526bb2c0711238ec2649c4b91598fb9a6cf1d2cb8559c337c5f3dd5ea1769e</checksum>
                  </file_locator>
                  <file_locator>
                    <name>configlocator-2</name>
                    <remote_file>
                      <file_server_id>server-2</file_server_id>
                      <remote_path>/secure/requiredData.txt</remote_path>
                      <local_target>day0/requiredData.txt</local_target>
                      <persistence>FETCH_ALWAYS</persistence>
                      <properties />
                    </remote_file>
                  </file_locator>
                </file_locators>
              </configuration>
            </config_data>
          </vm_group>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```

ファイルロケータのパラメータは次のとおりです。

- name：ファイルロケータのキーおよび識別子として使用されます。
- local_file または remote_file：ファイルの場所を選択します。ローカルファイルは、ESC VM ファイルシステムにすでに存在するファイルを指定するために使用されます。remote_file は、リモートサーバから取得するファイルを指定するために使用されます。
  - file_server_id：ファイルを取得するファイルサーバオブジェクトの ID。
  - remote_path：ファイルサーバオブジェクトで定義された base_url からのファイルのパス。
  - local_target：ファイルを保存するためのオプションのローカル相対ディレクトリ。
  - properties：必要な情報の名前と値のペア。
  - persistence：ファイルストレージのオプション。値には、CACHE、FETCH_ALWAYS、および FETCH_MISSING（デフォルト）が含まれます。
- checksum：転送されるファイルの有効性を検証するために使用する、オプションの BSD スタイルのチェックサム値。

詳細については、「リモートサーバからのファイルの取得」を参照してください。

ファイルを暗号化するには、「データ暗号化の設定」を参照してください。

vCD 展開のデイゼロ設定

vCD 展開のデイゼロ設定は、次のようなさまざまな方法で渡すことができます。

- ISO ファイルの作成
- OVF プロパティ

- カタログ内の既存の ISO ファイル（OOB ISO ファイル）

（注）
- 初回展開の場合、データモデルで定義されている VM グループの数は、vApp テンプレートの VM の数と同じである必要があります。展開では、各 VM グループのイメージ値は一意である必要があります。
- アウトオブバンド（OOB）ISO ファイルは、ISO ファイルメソッドの構築時に一緒に使用できません。これは、 VM がいずれも考慮できるためです。ovf プロパティは、OOB ISO とともに使用することも、ISO を一緒に構築することもできます。

ISO ファイルの構築によるデイゼロ設定：

```
</rules>
            <config_data>
              <!-- take content from the file path and save it as config.sh into the ISO
 file -->
               <configuration>
                 <dst>config.sh</dst>
                 <file>file:///cisco/config.sh</file>
               </configuration>
               <!-- take content from the file path, replace variables with values, and
save it as data/config.sh into the ISO file -->
               <configuration>
                 <dst>data/params.cfg</dst>
                 <file>file:///cisco/template.cfg</file>
                 <variable>
                   <name>CF_VIP_ADDR</name>
                   <val>10.0.0.9</val>
                 </variable>
                 <variable>
                   <name>CF_DOMAIN_NAME</name>
                   <val>cisco.com</val>
                 </variable>
                 <variable>
                   <name>CF_NAME_SERVER</name>
                   <val>172.16.180.7</val>
                 </variable>
               </configuration>
               <!-- take the data section as the content of the file, replace variables
with values, and save it as user-data.txt into the ISO file-->
               <configuration>
                 <dst>user-data.txt</dst>
                 <data>#cloud-config
manage_etc_hosts: true
hostname: $HOST_NAME
local-hostname: $HOST_NAME
</data>
                 <variable>
                   <name>$HOST_NAME</name>
                   <val>something.cisco.com</val>
                 </variable>
               </configuration>
             </config_data>
```

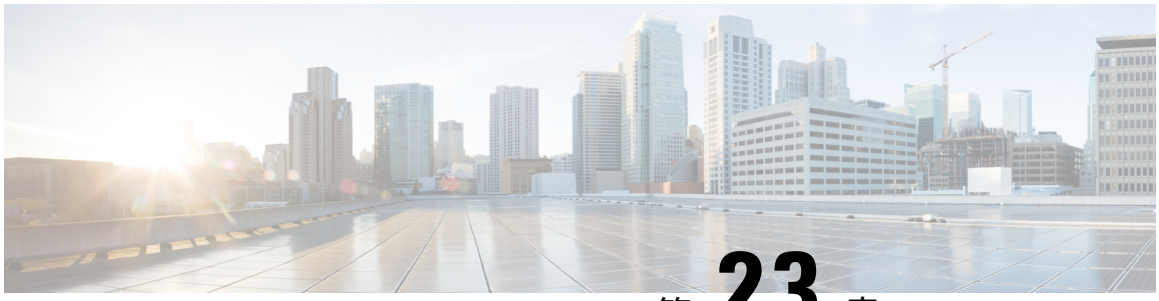
OOB ISO ファイルによるデイゼロ設定：

```
</rules>
   <config_data>
     <configuration>
       <!-- ISO file stored in catalog-1 -->
       <dst>vcdCatalog:catalog-1</dst>
       <data>h2.iso</data>
     </configuration>
   </config_data>
```

OVF プロパティによるデイゼロ設定：

```
  <config_data>
     <configuration>
       <!-- ovf properties as day0 -->
       <dst>ovfProperty:mgmt-ipv4-addr</dst>
       <data>$NICID_0_IP_ADDRESS/24</data>
     </configuration>
   </config_data>
```

vCD での VNF の導入については、VMware vCloud Director（vCD）での仮想ネットワーク機能の展開 （152 ページ）を参照してください。



第 23 章

KPI、ルール、およびメトリック

KPI、ルール、およびメトリック

Cisco Elastic Services Controller VNF モニタリングは、重要業績評価指標（KPI）メトリックの定義に基づいて行われます。コアメトリックには ESC がプリロードされています。プログラム可能なインターフェイスにより、エンドユーザはメトリックを追加および削除できるだけでなく、指定された条件でトリガーされるアクションを定義できます。これらのメトリックとアクションは、展開時に定義されます。

ESC のメトリックとアクションのデータモデルは、次の 2 つのセクションに分かれています。

1. KPI：モニタリングのタイプ、イベント、ポーリング間隔、およびその他のパラメータを定義します。対象には、event_name、しきい値、およびメトリック値が含まれます。event_name はユーザが定義した名前です。metric_values ではしきい値の条件およびその他の詳細を指定します。しきい値の条件に達したときに、イベントがトリガーされます。
2. ルール：KPI モニタリングイベントがトリガーされたときのアクションを定義します。action 要素では、event_name に対応するイベントがトリガーされたときに実行されるアクションを定義します。

ルール

ESC オブジェクトモデルは、vm_group ごとにセクションを定義します。このセクションでは、エンドユーザが、選択した KPI メトリックコレクタの結果に基づいて適用する管理ルールを指定できます。

```
<rules>
    <admin_rules>
        <rule>
            <event_name>VM_ALIVE</event_name>
            <action>TRUE esc_vm_alive_notification</action>
            <action>FALSE recover autohealing</action>
        </rule>
        : : : : : : : : : : : : : : : : : :
```

```
    </admin_rules>
</rules>
```

「KPI」セクションで説明したように、KPI とルールの相関は、<event_name> タグの値に基づいて実行されます。

上記の [ルール（Rules）] セクションで、event_name を定義する KPI の結果が VM_ALIVE で、選択されたメトリックコレクタが TRUE の場合、キー TRUE esc_vm_alive_notification によって識別されるアクションが実行用に選択されます。

event_name を定義する KPI の結果が VM_ALIVE であり、選択したメトリックコレクタが FALSE の場合、キー FALSE recover autohealing によって識別されるアクションが実行用に選択されます。

KPI とルールの更新については、KPI とルールの更新 （245 ページ）を参照してください。

メトリックおよびアクション

ESC メトリックおよびアクション（ダイナミックマッピング）フレームワークは、KPI およびルールセクションの基盤です。「KPI」セクションで説明したように、メトリックタイプはメトリックとそのメタデータを一意に識別します。

メトリックとアクションは次のとおりです。

```
<metrics>
    <metric>
        <name>ICMPPING</name>
        <userLabel>ICMP Ping</userLabel>
        <type>MONITOR_SUCCESS_FAILURE</type>
        <metaData>
            <type>icmp_ping</type>
            <properties>
                <property>
                    <name>ip_address</name>
                    <value />
                </property>
                <property>
                    <name>enable_events_after_success</name>
                    <value>true</value>
                </property>
                <property>
                    <name>vm_gateway_ip_address</name>
                    <value />
                </property>
                <property>
                    <name>enable_check_interface</name>
                    <value>true</value>
                </property>
            </properties>
        </metaData>
    </metric>
    : : : : : : : : :
</metrics>
```

上記のメトリックは、一意の名前 ICMPPING によって識別されます。<type> タグは、メトリックタイプを識別します。

現在、ESC は次の 2 種類のメトリックをサポートしています。

- MONITOR_SUCCESS_FAILURE
- MONITOR_THRESHOLD

<metadata>セクションは、モニタリングエンジンによって処理される属性とプロパティを定義します。

KPI の metric_collector タイプは、次の動作を示します。

ICMPPING 識別子に関連付けられた動作が 3 秒間隔でトリガーされます。ICMPPING メトリックのタイプは MONITOR_SUCCESS_FAILURE です。つまり、モニタリングアクションの結果は成功または失敗となります。上記のサンプルでは、icmp_ping は <metadata> セクションで定義されている <ip_address> フィールドを使用して実施されます。SUCCESS の場合、プレフィックスが TRUE のルールアクションが選択されて実行されます。FAILURE の場合、プレフィックスが FALSE のルールアクションが選択されて実行されます。

```
<actions>
    <action>
        <name>TRUE servicebooted.sh esc_vm_alive_notification</name>
        <type>ESC_POST_EVENT</type>
        <metaData>
            <type>esc_post_event</type>
            <properties>
                <property>
                    <name>esc_url</name>
                    <value />
                </property>
                <property>
                    <name>vm_external_id</name>
                    <value />
                </property>
                <property>
                    <name>vm_name</name>
                    <value />
                </property>
                <property>
                    <name>event_name</name>
                    <value />
                </property>
                <property>
                    <name>esc_event</name>
                    <value>SERVICE_BOOTED</value>
                </property>
            </properties>
        </metaData>
    </action>
    : : : : : : : : :
</actions>
```

上記のアクションサンプルは、SUCCESS 値に関連付けられた動作について説明しています。ESC ルールアクション名 TRUE servicebooted.sh esc_vm_alive_notification は、選択するアクションを指定します。アクションを選択すると、<type> ESC_POST_EVENT は、モニタリングエンジンが選択するアクションを識別します。

メトリックおよびアクション API

Cisco ESC リリース 2.1 以前では、データモデルで定義されたアクションおよびメトリックから、モニタリングエージェントで使用可能な有効なアクションおよびメトリックへのマッピン

グは、dynamic_mappings.xml ファイルを使用して有効化されていました。ファイルは ESC VM に保存され、テキストエディタを使用して変更されました。ESC 2.2 以降には、*esc-dynamic-mapping* ディレクトリと *dynamic_mappings.xml* ファイルがありません。したがって、ESC VM に追加する既存の dynamic_mapping.xml ファイルがある場合は、次の手順を実行します。

1. このファイルを、ホームディレクトリなどの ESC 以外の場所にバックアップします。
2. ESC VM で *esc-dynamic-mapping* ディレクトリを作成します。読み取りアクセス許可が設定されていることを確認します。
3. 次の bootvm 引数を使用して、ESC VM にインストールします。

```
 --file
root:root:/opt/cisco/esc/esc-dynamic-mapping/dynamic_mappings.xml:<path-to-local-copy-of-dynamic-mapping.xml>
```

アクションとメトリックをマッピングするための CRUD 操作は、REST API を介して実行できます。マッピングされたメトリックとアクションの定義については、以下の API の表を参照してください。

既存のマッピングを更新するには、REST API を使用してそのマッピングを削除して、新しいマッピングを追加します。

（注） 以前のバージョンの ESC を ESC 2.2 以降にアップグレードする場合、VNF モニタリングルールを維持するには、dynamic_mappings.xml ファイルをバックアップしてから、アップグレードした ESC VM でファイルを復元する必要があります。モニタリングルールのアップグレードの詳細については、Cisco Elastic Services Controller インストールおよびアップグレードガイド [英語] の「Upgrade VNF Monitoring Rules」を参照してください。Cisco ESC リリース 2.3.2 以降では、ダイナミックマッピング API は ESC VM でのみローカルにアクセスできます。

表 7: マッピングされたアクション

ユーザ操作	パス	HTTP 動作	ペイロード	応答	説明
読み取り	internal/dynamic_mapping/actions/<action_name>	GET	該当なし	アクション XML	名前でアクションを取得する
すべて読み取り	internal/dynamic_mapping/actions	GET	該当なし	アクション XML	定義済みのすべてのアクションを取得する
作成	internal/dynamic_mapping/actions	POST	アクション XML	予期されるアクション XML	1 つまたは複数のアクションを作成する

ユーザ操作	パス	HTTP 動作	ペイロード	応答	説明
削除	internal/dynamic_mapping/actions/<action_name>	DELETE	該当なし	該当なし	名前でアクションを削除する
すべてクリア	internal/dynamic_mapping/actions	DELETE	該当なし	該当なし	すべての非コアアクションを削除する

アクション API の応答は次のとおりです。

```
<actions>
    <action>
        <name>{action name}</name>
        <type>{action type}</type>
        <metaData>
            <type>{monitoring engine action type}</type>
            <properties>
                <property>
                    <name />
                    <value />
                </property>
                : : : : : : :
            </properties>
        </metaData>
    </action>
    : : : : : : : : :
</actions>
```

それぞれの説明は次のとおりです。

{action name}：アクションの一意の識別子。ESC オブジェクトモデルに準拠するために、成功または失敗のアクションの場合、名前は TRUE または FALSE で始める必要があります。

{action type}：現在のリリースのアクションタイプは ESC _POST_EVENT、SCRIPT、または CUSTOM_SCRIPT です。

{monitoring engine action type}：モニタリングエンジンタイプは、icmp_ping、icmp4_ping、icmp6_ping、esc_post_event、script、custom_script、snmp_get です。詳細については、「VNF のモニタリング」を参照してください。

コアおよびデフォルトアクションリスト

表 *8:* コアおよびデフォルトアクションリスト

名前	タイプ	説明
TRUE esc_vm_alive_notification	コア	サービスの開始
TRUE servicebooted.sh	コア/レガシー	サービスの開始
FALSE recover autohealing	コア	サービスの回復
TRUE servicescaleup.sh	コア/レガシー	スケール アウト

名前	タイプ	説明
TRUE esc_vm_scale_out_notification	コア	スケール アウト
TRUE servicescaledown.sh	コア/レガシー	スケールイン
TRUE esc_vm_scale_in_notification	コア	スケールイン
TRUE apply_netscaler_license.py	デフォルト	NetScaler ライセンスの適用

コアアクションとメトリックはESCによって定義され、削除したり、更新したりできません。

デフォルトのアクションまたはメトリックは ESC によって定義され、より複雑なモニタリング機能のコアアクションまたはメトリックを補完するために存在します。これらは、ユーザが削除および変更できます。デフォルトのアクションまたはメトリックは、同じ名前のアクションまたはメトリックがデータベースで見つからないたびに、ESC の起動時にリロードされます。

メトリック API

表 *9:* マッピングされたメトリック

ユーザ操作	パス	HTTP 動作	ペイロード	応答	説明
読み取り	internal/dynamic_mapping/actions/<metric_name>	GET	該当なし	メトリック XML	名前でメトリックを取得する
すべて読み取り	internal/dynamic_mapping/metrics/	GET	該当なし	メトリック XML	定義されているすべてのメトリックを取得する
作成	internal/dynamic_mapping/metrics/	POST	メトリック XML	予想されるメトリック XML	1 つまたは複数のメトリックを作成する
削除	internal/dynamic_mapping/actions/<metric_name>	DELETE	該当なし	該当なし	名前でメトリックを削除する
すべてクリア	internal/dynamic_mapping/metrics	DELETE	該当なし	該当なし	すべての非コアメトリックを削除する

メトリック API の応答は次のとおりです。

```
<metrics>
    <metric>
        <name>{metric name}</name>
        <type>{metric type}</type>
```

```
        <metaData>
            <type>{monitoring engine action type}</type>
            <properties>
                <property>
                    <name />
                    <value />
                </property>
                : : : : : : : :
            </properties>
        </metaData>
    </metric>
    : : : : : : : : :
</metrics>
```

それぞれの説明は次のとおりです。

{metric name}：メトリックの一意の識別子。

{metric type}：メトリックタイプはMONITOR_SUCCESS_FAILURE、MONITOR_THRESHOLD、または MONITOR_THRESHOLD_COMPUTE です。

{monitoring engine action type}：モニタリングエンジンタイプは、icmp_ping、icmp4_ping、icmp6_ping、esc_post_event、script、custom_script、snmp_get です。詳細については、「モニタリング」を参照してください。

コアおよびデフォルトアクションリスト

表 *10*：コアおよびデフォルトアクションリスト

名前	タイプ	説明
ICMPPING	コア	ICMP Ping
MEMORY	デフォルト	メモリのコンピューティング使用率
CPU	デフォルト	CPU のコンピューティング使用率
CPU_LOAD_1	デフォルト	CPU の 1 分間の平均負荷
CPU_LOAD_5	デフォルト	CPU の 5 分間の平均負荷
CPU_LOAD_15	デフォルト	CPU の15 分間の平均負荷
PROCESSING_LOAD	デフォルト	CSR の処理負荷
OUTPUT_TOTAL_BIT_RATE	デフォルト	CSR の合計ビットレート
SUBSCRIBER_SESSION	デフォルト	CSR 加入者セッション

ESC サービスの展開

KPI セクションでは、モニタリングメトリックを使用して新しい KPI を定義します。

```
<kpi>
    <event_name>DEMO_SCRIPT_SCALE_OUT</event_name>
    <metric_value>20</metric_value>
    <metric_cond>GT</metric_cond>
    <metric_type>UINT32</metric_type>
    <metric_collector>
        <type>custom_script_count_sessions</type>
        <nicid>0</nicid>
        <poll_frequency>15</poll_frequency>
        <polling_unit>seconds</polling_unit>
        <continuous_alarm>false</continuous_alarm>
    </metric_collector>
</kpi>
<kpi>
    <event_name>DEMO_SCRIPT_SCALE_IN</event_name>
    <metric_value>1</metric_value>
    <metric_cond>LT</metric_cond>
    <metric_type>UINT32</metric_type>
    <metric_occurrences_true>1</metric_occurrences_true>
    <metric_occurrences_false>1</metric_occurrences_false>
    <metric_collector>
        <type>custom_script_count_sessions</type>
        <nicid>0</nicid>
        <poll_frequency>15</poll_frequency>
        <polling_unit>seconds</polling_unit>
        <continuous_alarm>false</continuous_alarm>
    </metric_collector>
</kpi>
```

前述のサンプルでは、最初の KPI セクションで、custom_script_count_sessions で識別されるメトリックが 15 秒間隔で実行されます。メトリックによって返される値が 20 より大きい場合、イベント名 DEMO_SCRIPT_SCALE_OUT がトリガーされ、rule セクションで処理されます。

前述のサンプルでは、2 番目の KPI セクションで、custom_script_count_sessions で識別されるメトリックが 15 秒間隔で実行されます。メトリックによって返される値が 1 未満の場合、イベント名 DEMO_SCRIPT_SCALE_IN がトリガーされ、rule セクションで処理されます。

rule セクションでは、KPI で使用されている event_name を使用してルールを定義します。action タグでは、event_name がトリガーされたときに実行されるアクションを定義します。次の例では、イベント DEMO_SCRIPT_SCALE_OUT がトリガーされると、TRUE ScaleOut 識別子によって識別されるアクションが実行されます。

```
<rule>
    <event_name>DEMO_SCRIPT_SCALE_OUT</event_name>
    <action>ALWAYS log</action>
    <action>TRUE ScaleOut</action>
</rule>
<rule>
    <event_name>DEMO_SCRIPT_SCALE_IN</event_name>
    <action>ALWAYS log</action>
    <action>TRUE ScaleIn</action>
</rule>
```

スクリプトアクション

次の 2 種類のアクションがサポートされています。

1. 事前定義されたアクション

2. スクリプトアクション

ポリシー主導型データモデルの一部としてスクリプトの実行を指定できます。*script_filename* プロパティは、ESC VM 上のスクリプトへの絶対パスを指定するスクリプトアクションに必須です。次の XML スニペットは、スクリプトアクションの動作例を示しています。

```
<action>
    <name>GEN_VPC_CHASSIS_ID</name>
    <type>SCRIPT</type>
    <properties>
        <property>
            <name>script_filename</name>
            <value>/opt/cisco/esc/esc-scripts/esc_vpc_chassis_id.py</value>
        </property>
        <property>
            <name>CHASSIS_KEY</name>
            <value>164c03a0-eebb-44a8-87fa-20c791c0aa6d</value>
        </property>
    </properties>
</action>
```

スクリプトのタイムアウトは、デフォルトでは 15 分です。ただし、プロパティセクションに *wait_max_timeout* プロパティを追加することで、スクリプトごとに異なるタイムアウト値を指定できます。次に、このスクリプトにのみタイムアウトを 5 分に設定する例を示します。

```
<action>
    <name>GEN_VPC_CHASSIS_ID</name>
    <type>SCRIPT</type>
    <properties>
        <property>
            <name>script_filename</name>
            <value>/opt/cisco/esc/esc-scripts/esc_vpc_chassis_id.py</value>
        </property>
        <property>
            <name>CHASSIS_KEY</name>
            <value>164c03a0-eebb-44a8-87fa-20c791c0aa6d</value>
        </property>
        <property>
            <name>wait_max_timeout</name>
            <value>300</value>
        </property>
    </properties>
</action>
```

上記の例では、GEN_VPC_CHASSIS_ID のタイムアウト値は 300 秒、つまり 5 分です。ESC には、実行中のすべてのスクリプトに対してデフォルトのタイムアウト時間を指定するグローバルパラメータもあり、MONA カテゴリの SCRIPT_TIMEOUT_SEC と呼ばれます。スクリプトで wait_max_timeout プロパティが定義されていない限り、これがデフォルト値として機能します。

事前定義されたアクションのトリガー

ESC では、必要に応じて、Dynamic Mapping API で定義された既存の（事前定義済みの）アクションをトリガーする新しい REST API が導入されています。メトリックおよびアクション API の詳細については、メトリックおよびアクション API（183 ページ）を参照してください。

定義済みアクションの例は次のとおりです。

```
<actions>
    <action>
        <name>SaidDoIt</name>
        <userlabel>My Friendly Action</userlabel>
        <type>SCRIPT</type>
        <metaData>
            <type>script</type>
            <properties>
                <property>
                    <name>script_filename</name>
                    <value>/opt/cisco/esc/esc-scripts/do_somethin.py</value>
                </property>
                <property>
                    <name>arg1</name>
                    <value>some_val</value>
                </property>
                <property>
                    <name>notification</name>
                    <value>true</value>
                </property>
            </properties>
        </metaData>
    </action>
</actions>
```

（注） リモートサーバにあるスクリプトファイルもサポートされます。<value> タグに詳細を入力する必要があります。例：
http://myremoteserverIP:80/file_store/do_somethin.py</value>http://myremoteserverIP:80/file_store/do_somethin.py</value>

前述の事前定義済みアクションは、トリガー API を使用してトリガーされます。

次の HTTP または HTTPS POST 操作を実行します。

POST `http://<IP_ADDRESS>:8080/ESCManager/v0/trigger/action/`

POST `https://<IP_ADDRESS>:8443/ESCManager/v0/trigger/action/`

次のペイロードは、API によってトリガーされたアクションおよび受信した応答を示します。

```
<triggerTarget>
  <action>SaidDoIt</action>
  <properties>
   <property>
     <name>arg1</name>
     <value>real_value</value>
   </property>
  </properties>
</triggerTarget>
```

応答、

```
<triggerResponse>
    <handle>c11be5b6-f0cc-47ff-97b4-a73cce3363a5</handle>
    <message>Action : 'SAIDDOIT' triggered</message>
</triggerResponse>
```

ESC は要求を受け入れ、応答ペイロードとステータスコードを返します。

HTTP ステータスコード 200 は、トリガーされたアクションが存在し、正常にトリガーされたことを示します。HTTP ステータスコード 400 または 404 は、トリガーされるアクションが見つからないことを示します。

さまざまなライフサイクルステージで NB に送信されるカスタムスクリプト通知を使用して、ステータスを確認できます。

ESC は、MANUAL_TRIGGERED_ACTION_UPDATE コールバックイベントを、アクションの実行の成功または失敗を示すステータスメッセージとともに送信します。

通知は次のとおりです。

```
<esc_event xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <event_type>MANUAL_TRIGGERED_ACTION_UPDATE</event_type>
    <properties>
        <property>
            <name>handle</name>
            <value>c11be5b6-f0cc-47ff-97b4-a73cce3363a5</value>
        </property>
        <property>
            <name>message</name>
            <value>Action execution success</value>
        </property>
        <property>
            <name>exit_code</name>
            <value>0</value>
        </property>
        <property>
            <name>action_name</name>
            <value>SAIDDOIT</value>
        </property>
    </properties>
</esc_event>
```

（注） script_filename プロパティは、トリガー API 要求によって上書きできません。トリガー API には、事前定義済みアクションに存在しない追加のプロパティを含めることはできません。

新しい API では、次に示す（アクションの）特別なプロパティの一部をオーバーライドできます。

- [通知（Notification）]：スクリプトが実行時に進捗通知を生成する場合に設定します。デフォルト値は false です。この値は、アクションまたはトリガーペイロードで true に設定できます。
- wait_max_timeout：この時間が終了するまで、スクリプトの実行完了を待機します。デフォルトは、900 秒です。

（注）
- トリガー API は、スクリプトタイプのアクションのみをサポートします。
- ESC VM にあるスクリプトアクションが、アクティブ HA インスタンスとスタンバイ HA インスタンスの両方で同じパスにコピーされることを確認します。詳細については、『*Cisco Elastic Services Controller Install and Upgrade Guide*』の「高可用性」章を参照してください。
- ESC サービスのフェールオーバー、シャットダウン、または再起動が発生すると、スクリプトの実行は終了します。

カスタム スクリプト メトリック モニタリング KPI およびルールの設定

カスタム スクリプト メトリック モニタリングは、次の手順で実行できます。

1. スクリプトの作成
2. メトリックの追加
3. アクションの追加
4. 展開の定義
5. KPI データまたはルールの更新
6. KPI とルールを使用したリモートサーバの認証

実行するスクリプトは、MONITOR_THRESHOLD アクションに指定されたルールに準拠している必要があります。しきい値超過の評価は、スクリプト実行の終了値に基づいて行われます。次のサンプルスクリプトでは、戻り値は IP セッションの数です。

```
#!/usr/bin/env python
import pexpect
import re
import sys
ssh_newkey = 'Are you sure you want to continue connecting'
# Functions
def get_value(key):
    i = 0
    for arg in sys.argv:
        i = i + 1
        if arg == key:
            return sys.argv[i]
    return None
def get_ip_addr():
    device_ip = get_value("vm_ip_address")
    return device_ip
# Main
CSR_IP = get_ip_addr()

p=pexpect.spawn('ssh admin@' + CSR_IP + ' show ip nat translations total')
i=p.expect([ssh_newkey,'assword:',pexpect.EOF])
if i==0:
    p.sendline('yes')
```

```
    i=p.expect([ssh_newkey,'assword:',pexpect.EOF])
if i==1:
    p.sendline("admin")
    p.expect(pexpect.EOF)
elif i==2:
    pass
n = p.before
result = re.findall(r'\d+', n)[0]
sys.exit(int(result))
```

ESC モニタリングおよびアクションエンジンは、スクリプトの終了値を処理します。

スクリプトは、ESC VM ディレクトリ /opt/cisco/esc/esc-scripts/ にインストールする必要があります。

次のペイロードは、スクリプトで定義された custom_script を使用したメトリックを示しています。

```
<!-- Demo Metric Counting Sessions -->
<metrics>
    <metric>
        <name>custom_script_count_sessions</name>
        <type>MONITOR_THRESHOLD</type>
        <metaData>
            <properties>
                <property>
                    <name>script_filename</name>
                    <value>/cisco/esc-scripts/countSessions.py</value>
                </property>
                <property>
                    <name>for_threshold</name>
                    <value>true</value>
                </property>
            </properties>
            <type>custom_script_threshold</type>
        </metaData>
    </metric>
</metrics>
<!-- -->
```

メトリックペイロードは、マッピング API を使用してサポートされる ESC メトリックのリストに追加する必要があります。

次の URI で HTTP POST 操作を実行します。

http://<my_esc_ip>:8080/ESCManager/internal/dynamic_mapping/metrics

次のペイロードは、マッピング API を使用してサポートされる ESC アクションのリストに追加できるカスタムアクションを示しています。

```
<actions>
    <action>
        <name>TRUE ScaleOut</name>
        <type>ESC_POST_EVENT</type>
        <metaData>
            <type>esc_post_event</type>
            <properties>
                <property>
                    <name>esc_url</name>
                    <value />
```

```
                    </property>
                    <property>
                        <name>vm_external_id</name>
                        <value />
                    </property>
                    <property>
                        <name>vm_name</name>
                        <value />
                    </property>
                    <property>
                        <name>event_name</name>
                        <value />
                    </property>
                    <property>
                        <name>esc_event</name>
                        <value>VM_SCALE_Out</value>
                    </property>
                    <property>
                        <name>esc_config_data</name>
                        <value />
                    </property>
                    <properties />
                </properties>
            </metaData>
        </action>
        <action>
            <name>TRUE ScaleIn</name>
            <type>ESC_POST_EVENT</type>
            <metaData>
                <type>esc_post_event</type>
                <properties>
                    <property>
                        <name>esc_url</name>
                        <value />
                    </property>
                    <property>
                        <name>vm_external_id</name>
                        <value />
                    </property>
                    <property>
                        <name>vm_name</name>
                        <value />
                    </property>
                    <property>
                        <name>event_name</name>
                        <value />
                    </property>
                    <property>
                        <name>esc_event</name>
                        <value>VM_SCALE_IN</value>
                    </property>
                    <properties />
                </properties>
            </metaData>
        </action>
    </actions>
```

次の URI で HTTP POST 操作を実行します。

http://<IP_ADDRESS>:8080/ESCManager/internal/dynamic_mapping/actions

カスタムスクリプト通知

ESCは、特定のライフサイクルステージでの展開の一環として実行される、カスタマイズされたスクリプトに関するノースバウンドへの通知の送信をサポートするようになりました。この通知によって、実行されたスクリプトの進行状況を確認することもできます。通知を使用してカスタムスクリプトを実行するには、アクションタイプ属性を SCRIPT として定義し、プロパティ属性名を notification として定義し、値を true に設定します。

たとえば、次のデータモデルでは、展開が POST_DEPLOY_ALIVE ステージに達したときに、/var/tmp/esc-scripts/senotification.py にあるカスタマイズされたスクリプトを実行します。

```
<policies>
    <policy>
        <name>PCRF_POST_DEPLOYMENT</name>
        <conditions>
            <condition>
                <name>LCS::POST_DEPLOY_ALIVE</name>
            </condition>
        </conditions>
        <actions>
            <action>
                <name>ANY_NAME</name>
                <type>SCRIPT</type>
                <properties>
                    <property>
                        <name>script_filename</name>
                        <value>/var/tmp/esc-scripts/senotification.py</value>
                    </property>
                    <property>
                        <name>notification</name>
                        <value>true</value>
                    </property>
                </properties>
            </action>
        </actions>
    </policy>
</policies>
```

次の出力を使用して、スクリプトの進行状況をノースバウンドに通知できます。

- 標準 JSON 出力
- REST API コール
- NETCONF 通知

標準 JSON 出力

標準 JSON 出力は MONA 通知規則に従います。MONA は、このエントリをキャプチャして通知を生成します。

```
{"esc-notification":{"items":{"properties":
[{"name":"name1","value":"value1"},{"name":"name2","value":"value2"}...]}}}
```

表 *11 :* 項目品目リスト

名前	説明
タイプ	通知のタイプを示します。 `progress_steps | progress_percentage | log | alert | error`
progress （注） 進捗項目は、タイプが progress-steps または progress-percentage の場合にのみ必要です。	progress-steps タイプの場合： `{current_step}|{total_steps}`
	progress-percentage タイプの場合： `{percentage}`
msg	通知メッセージ。

JSON の出力例は次のとおりです。

```
{"esc-notification":{"items":{"properties": [{"name":"type",
"value":"progress_percentage"},{"name":"progress","value":"25"},{"name":"msg","value":"Installation
in progress."}]}}}
```

（注） カスタムスクリプトが Python で記述されている場合、標準出力はデフォルトでバッファされるため、各通知の print ステートメントの後に、スクリプトは sys.stdout.flush() を呼び出してバッファをフラッシュする必要があります（Python 3.0 より前）。そうでない場合、MONA はスクリプト stdout をリアルタイムで処理できません。`print '{"esc-notification":{"items":{"properties": [{"name":"type", "value":"progress_percentage"},{"name":"progress","value":"25"},{"name":"msg","value":"Installation in progress."}]}}}'sys.stdout.flush()`

REST API コール

```
http://localhost:8090/mona/v1/actions/notification
```

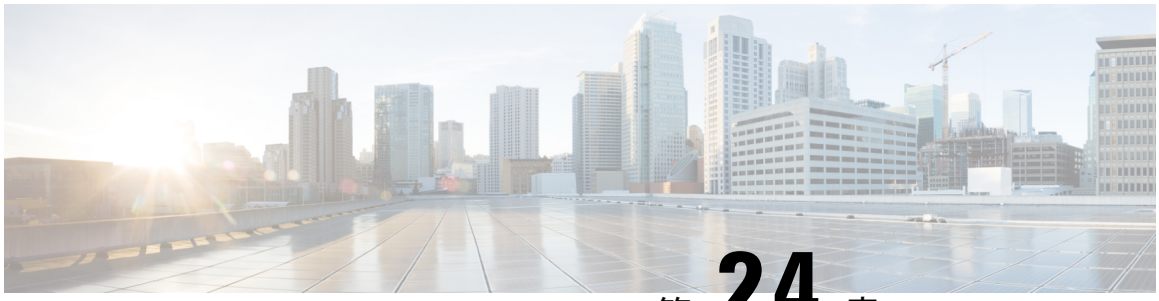
REST API では、スクリプトは最後のパラメータとしてスクリプトハンドルを受け入れる必要があります。スクリプトハンドルは、UUID、MONA アクション、または実行ジョブ ID です。たとえば、MONA 通知をサポートするためにスクリプトが元々 3 つのコマンドラインパラメータを受け入れる場合、スクリプトはハンドル UUID の追加パラメータを考慮します。これにより、MONA は通知ソースを識別できます。スクリプトは、通知ごとに、スクリプト内で MONA のエンドポイントへの POST REST コールを作成します。

ペイロードは次のとおりです。

```
{
 "esc-notification" : {
  "items" : {
   "properties" : [{
     "name" : "type",
     "value" : "log",
```

```
        "hidden" : false
       }, {
        "name" : "msg",
        "value" : "Log info",
        "hidden" : false
       }
      ]
     },
     "source" : {
      "action_handle" : "f82fe86d-6625-4b13-99f7-89d169e427ad"
     }
    }
}
```

（注） action_handle 値は、MONA がスクリプトに渡すハンドル UUID です。



第 24 章

ポリシー駆動型データモデル

ポリシー駆動型データモデル

ESC は、新しいポリシー駆動型データモデルをサポートします。新しい <policy> セクションは、展開および VM グループレベルの両方で、<policies> の下に導入されています。

ポリシーデータモデルを使用すると、ユーザは条件に基づいてアクションを実行できます。ESC は、特定のライフサイクルステージ（LCS）に基づいて、展開時に定義済みのアクションまたはカスタマイズされたスクリプトをサポートします。たとえば、再展開ポリシーは、ライフサイクルステージ（LCS）に基づいて事前定義されたアクションを使用して VM を再展開します。詳細については、再展開ポリシー （353 ページ）を参照してください。

ポリシーデータモデル

ポリシーデータモデルは、条件とアクションで構成されます。条件は、展開のライフサイクルステージ（LCS）です。アクションは、定義済みまたはカスタムスクリプトです。

- 事前定義アクション：アクションは事前定義され、条件が満たされたときに実行されます。

  次のデータモデルでは、condition2 が満たされると、Action2 が実行されます。アクション <type> は事前定義されています。

- カスタムスクリプト：アクションはカスタムスクリプトであり、条件が満たされたときに実行されます。

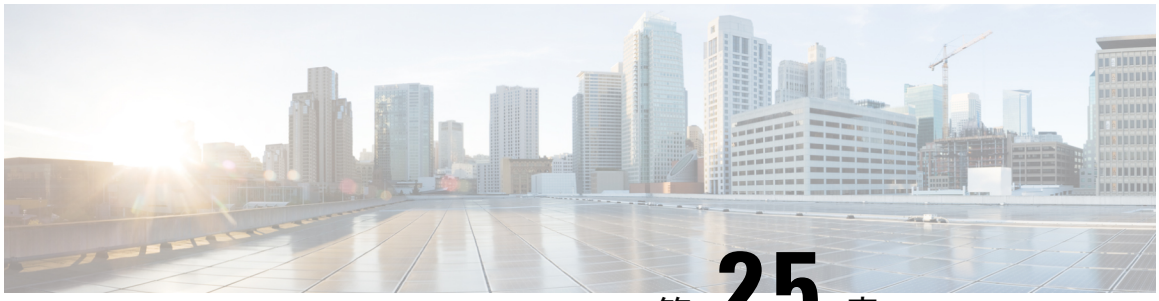
  次のデータモデルでは、condition1 が満たされると、Action1-1 と Action 1-2 が実行されます。アクション <type> はスクリプトです。

```
<policies>
    <policy>
        <name>Name1</name>
        <conditions>
            <condition>
                <name>Condition1</name>
            </condition>
        </conditions>
```

```
            <actions>
                <action>
                    <name>Action1-1</name>
                    <type>SCRIPT</type>
                </action>
                <action>
                    <name>Action1-2</name>
                    <type>SCRIPT</type>
                </action>
            </actions>
        </policy>
        <policy>
            <name>Name2</name>
            <conditions>
                <condition>
                    <name>Condition2</name>
                </condition>
            </conditions>
            <actions>
                <action>
                    <name>Action2</name>
                    <type>PRE-DEFINED</type>
                </action>
            </actions>
        </policy>
    </policies>
```

事前定義アクション、およびスクリプトの詳細については、リカバリポリシーと再展開ポリシー （350 ページ）を参照してください。

次の表に、展開内の LCS とその説明を示します。リカバリポリシーと再展開ポリシー、および VNF ソフトウェア アップグレード ポリシーは、ポリシー駆動型データモデルを使用します。これらのポリシーは、単一 VIM 展開と複数 VIM 展開の両方でサポートされます。詳細については、「仮想ネットワーク機能の展開」を参照してください。ポリシーフレームワークを使用したリカバリおよび再展開ポリシーの設定の詳細については、リカバリポリシーと再展開ポリシー （350 ページ）を参照してください。VNF ソフトウェア アップグレード ポリシーのアップグレードの詳細については、ボリュームを使用した VNF ソフトウェアのアップグレード （282 ページ）を参照してください。



第 25 章

サポート対象のライフサイクルステージ（LCS）

サポート対象のライフサイクルステージ（LCS）

表 *12 :* 条件とその範囲

条件名	範囲	説明
LCS::PRE_DEPLOY	展開	展開の VM を展開する直前に発生します。
LCS::POST_DEPLOY_ALIVE	展開	展開がアクティブになった直後に発生します。
LCS::DEPLOY_ERR	展開	展開が失敗した直後に発生します。
LCS::POST_DEPLOY:: VM_RECOVERY_ERR	展開	1 つの VM のリカバリが失敗した直後に発生します （これは展開レベルで指定され、すべての VM グループに適用されます）。
LCS::POST_DEPLOY:: VM_RECOVERY _REDEPLOY_ERR	展開	1 つの VM の再展開が失敗した直後に発生します （これは展開レベルで指定され、すべての VM グループに適用されます）。

LCS::DEPLOY_UPDATE::VM_ PRE_VOLUME_DETACH	展開	ESC がボリュームをデタッチする直前にトリガーされます （これは個々の VM のグループに対して指定され、展開全体ではなく <vm_group> の下で指定されます）。
LCS::DEPLOY_UPDATE:: VM_VOLUME_ATTACHED	展開	ESC が新しいボリュームをアタッチした直後にトリガーされます （これは個々の VM のグループに対して指定され、展開全体ではなく <vm_group> の下で指定されます）。
LCS::DEPLOY_UPDATE:: VM_SOFTWARE_VERSION_UPDATED	展開	ESC が VM のソフトウェアバージョンを更新した直後にトリガーされます （これは個々の VM のグループに対して指定され、展開全体ではなく <vm_group> の下で指定されます）。

LCS アクションを使用したリモートサーバからのファイルの取得

ESC リリース 4.0 より前では、外部設定ファイルを取得するためにファイルロケータを LCS アクションスクリプトに追加していました。ファイルロケータには、ファイルサーバへの参照と、ダウンロードするファイルへの相対パスが含まれています。ESC リリース 4.0 以降、ファイルロケータ属性は展開レベルで定義されます。つまり、ポリシーアクションやデイゼロ設定セクションではなく、展開コンテナの直下で定義されます。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>test-tenant</name>
      <deployments>
        <deployment>
          <name>test-deployment</name>
          <file_locators>
            <file_locator>
              <name>custom_bool_action</name>
              <remote_file>
                <file_server_id>http-my-server</file_server_id>
                <remote_path>share/qatest/custom_bool_action.sh</remote_path>
              </remote_file>
            </file_locator>
            <file_locator>
              <name>custom_bool_metric</name>
              <remote_file>
                <file_server_id>http-my-server</file_server_id>
                <remote_path>/share/qatest/custom_bool_metric.sh</remote_path>
              </remote_file>
            </file_locator>
          </file_locators>
          <!-- truncated for brevity -->
          <vm_group>
            <name>ASA-group</name>
```

```
                <!-- truncated for brevity -->
                <kpi_data>
                  <kpi>
                    <event_name>MY_CUSTOM_BOOL_ACTION</event_name>
                    <metric_value>5</metric_value>
                    <metric_cond>LT</metric_cond>
                    <metric_type>UINT32</metric_type>
                    <metric_occurrences_true>1</metric_occurrences_true>
                    <metric_occurrences_false>1</metric_occurrences_false>
                    <metric_collector>
                      <type>MY_CUSTOM_BOOL_METRIC</type>
                      <nicid>0</nicid>
                      <poll_frequency>3</poll_frequency>
                      <polling_unit>seconds</polling_unit>
                      <continuous_alarm>false</continuous_alarm>
                      <properties>
                        <!-- Add file locator reference here -->
                        <property>
                          <name>file_locator_name</name>
                          <value>custom_bool_action</value>
                        </property>
                      </properties>
                    </metric_collector>
                  </kpi>
                </kpi_data>
                <rules>
                  <admin_rules>
                    <rule>
                      <event_name>MY_CUSTOM_BOOL_ACTION</event_name>
                      <action>ALWAYS log</action>
                      <action>TRUE my_custom_bool_action</action>
                      <properties>
                        <!-- Add file locator reference here -->
                        <property>
                          <name>file_locator_name</name>
                          <value>custom_bool_action</value>
                        </property>
                      </properties>
                    </rule>
                  </admin_rules>
                </rules>
              </vm_group>
            </deployment>
          </deployments>
        </tenant>
      </tenants>
    </esc_datamodel>
```

詳細については、「リモートサーバからのファイルの取得」を参照してください。

ファイルを暗号化するには、「外部設定ファイルの認証」を参照してください。

さまざまなステージで定義されているライフサイクルステージ（LCS）ポリシーの条件

次の表に、データモデルで定義されているすべてのポリシー条件を示します。

表 *13 :* ライフサイクルステージ

条件名	範囲
LCS :: VM :: PRE_VM_DEPLOY	VM
LCS::VM::POST_VM_DEPLOYED	VM
LCS::VM::POST_VM_ALIVE	VM
展開のライフサイクルステージ	
LCS::PRE_DEPLOY	VM/展開
LCS::DEPLOY:: POST_VM_DEPLOYED	VM
LCS::POST_DEPLOY_ALIVE	展開
LCS::DEPLOY_ERR	展開
展開アップデートのライフサイクルステージ	
LCS::DEPLOY_UPDATE::POST_VM_ALIVE	VM
LCS::DEPLOY_UPDATE::	VM
LCS::DEPLOY_UPDATE:: POST_VM_VOLUME_DETACHED	VM
LCS::DEPLOY_UPDATE:: POST_VM_VOLUME_ATTACHED	VM
LCS::DEPLOY_UPDATE:: PRE_VM_SOFTWARE_VERSION_UPDATED	VM
リカバリのライフサイクルステージ	
LCS::POST_DEPLOY:: POST_VM_RECOVERY_COMPLETE	VM
LCS::POST_DEPLOY:: VM_RECOVERY_ERR	VM
リカバリと再展開のライフサイクルステージ	
LCS::POST_DEPLOY:: VM_RECOVERY_REDEPLOY_ERR	VM

第 26 章

アフィニティルールとアンチアフィニティルール

アフィニティルールとアンチアフィニティルール

アフィニティルールとアンチアフィニティルールでは、仮想マシン（VM）とホスト間の関係を作成します。ルールは、VM、または VM とホストに適用できます。ルールは、VM とホストをまとめて保持する（アフィニティ）か、分離します（アンチアフィニティ）。

ポリシーは、個々の VM の展開時に適用されます。既存の展開データモデルをアップロードするか、新しい展開データモデルを作成することで、ESC ポータルを介して単一の VN Fまたは複数の VNF をまとめて展開できます。詳細については、「ESC ポータルダッシュボード」を参照してください。

アフィニティポリシーとアンチアフィニティポリシーにより、展開プロセスが合理化されます。

アフィニティルールとアンチアフィニティルールは、展開時に作成されて、VM に適用されます。展開ワークフローが初期化されると、VM は配置ポリシーを受信します。

複合 VNF の展開中、いくつかの VM が相互に絶えず通信する必要がある場合、それらの VM をグループ化（アフィニティルール）して、同じホストに配置できます。

2 つの VM によりネットワークが過負荷になっている場合は、それらの VM を分離（アンチアフィニティルール）して、異なるホストに配置し、ネットワークのバランスを取ることができます。

展開時に VM とホストをグループ化または分離することで、ESC はネットワーク内の VM とホスト間の負荷を管理できます。これらの VM のリカバリとスケールアウトは、アフィニティルールとアンチアフィニティルールに影響を与えません。

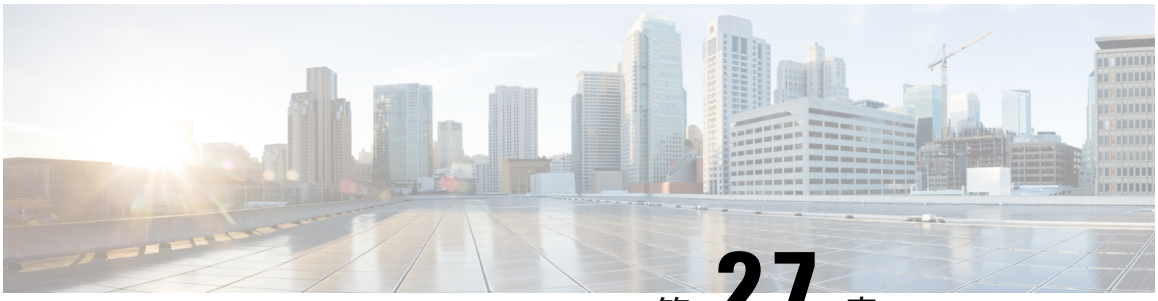
アンチアフィニティルールは、同じグループ内および異なるホストにある VM 間にも適用できます。これらの VM は同様の機能を実行し、相互にサポートします。一方のホストがダウンしても、もう一方のホストの VM は動作を継続し、サービスの損失を防ぎます。

次の表に、展開におけるアフィニティポリシーとアンチアフィニティポリシーのタイプを示します。

表 *14:* グループ内およびグループ間のアフィニティポリシーとアンチアフィニティポリシー

ポリシー	ポリシー	VM グループ	ホスト	ゾーン
アフィニティ	グループ内アフィニティ	同じ VM グループ	同じホスト	同じゾーン
	グループ間アフィニティ	異なる VM グループ	同じホスト	同じゾーン
アンチアフィニティ	グループ内アンチアフィニティ	同じ VM グループ	異なるホスト	同じゾーン
	グループ間アンチアフィニティ	異なる VM グループ	異なるホスト	同じゾーン

（注） ゾーンが OpenStack で指定されていない場合、VM はグループ間およびグループ内のアンチアフィニティルールに対して異なるホストおよび異なるゾーンに配置されます。



第 27 章

OpenStackのアフィニティルールとアンチアフィニティルール

OpenStack のアフィニティルールとアンチアフィニティルール

次の項では、アフィニティポリシーとアンチアフィニティポリシーについて例を挙げて説明します。

グループ内アフィニティポリシー

同じ VM グループ内の VNF は、同じホストまたは同じアベイラビリティゾーンに展開できます。

グループ内アフィニティポリシーの例：

```
<vm_group>
     <name>affinity-test-gp</name>
     <placement>
 <type>affinity</type>
 <enforcement>strict</enforcement>
     </placement>
...
```

タイプ *zone-host* は、同じホストまたは同じアベイラビリティゾーンに VNF を展開するために使用されます。

ゾーンまたはホストベースの配置

VNF は同じ VM グループ内にあり、同じホストまたは同じアベイラビリティゾーンに展開されます。*host* タグは同じホストに VM を展開するために使用され、*zone* タグは同じアベイラビリティゾーンに VM を展開するために使用されます。展開する前に、ホストが OpenStack に存在することを確認する必要があります。ESC は OpenStack 上の指定されたホストを検証します。*zone-host* タグは、配置のタイプを指定します。したがって、展開時にホストまたはゾーンが指定されていない場合、展開は失敗します。

重要　同じホストまたは同じアベイラビリティゾーンにVMを展開するために、ホストタグとゾーンタグの両方を指定することはできません。

ホスト配置の例：

```
<vm_group>
     <name>zone-host-test-gp1</name>
     <placement>
 <type>zone_host</type>
 <enforcement>strict</enforcement>
 <host>my-server</host>
     </placement>
...
```

ゾーン配置の例：

```
<vm_group>
     <name>zone-host-test-gp2</name>
     <placement>
 <type>zone_host</type>
 <enforcement>strict</enforcement>
        <zone>dt-zone</zone>
     </placement>
...
```

グループ内アンチアフィニティポリシー

同じVMグループ内のVNFは、異なるホストに明示的に展開されます。たとえば、バックアップ VNF などです。

グループ内アンチアフィニティポリシーの例：

```
<vm_group>
        <name>anti-affinity-test-gp</name>
        <placement>
            <type>anti_affinity</type>
            <enforcement>strict</enforcement>
        </placement>
...
```

グループ間アフィニティポリシー

同じ導入環境にあるが VM グループが異なる VNF は、同じホストに明示的に展開できます。例としては、VNFバンドルです。複数のVMグループは、vm_group_refタグを追加し、VMグループ名を値として 指定することで、このポリシーに従うことができます。

（注）　placement タグの下で 1 つ以上の vm_group_ref タグ、type タグ、および enforcement タグを使用できます。ホストやゾーンは指定できません。

グループ間アフィニティポリシーの例：

```
<deployments>
            <deployment>
            <name>intergroup-affinity-dep</name>
             <policies>
              <placement>
                <target_vm_group_ref>affinity-test-gp1</target_vm_group_ref>
                <type>affinity</type>
                <vm_group_ref>affinity-test-gp2</vm_group_ref>
                <enforcement>strict</enforcement>
               </placement>
          </policies>
…
```

グループ間アンチアフィニティポリシー

同じ導入環境にあるがVMグループが異なるVNFは、異なるホストに明示的に展開できます。たとえば、バックアップVNFや高可用性VNFなどです。複数のVMグループは、vm_group_refタグを追加し、VMグループ名を値として指定することで、このポリシーに従うことができます。

（注） placementタグの下で使用できる<target_vm_group_ref>タグ、typeタグ、およびenforcementタグは1つだけです。ホストまたはゾーンは指定できません。複数の<vm_group_ref>タグを使用できますが、アンチアフィニティポリシーは、 <vm_group_ref>およびそれらの<target_vm_group_ref>の間にのみ適用されます。つまり、許容できる<target_vm_group_ref>からそれぞれが別のホストに展開されている限り、2つ以上の<vm_group_ref>を同じホストに展開できます。

グループ間アンチアフィニティポリシーの例：

```
<deployments>
            <deployment>
             <name>intergroup-anti_affinity-dep</name>
              <policies>
               <placement>
                   <target_vm_group_ref>affinity-test-gp1</target_vm_group_ref>
                   <type>anti_affinity</type>
                   <vm_group_ref>affinity-test-gp2</vm_group_ref>
                   <enforcement>strict</enforcement>
                </placement>
          </policies>
…
```

マルチVIM展開では、配置ポリシーのVMグループは同じVIMに属している必要があります。つまり、VIMコネクタは（VMグループのロケータタグのvim_id属性で指定される）VMグループで同一である必要があります。VMグループ間のアフィニティポリシーおよびアンチアフィニティポリシーが異なるVIM上にある場合、ESCは展開を拒否します。複数展開でのVMの展開の詳細については、「複数のOpenStack VIMへのVNFの展開」を参照してください。

配置グループタグがポリシーの下に追加されます。それぞれの <placement_group> には次が含まれます。

- name：展開ごとに一意の名前。
- type：アフィニティまたはアンチアフィニティ
- enforcement：strict
- vm_group：各 vm_group の内容は、同じ展開でリストされた VM グループ名である必要があります。

配置グループタグは、配置ポリシー内に配置されます。配置ポリシーは、ターゲット VM グループと VM グループメンバー間の関係を記述します。配置グループポリシーは、すべての VM グループメンバー間の相互関係を記述します。配置グループポリシーは、ターゲット VM グループには適用されません。

次に、データモデルを示します。

```
<policies>
 <placement_group>
  <name>placement-affinity-1</name>
  <type>affinity</type>
  <enforcement>strict</enforcement>
  <vm_group>t1g1</vm_group>
  <vm_group>t1g2</vm_group>
  <vm_group>t1g7</vm_group>
 </placement_group>
 <placement_group>
  <name>placement-affinity-2</name>
  <type>affinity</type>
  <enforcement>strict</enforcement>
  <vm_group>t1g3</vm_group>
  <vm_group>t1g4</vm_group>
 </placement_group>
 <placement_group>
  <name>placement-affinity-3</name>
  <type>affinity</type>
  <enforcement>strict</enforcement>
  <vm_group>t1g5</vm_group>
  <vm_group>t1g6</vm_group>
 </placement_group>
 <placement_group>
  <name>placement-anti-affinity-1</name>
  <type>anti_affinity</type>
  <enforcement>strict</enforcement>
  <vm_group>t1g1</vm_group>
  <vm_group>t1g3</vm_group>
  <vm_group>t1g5</vm_group>
 </placement_group>
</policies>
```

（注） ポリシーの下の新しい配置グループタグでは、<target_vm_group_ref>および<vm_group_ref>が<vm_group>に置き換えられます。参照ベースのアフィニティタグとアンチアフィニティタグは廃止されました。

配置グループポリシーは、グループ間アフィニティおよびアンチアフィニティポリシーにのみ適用されます。

グループ間アフィニティおよびアンチアフィニティポリシーでは、配置タグと配置グループタグの両方を同時に使用することはできません。

配置グループ名タグは、配置グループポリシーごとに一意である必要があります。

制限事項

単一のVMは、アフィニティポリシーとアンチアフィニティポリシーの1つのサーバグループでのみ使用できます。

展開間アンチアフィニティポリシー

展開間アンチアフィニティルールは、ホストの配置に関して異なる展開間の関係を定義します。展開間のアンチアフィニティは、1つの展開のVMが、他の展開の他のVMと同じホストに配置されないように定義されます。

（注） 展開間アンチアフィニティは、OpenStackでのみサポートされます。展開間アンチアフィニティは、ホストの配置（アフィニティまたはアンチアフィニティ）では機能しません。これは、後者が展開間アンチアフィニティルールよりも優先されるためです。

ESCデータモデルでは、アンチアフィニティグループを使用して展開間アンチアフィニティが定義されます。アンチアフィニティグループのすべてのメンバー展開には、メンバー間にアンチアフィニティ関係があります。たとえば、3つの展開dep-1、dep-2、およびdep-3を持つdefault-antiと呼ばれるアンチアフィニティグループでは、dep-1はdep-2およびdep-3展開に対するアンチアフィニティ、dep-2はdep-1およびdep-3展開に対するアンチアフィニティ、dep-3はdep-1およびdep-2展開に対するアンチアフィニティです。展開では、以下に示すように、関連するすべてのグループ名を参照することによって、アンチアフィニティグループのメンバーシップを指定します。

```
<deployment>
<name>VPC-dep</name>
 <deployment_groups>
  <anti_affinity_group>VPC-ANTI-AFFINITY</anti_affinity_group>
  <anti_affinity_group>VPNAAS-ANTI-AFFINITY</anti_affinity_group>
 </deployment_groups>
….
</deployment>
```

前述の例では、VPC-dep は 2 つのアンチアフィニティグループに属しています。これら 2 つのグループのいずれかを参照する他の展開は、VPC-dep とのアンチアフィニティ関係を持ちます。

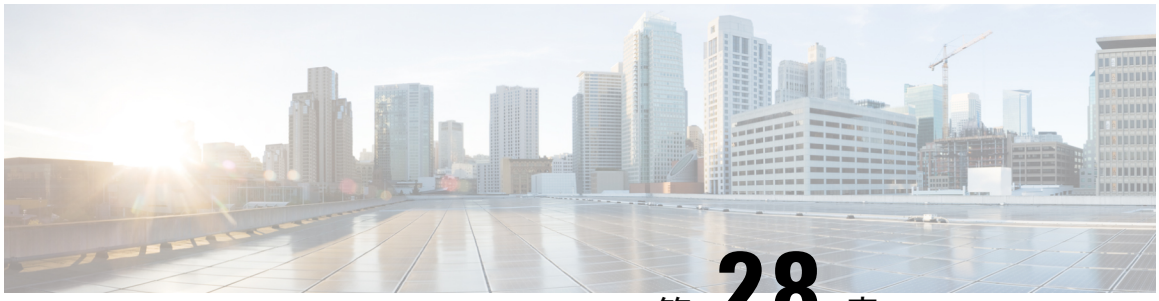
展開間配置グループ

アンチアフィニティグループは、配置グループの例です。アンチアフィニティグループには、ESC に次のプロパティがあります。

- 配置グループを作成または削除する必要はありません。
- 配置グループは、最初は 1 つの展開と複数の展開で同時に参照できます。
- 配置ルールは、次のようなサービスの展開フェーズで適用されます。
  - 初期展開
  - スケール アウト
  - VM グループ更新の追加
  - VM グループの最小スケーリング更新（VM を追加するための最小スケーリングの増加）
  - リカバリ

複数の VIM 展開は、展開間アンチアフィニティをサポートします。ただし、次の場合、ESC は展開を拒否します。

- 複数の VIM 展開（VM グループ内のロケータを使用）とデフォルトの VIM 展開（ロケータを使用しない）との間で展開間アンチアフィニティポリシーが定義されている場合。
- 展開間アンチアフィニティグループのすべての展開が同じ VIM（同じ vim_id）に展開されていない場合。複数の VIM 展開の詳細については、複数の OpenStack VIM への VNF の展開 （119 ページ）を参照してください。



第 28 章

VMware vCenter のアフィニティルールとアンチアフィニティルール

VMware vCenter のアフィニティルールとアンチアフィニティルール

VMware vCenter のアフィニティルールとアンチアフィニティルールについて例を挙げて説明します。これらのルールは、クラスタおよびターゲットホスト用に作成されます。

すべての VMware vCenter 展開には、常にゾーンホスト配置ポリシーが必要です。ゾーンホストでは、クラスタまたはホストのいずれかであるターゲット VM グループを定義します。

グループ内アフィニティポリシー

同じ VM グループを持つ VNF は、同じホストに展開できます。

展開中に、ESC はアフィニティのアンカー VM として最初の VM を展開します。同じアフィニティルールに従う他のすべての VM は、アンカー VM と同じホストに展開されます。アンカー VM の展開は、リソース使用率の最適化に役立ちます。

グループ内アフィニティポリシーの例：

```
…
<vm_group>
<name>vm-gp</name>
…
<placement>
<type>zone_host</type>
<enforcement>strict</enforcement>
<zone>cluster1</zone>
</placement>
<placement>
<type>affinity</type>
<enforcement>strict</enforcement>
</placement>
```

…

（注） strict 属性のみが適用されます。

（注） ホスト配置ポリシーを備えたアフィニティおよびアンチアフィニティポリシーが正しくないため、展開が失敗する可能性があります。ホスト配置を単独で（VM グループ内のアフィニティおよびアンチアフィニティ配置ポリシーなし）使用して、グループ内アフィニティを実現できます。

グループ内アンチアフィニティ

同じ VM グループを持つ VNF は、異なるホストに展開できます。展開時に、ESC は同じ VM グループを使用して VNF を次々に展開します。各 VNF 展開の最後に、ESC はそのホストをリストに記録します。各 VNF 展開の開始時に、ESC はリストに含まれていないコンピューティングホストに VNF を展開します。使用可能なすべてのコンピューティングホストがリストに含まれている場合、ESC による展開全体が失敗します。

グループ内アンチアフィニティポリシーの例：

```
…
<vm_group>
<name>vm-gp</name>
…
<placement>
<type>zone_host</type>
<enforcement>strict</enforcement>
<zone>cluster1</zone>
</placement>
<placement>
<type>anti_affinity</type>
<enforcement>strict</enforcement>
</placement>
```

クラスタの配置

VM グループ内のすべての VM をクラスタに展開できます。たとえば、VM グループ CSR-gp1 内のすべての VM をクラスタ dc-cluster2 に展開できます。

（注） VMware vCenter クラスタは、管理者が作成する必要があります。

クラスタ配置の例：

```
<name>CSR-gp1</name>
          <placement>
```

```
    <type>zone_host</type>
    <enforcement>strict</enforcement>
    <zone>dc-cluster2</zone>
</placement>
```

ホストの配置

VM グループ内のすべての VMS をホストに展開できます。たとえば、VM グループ CSR-gp1 のすべての VM がホスト 10.2.0.2 に展開されます。

```
<name>CSR-gp1</name>
           <placement>
             <type>zone_host</type>
             <enforcement>strict</enforcement>
              <host>10.2.0.2</host>
            </placement>
```

グループ間アフィニティポリシー

異なる VM グループの VM を同じホストに展開できます。たとえば、VM グループ ASA-gp1 内のすべての VM を、VM グループ CSR-gp1 内の VM と同じホストに展開できます。

展開中、ESC は最初の VM をアンカー VM として展開します。同じアフィニティルールに従う他のすべての VM は、アンカー VM と同じホストに展開されます。

（注） グループ間アフィニティルールが単一のクラスタ内に適用されるようにするには、展開内のすべての VM グループが同じクラスタ（ESC data_model の <zone>）に指定されていることを確認します。

グループ間アフィニティポリシーの例：

```
<deployment>
<deployment>
<name>test-affinity-2groups</name>
<policies>
<placement>
<target_vm_group_ref>CSR-gp1</target_vm_group_ref>
<type>affinity</type>
<vm_group_ref>CSR-gp2</vm_group_ref>
<vm_group_ref>ASA-gp1</vm_group_ref>
<enforcement>strict</enforcement>
</placement>
</policies>
```

グループ間アンチアフィニティポリシー

同じ導入環境にあるが VM グループが異なる VNF は、異なるホストに明示的に展開できます。たとえば、バックアップ VNF や高可用性 VNF などです。複数の VM グループは、vm_group_ref タグを追加し、VM グループ名を値として指定することで、このポリシーに従うことができます。

（注） placement タグの下で使用できる <target_vm_group_ref> タグ、type タグ、および enforcement タグは 1 つだけです。ホストまたはゾーンは指定できません。複数の <vm_group_ref> タグを使用できますが、アンチアフィニティポリシーは、 <vm_group_ref> およびそれらの <target_vm_group_ref> の間にのみ適用されます。つまり、許容できる <target_vm_group_ref> からそれぞれが別のホストに展開されている限り、2 つ以上の <vm_group_ref> を同じホストに展開できます。

グループ間アンチアフィニティポリシーの例：

```
<deployments>
           <deployment>
            <name>intergroup-anti_affinity-dep</name>
             <policies>
              <placement>
                  <target_vm_group_ref>affinity-test-gp1</target_vm_group_ref>
                  <type>anti_affinity</type>
                  <vm_group_ref>affinity-test-gp2</vm_group_ref>
                  <enforcement>strict</enforcement>
               </placement>
          </policies>
…
```

マルチ VIM 展開では、配置ポリシーの VM グループは同じ VIM に属している必要があります。つまり、VIM コネクタは（VM グループのロケータタグの vim_id 属性で指定される）VM グループで同一である必要があります。VM グループ間のアフィニティポリシーおよびアンチアフィニティポリシーが異なる VIM 上にある場合、ESC は展開を拒否します。複数展開での VM の展開の詳細については、「複数の OpenStack VIM への VNF の展開」を参照してください。

配置グループタグがポリシーの下に追加されます。それぞれの <placement_group> には次が含まれます。

- name：展開ごとに一意の名前。
- type：アフィニティまたはアンチアフィニティ
- enforcement：strict
- vm_group：各 vm_group の内容は、同じ展開でリストされた VM グループ名である必要があります。

配置グループタグは、配置ポリシー内に配置されます。配置ポリシーは、ターゲット VM グループと VM グループメンバー間の関係を記述します。配置グループポリシーは、すべての VM グループメンバー間の相互関係を記述します。配置グループポリシーは、ターゲット VM グループには適用されません。

次に、データモデルを示します。

```
<policies>
 <placement_group>
  <name>placement-affinity-1</name>
```

```
  <type>affinity</type>
  <enforcement>strict</enforcement>
  <vm_group>t1g1</vm_group>
  <vm_group>t1g2</vm_group>
  <vm_group>t1g7</vm_group>
 </placement_group>
 <placement_group>
  <name>placement-affinity-2</name>
  <type>affinity</type>
  <enforcement>strict</enforcement>
  <vm_group>t1g3</vm_group>
  <vm_group>t1g4</vm_group>
 </placement_group>
 <placement_group>
  <name>placement-affinity-3</name>
  <type>affinity</type>
  <enforcement>strict</enforcement>
  <vm_group>t1g5</vm_group>
  <vm_group>t1g6</vm_group>
 </placement_group>
 <placement_group>
  <name>placement-anti-affinity-1</name>
  <type>anti_affinity</type>
  <enforcement>strict</enforcement>
  <vm_group>t1g1</vm_group>
  <vm_group>t1g3</vm_group>
  <vm_group>t1g5</vm_group>
 </placement_group>
</policies>
```

（注） ポリシーの下の新しい配置グループタグでは、<target_vm_group_ref> および <vm_group_ref> が <vm_group> に置き換えられます。参照ベースのアフィニティタグとアンチアフィニティタグは廃止されました。

配置グループポリシーは、グループ間アフィニティおよびアンチアフィニティポリシーにのみ適用されます。

グループ間アフィニティおよびアンチアフィニティポリシーでは、配置タグと配置グループタグの両方を同時に使用することはできません。

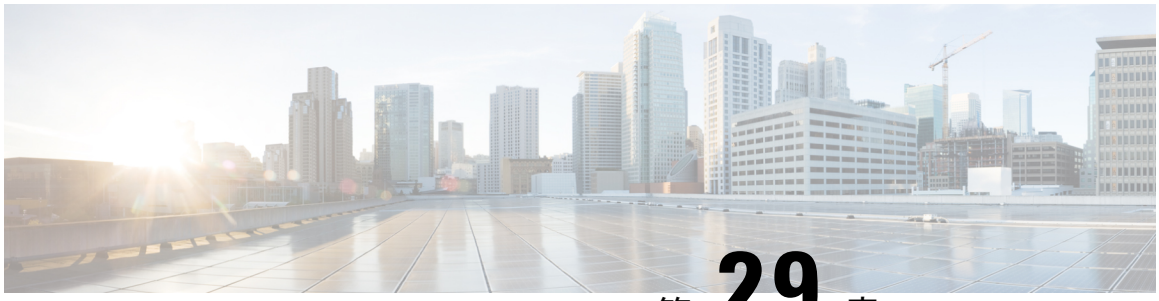
配置グループ名タグは、配置グループポリシーごとに一意である必要があります。

制限事項

アフィニティルールとアンチアフィニティルールが VMware vCenter に適用される場合の制限は次のとおりです。

- VMware vCenter で定義されたすべてのアフィニティルールは、クラスタに実装されます。
- DPM、HA、および vMotion はオフにする必要があります。
- VM の展開とリカバリは ESC によって管理されます。
- DRS がオンになっている場合は、手動モードに設定する必要があります。

- DRS 展開を活用するには、共有ストレージが必要です。
- <enforcement> タグでサポートされる値は「strict」にする必要があります。
- <zone_host> を VM グループに使用する必要があります。



第 29 章

VMware vCloud Director のアフィニティルールとアンチアフィニティルール

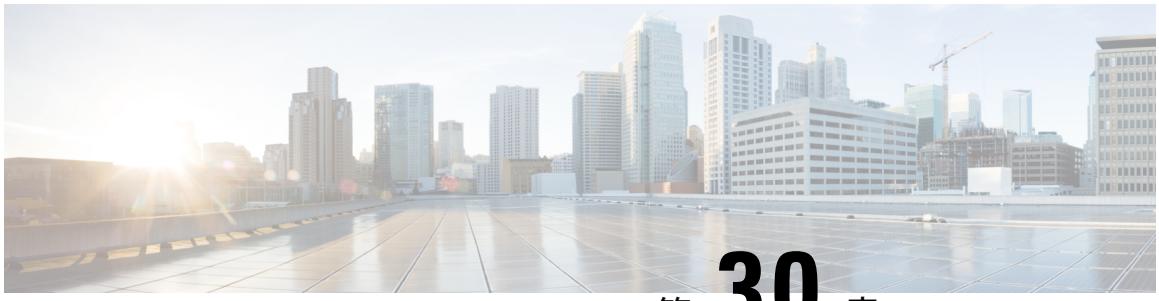
VMware vCloud Director のアフィニティルールとアンチアフィニティルール

ESC は、vCD のアフィニティおよびアンチアフィニティ配置ポリシーをサポートします。ただし、ゾーンホスト配置ポリシーはサポートされません。

ESC のアフィニティとアンチアフィニティの実装は、vCloud Director のアフィニティルール（vSphere の VM-VM アフィニティルール）に依存します。次の例は、vCD VNF 展開データモデルのアフィニティルールとアンチアフィニティルールを示しています。

```
<deployments>
        <deployment>
          <!-- vApp instance name -->
          <name>d1</name>
          <policies>
            <placement_group>
              <name>d1-placement-affinity-1</name>
              <type>affinity</type>
              <enforcement>strict</enforcement>
              <vm_group>g1</vm_group>
              <vm_group>g2</vm_group>
            </placement_group>
          </policies>
         …….....
         …….....
       </deployment>
    </deployments>
```

vCD の展開については、VMware vCloud Director（vCD）での仮想ネットワーク機能の展開 （152 ページ）を参照してください。



第 30 章

カスタム VM 名の設定

カスタム VM 名の設定

ESC で VM 名を自動生成しない場合は、VM 名をカスタマイズできます。VM 名をカスタマイズするには、展開データモデルの VM グループセクションで vim_vm_name を指定します。vim_vm_name が指定されていない場合、ESC によって VM 名が自動生成されます。

カスタム名の指定時に、VM グループに複数の VM が存在する場合、出力のカスタム VM 名に「_<index>」が追加されます。たとえば、グループ内の最初の VM は vim_vm_name で指定された名前になり、2 番目以降の VM のカスタム名にはインデックス「_1」、「_2」が追加されます。ABC として指定されたカスタム名の場合、出力の VM 名は、VMname、VMname_1、VMname_2 などと表示されます。VM グループ内の VM が 1 つだけの場合、カスタム VM 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_vm_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_vm_name を指定すると、すべての VM の名前が前述のように生成されます。2 番目の VM グループでは vim_vm_name を指定しないため、このグループから作成されるすべての VM 名は自動生成されます。

カスタム VM 名は、1 つの OpenStack 展開における展開およびテナント内で一意である必要があります。つまり、カスタム VM 名は異なるテナント間で複製できます。また、異なる展開用であれば、同じテナント内でも複製できます。VMware 展開の場合、カスタム VM 名は vCenter サーバ全体で一意である必要があります。つまり、重複する VM 名は許可されません。

（注） カスタム名には最大 63 文字を使用できます。VM 名には特殊文字は使用できず、英数字と「_」および「-」のみ使用できます。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"> <tenants><tenant>
  <name>Admin</name>
   <deployments>
    <deployment>
```

```
 <deployment_name>NwDepModel_nosvc</deployment_name>

<vm_group>
<name>CIRROS</name>
<image>Automation-Cirros-Image</image>
<flavor>Automation-Cirros-Flavor</flavor>
<vim_vm_name>VMname</vim_vm_name>
<scaling>
    <min_active>1</min_active>
    <max_active>2</max_active>
    <elastic>true</elastic>
</scaling>
</vm_group>
```

（注）
- ESC ポータルには、展開時に設定された VM 名は表示されません。
- 重複する VM 名は、VMWare ではサポートされていません。
- 展開の完了後に VM 名を変更することはできません。

次に、カスタム VM 名を含む出力例を示します。展開時に vim_vm_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 によって VM 名が自動生成されます。

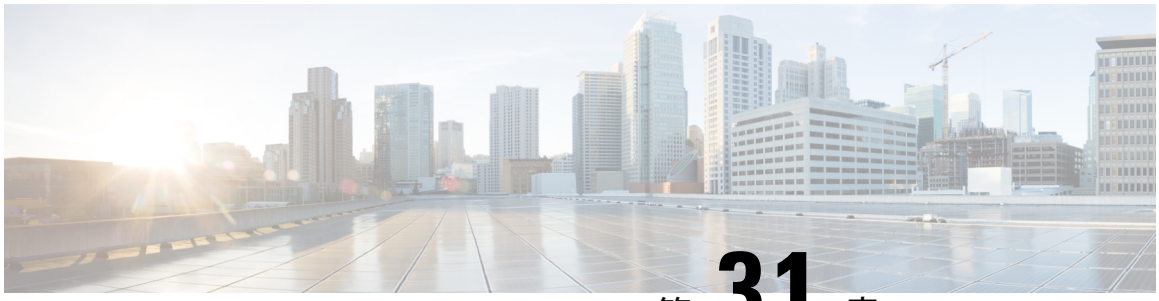
- 次に、カスタム VM 名を追加した後に esc_nc_cli スクリプトを使用して取得した運用データの出力例を示します。<vmname> という新しい要素が vm_group 要素の下に表示されます。<status_message> フィールドの値も、カスタム VM 名を反映するために更新されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
      <opdata>
        <tenants>
          <tenant>
            <name>xyzzy</name>
            <deployments>
              <deployment_name>my-deployment-123</deployment_name>
              <deployment_id>78d48bf8-5f67-45fc-8d92-5ad4676yf57</deployment_id>
              <vm_group>
                <name>Grp1</name>
                <vm_instance>
                  <vm_id>df108144-ec4f-4d66-a62f-98096ecddef0</vm_id>
                  <name>VMname</name>
              </vm_group>
```

- 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"?>
<deployment xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <datacenter>
    <default>false</default>
  </datacenter>
  <deployment_details>
```

```
    <host_uuid>8623f1476302a5815608dbd4c2f836c570e8c74cbfbaff41c78564b1</host_uuid>

    <host_name>my-server</host_name>
    <vm_uuid>e7e5a905-e0c7-4652-ae1f-23a409a58219</vm_uuid>
    <interfaces>
      <interface>

      </interface>
    </interfaces>
    <vm_group_name>Grp1</vm_group_name>
    <vm_name>VMname_1</vm_name><-- ##### custom vm name, single VM in the VM group,
 so no appended "_<index>" -->
    <vm_state_machine_state>VM_ALIVE_STATE</vm_state_machine_state>
  </deployment_details>
</deployment>
```



第 31 章

既存の展開の管理

展開が正常に作成されたら、展開内のリソースを更新できます。展開管理の一環として、リソースを追加または削除したり、既存のリソースの設定を更新したりできます。これらの更新は、実行中の展開で行うことができます。この章では、これらのリソースの管理について詳しく説明します。

既存の展開の更新

新しいVMグループ、インターフェイス、ネットワークなどを追加することで、既存の展開を更新できます。VM グループのデイゼロ設定、KPI、およびルールも更新できます。vm_group の追加や削除、vm_group 内のエフェメラルネットワークの追加や削除、VM グループ内のインターフェイスの追加や削除は、正常に展開された後に実行できます。

OpenStack では、vm_group、エフェメラルネットワーク vm_group、およびインターフェイスの追加や削除など、すべての更新を単一の展開で実行できます。

サービスの更新中に、自動リカバリアクションによってサービスが不整合な状態になることがあります。自動リカバリアクションがトリガーされるのを防ぐため、モニタはサービス更新ワークフローの前に無効になり、更新の完了後に有効になります。

（注） サービス更新要求の途中での VM リカバリ中、VM リカバリ通知を受信する前に、ノースバウンドクライアントが SERVICE_UPDATED FAILURE 通知を受信することがあります。そのため、サービスが SUCCESS または ERROR 状態に移行するのを確認してから、手動リカバリやその他のサービスレベル要求を送信することを推奨します。

既存の展開の更新は、OpenStack と VMware vCenter の両方でサポートされています。次の表に、既存の展開で更新できるコンポーネントを示します。

表 *15 : OpenStack*、*VMware vCenter*、および *vCloud Director* での既存の展開の更新

更新	OpenStack	VMware vCenter	vCloud Director
VM グループの追加	サポート対象	サポート対象	サポート対象
VM グループの削除	サポート対象	サポート対象	サポート対象
サービスがエラー状態になっている場合の VM グループの削除	サポート対象	サポート対象	サポート対象外
エフェメラルネットワークの追加	サポート対象	サポート対象外	サポート対象外
エフェメラルネットワークの削除	サポート対象	サポート対象外	サポート対象外
インターフェイスの追加	サポート対象	サポート対象外	サポート対象外
インターフェイスの削除	サポート対象	サポート対象外	サポート対象外
インターフェイスの更新	サポート対象	サポート対象	サポート対象外
スタティック IP プールの追加	サポート対象	サポート対象外	サポート対象外
スタティック IP プールの削除	サポート対象	サポート対象外	サポート対象外
VM グループのデイゼロ設定の更新	サポート対象	サポート対象	サポート対象外
KPI とルールの更新	サポート対象	サポート対象	サポート対象外
VM グループの VM 数（スケールインまたはスケールアウト）の更新	サポート対象	サポート対象	サポート対象外
リカバリ待機時間の更新	サポート対象	サポート対象	サポート対象外

更新	OpenStack	VMware vCenter	vCloud Director
リカバリポリシーの更新	サポート対象	サポート対象外	サポート対象外
イメージの更新	サポート対象	サポート対象外	サポート対象外

（注） 複数の OpenStack VIM における既存の展開の更新もサポートされています。ただし、VM グループ内のロケータ属性は更新できません。複数の VIM への VM の展開の詳細については、「複数の OpenStack VIM への VNF の展開」を参照してください。

VM グループの追加

既存のイメージとフレーバーを使用して、実行中の展開で vm_group を追加したり、削除したりできます。

vm_group を追加する NETCONF 要求：

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"> <tenants><tenant>
  <name>Admin</name>
   <deployments>
    <deployment>
     <deployment_name>NwDepModel_nosvc</deployment_name>

     <vm_group>
        <image></image>
        <Flavor></Flavor>
       .........
    </vm_group>
    <vm_group>
  <image></image>
 <Flavor></Flavor>
 .........
   </vm_group>
   <vm_group>
 <image></image>
 <Flavor></Flavor>
 .........
   </vm_group>
  </deployment>
</deployments>
    </tenant></tenants>
</esc_datamodel>
```

VM グループが正常に追加されたときの NETCONF 通知：

```
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
           VM_DEPLOYED
             VM_ALIVE
          SERVICE_UPDATED
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
```

VM グループの削除

vm_group を削除する NETCONF 要求：

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
 <tenants><tenant>
  <name>Admin</name>
   <deployments>
    <deployment>
     <deployment_name>NwDepModel_NoSvc</deployment_name>

     <vm_group>
        <image></image>
        <Flavor></Flavor>
       .........
    </vm_group>
    <vm_group nc:operation="delete">
 <image></image>
 <Flavor></Flavor>
 .........
   </vm_group>
   <vm_group nc:operation="delete">
 <image></image>
 <Flavor></Flavor>
  .........
   </vm_group>
  </deployment>
</deployments>
    </tenant></tenants>
</esc_datamodel>
```

vm_group が正常に削除されたときの NETCONF 通知：

```
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
             VM_UNDEPLOYED
             SERVICE_UPDATED
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
```

エラー状態の VM グループの削除

展開の更新を実行することで、展開がエラー状態のときにVMグループを削除できるようになりました。ただし、1 つ以上の VM グループの追加や、特定の VM グループの削除中に別の VM グループの属性値を変更するなど、VM グループに追加の設定を行うことはできません。

VM グループへのエフェメラルネットワークの追加

既存のイメージとフレーバを使用して、vm_group にエフェメラルネットワークを追加できます。

vm_group にエフェメラルを追加する NETCONF 要求：

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"> <tenants><tenant>
  <name>Admin</name>
   <deployments>
    <deployment>
     <deployment_name>NwDepModel_nosvc</deployment_name>
      <networks>
      <network>
 .........
 </network>
 <network>
 .........
 </network>
 <network>
 .........
 </network>
```

```
     </networks>
     <vm_group>
        <image></image>
        <Flavor></Flavor>
       .........
    </vm_group>
  </deployment>
</deployments>
    </tenant></tenants>
</esc_datamodel>
```

エフェメラルネットワークが vm_group に正常に追加されたときの NETCONF 通知：

```
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
           CREATE_NETWORK
           CREATE_SUBNET
          SERVICE_UPDATED
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
```

VM グループのエフェメラルネットワークの削除

vm_group のエフェメラルネットワークを削除する NETCONF 要求

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"> <tenants><tenant>
  <name>Admin</name>
   <deployments>
    <deployment>
     <deployment_name>NwDepModel</deployment_name>
      <networks>
      <network nc:operation="delete">
 .........
 </network>
 <network>
 .........
 </network>
 <network nc:operation="delete">
 .........
 </network>
     </networks>
     <vm_group>
        <image></image>
        <Flavor></Flavor>
       .........
    </vm_group>
  </deployment>
</deployments>
    </tenant></tenants>
</esc_datamodel>
```

vm_group 内のエフェメラルネットワークが正常に削除されたときの NETCONF 通知：

```
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
          DELETE_SUBNET
          DELETE_NETWORK
          SERVICE_UPDATED
UPDATE SERVICE REQUEST RECEIVED (UNDER TENANT)
```

VM グループへのインターフェイスの追加（OpenStack）

既存のイメージとフレーバを使用して、実行中の展開から vm_group にインターフェイスを追加できます。

vm_group にインターフェイスを追加する NETCONF 要求：

```
<interfaces>
  <interface>
    <nicid>0</nicid>
    <network>my-network</network>
  </interface>
  <interface>
    <nicid>1</nicid>
    <network>utr-net</network>
  </interface>
  <interface>
    <nicid>2</nicid>
    <network>utr-net-1</network>
  </interface>
</interfaces>
```

（注） ESC リリース 2.3 以降では、ESC Portal for OpenStack を使用したインターフェイスの追加と削除がサポートされています。

ESC は、REST API と NETCONF API の両方を使用した vm_group からのインターフェイスの追加と削除をサポートします。

VM グループのインターフェイスの削除（OpenStack）

vm_group のインターフェイスを削除する NETCONF 要求：

```
<interfaces>
  <interface>
    <nicid>0</nicid>
    <network>my-network</network>
  </interface>
  <interface>
    <nicid>1</nicid>
    <network>utr-net</network>
  </interface>
  <interface nc:operation="delete">
    <nicid>2</nicid>
    <network>utr-net-1</network>
  </interface>
</interfaces>
```

同じ展開要求で、VM グループ（OpenStack のみ）のインターフェイスを同時に追加および削除できます。

（注） ESC は以下をサポートしていません。

- 既存の vm_group、ネットワーク、またはサブネットのプロパティの更新。
- vm_group のイメージとフレーバの更新。
- リソース名の空白名（つまり、vm_group、ネットワーク、サブネット、またはインターフェイス）。

Cisco ESC リリース 2.0 以前では、エフェメラルネットワークまたはサブネットは追加または削除のみが可能です。

ESCは、展開の更新中に追加された新しいインターフェイスのデイゼロ設定をサポートしません。デイN設定の一部として、VNF で追加設定を個別に実行する必要があります。トークン置換を使用してインターフェイスを削除した場合、そのインターフェイスを削除するには、デイゼロ設定を更新する必要があります。将来、ESCは新しいデイゼロ設定をリカバリに使用します。

NIC ID のない新しいインターフェイスは、展開の更新時に設定されません。

既存のデイゼロ設定を持つ新しいインターフェイスが設定されます。

インターフェイスの更新（OpenStack）

OpenStackでインターフェイスを更新すると、以前のインターフェイスが削除され、既存のNIC ID を持つ新しいインターフェイスが作成されます。

次に、データモデルを示します。

```
<interfaces>
                <interface>
                  <nicid>0</nicid>
                  <network>my-network</network>
                </interface>
                 <interface>
                  <nicid>1</nicid>
                  <network>utr-net-2</network>
                </interface>
        </interfaces>
```

VM_UPDATED 通知は VM 内のすべてのインターフェイスの詳細とともに送信され、ワークフローの更新後に SERVICE_UPDATED 通知が送信されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-07-25T00:45:27.64+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>VM has been updated successfully. vm:
utr-80__7515__utr-80__utr-80utr-80utr-801.2__0__utr-80__0</status_message>
    <svcname>utr-80</svcname>
    <svcversion>1.2</svcversion>
    <depname>utr-80</depname>
    <tenant>utr-80</tenant>
```

```
    <svcid>c1294ad1-fd7b-4a73-8567-335160dce90f</svcid>
    <depid>ecedf755-502c-473a-82f2-db3a5485fdf5</depid>
    <vm_group>utr-80</vm_group>
    <vm_source>
      <vmid>4b20024f-d8c8-4b1a-8dbe-3bf1011a0bcb</vmid>
      <hostid>71c7f3afb281485067d8b28f1734ec6b63f9e3225045c581168cc39d</hostid>
      <hostname>my-server</hostname>
      <interfaces>
        <interface>
          <nicid>0</nicid>
          <port_id>6bbafbf5-51a1-48c0-a4a5-cd6092657e5c</port_id>
          <network>7af5c7df-6246-4d53-91bd-aa12a1607656</network>
          <subnet>7cb6815e-3023-4420-87d8-2b10efcbe14e</subnet>
          <ip_address>192.168.0.10</ip_address>
          <mac_address>fa:16:3e:bc:07:d5</mac_address>
          <netmask>255.255.255.0</netmask>
          <gateway>192.168.0.1</gateway>
        </interface>
        <interface>
          <nicid>1</nicid>
          <port_id>6d54d3a8-b793-40b8-9a32-c7e2f08e0917</port_id>
          <network>4f85613a-d3fc-4b49-9cb0-b91d4360918b</network>
          <subnet>c3724a64-ffed-43b6-aba8-63287c5344ea</subnet>
          <ip_address>10.91.90.2</ip_address>
          <mac_address>fa:16:3e:49:d0:00</mac_address>
          <netmask>255.255.255.0</netmask>
          <gateway>10.91.90.1</gateway>
        </interface>
        <interface>
          <nicid>3</nicid>
          <port_id>04189123-fc7a-4418-877b-61c24a5e8508</port_id>
          <network>f9c7978f-800e-4bfc-bc20-1c29acef87d9</network>
          <subnet>63ae5e39-c41a-4b28-9ac7-ed94b5e477b0</subnet>
          <ip_address>172.16.0.97</ip_address>
          <mac_address>fa:16:3e:5e:2e:e3</mac_address>
          <netmask>255.240.0.0</netmask>
          <gateway>172.16.0.1</gateway>
        </interface>
      </interfaces>
    </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>VM_UPDATED</type>
    </event>
  </escEvent>
</notification>
```

（注）
- インターフェイスは、NIC ID に基づいて一意です。新しいインターフェイスを追加する場合は、異なる NIC ID を使用する必要があります。インターフェイスが編集され、同じ NIC ID を使用する場合、既存のインターフェイスの更新と見なされます。

インターフェイスの更新（VMware vCenter）

既存の展開を更新しながら、インターフェイスに関連付けられたネットワークを更新できます。展開要求の古いネットワーク名を新しい名前に置き換えて、ネットワークを更新します。

インターフェイスのポートグループは、ネットワーク更新中にVMグループ内のすべてのVMで更新されます。

（注） IPの更新は、VMware vCenterでのインターフェイスの更新中はサポートされません。

VMグループ内の最小値が1を超える場合、VMware vCenterでのインターフェイスの更新中は、静的IPおよびMACプールの更新はサポートされません。

次に、データモデルの更新を示します。

既存のデータモデル：

```
<interface>
  <nicid>1</nicid>
  <network>MgtNetwork</network>
</interface>
```

新しいデータモデル：

```
<interface>
  <nicid>1</nicid>
  <network>VNFNetwork</network>
</interface>
```

更新が成功すると、次の通知が送信されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-08-17T12:03:12.518+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Updated 1 interface: [net=VNFNetwork,nicid=1]</status_message>
    <depname>u1-asa</depname>
    <tenant>admin</tenant>
    <tenant_id>SystemAdminTenantId</tenant_id>
    <depid>90139aa1-9705-4b07-9963-d60691d3b0ad</depid>
    <vm_group>utr-asa-1</vm_group>
    <vm_source>
      <vmid>50261fbc-88a0-8601-71a9-069460720d4f</vmid>
      <hostid>host-10</hostid>
      <hostname>172.16.103.14</hostname>
      <interfaces>
        <interface>
          <nicid>1</nicid>
          <type>virtual</type>
          <port_id/>
          <network>VNFNetwork</network>
          <subnet/>
          <ip_address>192.168.0.254</ip_address>
          <mac_address>00:50:56:a6:d8:1d</mac_address>
        </interface>
      </interfaces>
    </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>VM_UPDATED</type>
    </event>
  </escEvent>
```

```
</notification>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-08-17T12:03:12.553+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Service group update completed successfully</status_message>
    <depname>u1-asa</depname>
    <tenant>admin</tenant>
    <tenant_id>SystemAdminTenantId</tenant_id>
    <depid>90139aa1-9705-4b07-9963-d60691d3b0ad</depid>
    <vm_source>
  </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>SERVICE_UPDATED</type>
    </event>
  </escEvent>
</notification>
```

インターフェイスの更新（クラウド サービス プラットフォーム）

CSP展開のインターフェイス拡張を使用して、インターフェイスのVLAN、タイプ、および帯域幅のプロパティを設定および更新できます。管理ステータス（admin_state_up）およびネットワーク属性は、インターフェイスで設定および更新できます。

コンテナ名属性は、**nicid** 値と一致する必要があります。たとえば、**コンテナ名**が 1 の場合、インターフェイスプロパティを設定および更新するには、**nicid** 値も 1 にする必要があります。

Vlan

vlan プロパティを設定および更新するには、ESC から次のコマンドを実行します。

esc_nc_cli --user <username> --password <password> edit-config interfaceVlan.xml

サンプルの interfaceVlan.xml は次のとおりです。

```
<interfaces>
   <interface>
    <nicid>0</nicid>
    <type>virtual</type>
    <model>virtio</model>
    <network>Eth0-2</network>
    <ip_address>192.168.24.45</ip_address>
    <admin_state_up>true</admin_state_up>
  </interface>
 <interface>
    <nicid>1</nicid>
    <type>virtual</type>
    <model>virtio</model>
    <network>Eth0-2</network>
    <admin_state_up>true</admin_state_up>
  </interface>
</interfaces>
......
  .....
<extensions>
  <extension>
    <name>interfaces</name>
```

```
  <containers>
     <container>
      <name>0</name>
      <properties>
        <property>
          <name>passthroughMode</name>
          <value>none</value>
        </property>
        <property>
          <name>tagged</name>
          <value>false</value>
        </property>
        <property>
          <name>type</name>
          <value>access</value>
        </property>
        <property>
          <name>vlan</name>
          <value>1</value>
        </property>
      </properties>
    </container>
   <container>
      <name>1</name>
      <properties>
        <property>
          <name>passthroughMode</name>
          <value>none</value>
        </property>
        <property>
          <name>tagged</name>
          <value>false</value>
        </property>
        <property>
          <name>type</name>
          <value>access</value>
        </property>
        <property>
          <name>bandwidth</name>
          <value>750</value>
        </property>
        <property>
          <name>vlan</name>
          <value>11</value>
        </property>
      </properties>
    </container>
  </containers>
</extension>
<extension>
  <name>serial_ports</name>
  <containers>
    <container>
      <name>0</name>
      <properties>
        <property>
          <name>serial_type</name>
          <value>console</value>
        </property>
      </properties>
    </container>
  </containers>
</extension>
<extension>
```

```
    <name>image</name>
    <properties>
      <property>
        <name>disk-resize</name>
        <value>true</value>
      </property>
      <property>
        <name>disk_type</name>
        <value>virtio</value>
      </property>
    </properties>
  </extension>
</extensions>
```

帯域幅

インターフェイスの帯域幅を設定および更新できます。帯域幅の値は、メガビット/秒単位です。正の整数である必要があります。

帯域幅を設定および更新するには、ESC から次のコマンドを実行します。

esc_nc_cli --user <username> --password <password> edit-config bandwidth.xml

サンプルの bandwidth.xml は次のとおりです。

```
    <properties>
          <property>
            <name>passthroughMode</name>
            <value>none</value>
          </property>
          <property>
            <name>tagged</name>
            <value>false</value>
          </property>
          <property>
            <name>type</name>
            <value>access</value>
          </property>
          <property>
            <name>bandwidth</name>
            <value>750</value>
          </property>
          <property>
            <name>vlan</name>
            <value>11</value>
          </property>
        </properties>
      </container>
    </containers>
  </extension>
  <extension>
    <name>serial_ports</name>
    <containers>
      <container>
        <name>0</name>
        <properties>
          <property>
            <name>serial_type</name>
            <value>console</value>
          </property>
        </properties>
      </container>
    </containers>
```

```
    </extension>
    <extension>
      <name>image</name>
      <properties>
        <property>
          <name>disk-resize</name>
          <value>true</value>
        </property>
        <property>
          <name>disk_type</name>
          <value>virtio</value>
        </property>
      </properties>
    </extension>
  </extensions>
```

タイプ

プロパティタイプの有効な値は、access と trunk のみです。プロパティタイプを設定および更新するには、ESC から次のコマンドを実行します。

esc_nc_cli --user <username> --password <password> edit-config interfaceType.xml

サンプルの interfaceType.xml は次のとおりです。

```
<extensions>
      <extension>
        <name>interfaces</name>
        <containers>
           <container>
            <name>0</name>
            <properties>
              <property>
                <name>passthroughMode</name>
                <value>none</value>
              </property>
              <property>
                <name>tagged</name>
                <value>false</value>
              </property>
              <property>
                <name>type</name>
                <value>access</value>
              </property>
              <property>
                <name>vlan</name>
                <value>1</value>
              </property>
            </properties>
          </container>
         <container>
            <name>1</name>
            <properties>
              <property>
                <name>passthroughMode</name>
                <value>none</value>
              </property>
              <property>
                <name>tagged</name>
                <value>false</value>
              </property>
              <property>
                <name>type</name>
```

```
            <value>access</value>
          </property>
          <property>
            <name>bandwidth</name>
            <value>750</value>
          </property>
          <property>
            <name>vlan</name>
            <value>11</value>
          </property>
        </properties>
      </container>
    </containers>
  </extension>
  <extension>
    <name>serial_ports</name>
    <containers>
      <container>
        <name>0</name>
        <properties>
          <property>
            <name>serial_type</name>
            <value>console</value>
          </property>
        </properties>
      </container>
    </containers>
  </extension>
  <extension>
    <name>image</name>
    <properties>
      <property>
        <name>disk-resize</name>
        <value>true</value>
      </property>
      <property>
        <name>disk_type</name>
        <value>virtio</value>
      </property>
    </properties>
  </extension>
</extensions>
```

管理ステータス

インターフェイスの `admin_state_up` 属性を使用すると、VNIC を有効または無効にできます。`admin_state_up` の値は True または False に設定できます。True の場合、vNIC は有効です。`admin_state_up` の値が ESC で設定されていない場合、ステータスは CSP 上で **UP** です。admin_state_up の属性を設定および更新するには、ESC から次のコマンドを実行します。

esc_nc_cli --user <username> --password <password> edit-config adminStateUp.xml

サンプルの adminStateUp.xml は次のとおりです。

```
<interfaces>
              <interface>
               <nicid>0</nicid>
               <type>virtual</type>
               <model>virtio</model>
               <network>Eth0-2</network>
               <ip_address>192.168.24.45</ip_address>
               <admin_state_up>true</admin_state_up>
```

```
  </interface>
 <interface>
    <nicid>1</nicid>
    <type>virtual</type>
    <model>virtio</model>
    <network>Eth0-2</network>
    <admin_state_up>false</admin_state_up>
  </interface>
</interfaces>
........
  .......
<extensions>
  <extension>
    <name>interfaces</name>
    <containers>
       <container>
        <name>0</name>
        <properties>
          <property>
            <name>passthroughMode</name>
            <value>none</value>
          </property>
          <property>
            <name>tagged</name>
            <value>false</value>
          </property>
          <property>
            <name>type</name>
            <value>access</value>
          </property>
          <property>
            <name>vlan</name>
            <value>1</value>
          </property>
        </properties>
      </container>
     <container>
        <name>1</name>
        <properties>
          <property>
            <name>passthroughMode</name>
            <value>none</value>
          </property>
          <property>
            <name>tagged</name>
            <value>false</value>
          </property>
          <property>
            <name>type</name>
            <value>access</value>
          </property>
          <property>
            <name>vlan</name>
            <value>11</value>
          </property>
        </properties>
      </container>
    </containers>
  </extension>
  <extension>
    <name>serial_ports</name>
    <containers>
      <container>
        <name>0</name>
```

```
                <properties>
                  <property>
                    <name>serial_type</name>
                    <value>console</value>
                  </property>
                </properties>
              </container>
            </containers>
          </extension>
          <extension>
            <name>image</name>
            <properties>
              <property>
                <name>disk-resize</name>
                <value>true</value>
              </property>
              <property>
                <name>disk_type</name>
                <value>virtio</value>
              </property>
            </properties>
          </extension>
        </extensions>
.....................
```

ネットワーク（Network）

インターフェイスを介してネットワーク属性を設定および更新できます。ネットワークを設定および更新するには、ESC から次のコマンドを実行します。

esc_nc_cli --user <username> --password <password> edit-config NetworkNameChange.xml

サンプルの NetworkNameChange.xml は次のとおりです。

```
<interfaces>
             <interface>
              <nicid>0</nicid>
              <type>virtual</type>
              <model>virtio</model>
              <network>Eth0-2</network>
              <ip_address>192.168.24.45</ip_address>
              <admin_state_up>true</admin_state_up>
            </interface>
           <interface>
              <nicid>1</nicid>
              <type>virtual</type>
              <model>virtio</model>
              <network>Eth0-2</network>
              <admin_state_up>false</admin_state_up>
            </interface>
          </interfaces>
          .........
          ............
          <extensions>
            <extension>
              <name>interfaces</name>
              <containers>
                 <container>
                  <name>0</name>
                  <properties>
                    <property>
                      <name>passthroughMode</name>
                      <value>none</value>
```

```
            </property>
            <property>
              <name>tagged</name>
              <value>false</value>
            </property>
            <property>
              <name>type</name>
              <value>access</value>
            </property>
            <property>
              <name>vlan</name>
              <value>1</value>
            </property>
          </properties>
        </container>
        <container>
          <name>1</name>
          <properties>
            <property>
              <name>passthroughMode</name>
              <value>none</value>
            </property>
            <property>
              <name>tagged</name>
              <value>false</value>
            </property>
            <property>
              <name>type</name>
              <value>access</value>
            </property>
            <property>
              <name>vlan</name>
              <value>11</value>
            </property>
          </properties>
        </container>
      </containers>
    </extension>
    <extension>
      <name>serial_ports</name>
      <containers>
        <container>
          <name>0</name>
          <properties>
            <property>
              <name>serial_type</name>
              <value>console</value>
            </property>
          </properties>
        </container>
      </containers>
    </extension>
    <extension>
```

静的 IP プールの追加

既存の展開に新しい静的 IP プールを追加できます。

静的 IP プールを追加する NETCONF 要求：

```
<scaling>
<min_active>2</min_active>
<max_active>5</max_active>
<elastic>true</elastic>
```

```
<static_ip_address_pool>
<network>IP-pool-network-A</network>
<ip_address_range>
<start>172.16.5.13</start>
<end>172.16.5.13</end>
</ip_address_range>
</static_ip_address_pool>
<static_ip_address_pool>
<network>IP-pool-network-B</network>
<ip_address_range>
<start>172.16.7.13</start>
<end>172.16.7.13</end>
</ip_address_range>
</static_ip_address_pool>
</scaling>
```

静的 IP プールの削除

実行中の展開で既存の IP プールを削除できます。

静的 IP プールを削除する NETCONF 要求：

```
<scaling>
<min_active>2</min_active>
<max_active>5</max_active>
<elastic>true</elastic>
<static_ip_address_pool>
<network>IP-pool-network-A</network>
<ip_address_range>
<start>172.16.5.13</start>
<end>172.16.5.13</end>
</ip_address_range>
</static_ip_address_pool>
<static_ip_address_pool nc:operation="delete">
<network>IP-pool-network-B</network>
<ip_address_range>
<start>172.16.7.13</start>
<end>172.16.7.13</end>
</ip_address_range>
</static_ip_address_pool>
</scaling>
```

（注）
- 既存の展開では、すでに存在する静的 IP プールを更新することはできません。新しい静的 IP プールを追加するか、静的 IP プールが使用中でない場合は削除することができます。
- インターフェイスの IP アドレスは更新できません。つまり、1 つの IP アドレスで展開してから同じ NIC ID に新しい IP を追加することはできません。

静的 IP プール、インターフェイス、およびネットワーク内の依存関係により、次のシナリオがサポートまたは拒否されます。

要求	サポート対象または拒否
単一または異なる要求で新しい静的 IP プールを追加または削除します。	サポート対象

要求	サポート対象または拒否
静的 IP を備えたインターフェイスを追加します。	サポート対象
同じ要求にインターフェイスと対応する IP プールを追加します。	サポート対象
インターフェイスを削除し、対応する IP プールを保持します。	サポート対象
同じ要求でインターフェイスと対応する IP プールを削除します。	サポート対象
IP の 1 つが VM のインターフェイスで使用されている場合、IP プールを削除します。	拒否
ネットワークと、異なるネットワークを持つ静的 IP プールを単独の要求に追加します。	サポート対象
既存のネットワークに、同じ更新の対応するインターフェイスと IP プールを追加します。	サポート対象
更新に新しいネットワークを追加し、次の更新に対応する新しい IP プールを追加します。	サポート対象
対応するネットワークなしで IP プールを追加します。	拒否
いずれのインターフェイスでも IP が使用されていない場合は、同じ要求内のネットワークと参照元 IP プールを削除します。	サポート対象
IP プールとインターフェイスで使用されているネットワークを削除します。	拒否
既存のネットワークに、同じ更新のインターフェイスと IP プールを追加します。	サポート対象
サブネットを持つネットワークが存在しているにもかかわらず、インターフェイスで使用されている IP がない IP プールを削除します。	サポート対象
すでに存在する IP プールを追加します。	要求は NETCONF によって受け入れられますが、アクションは実行されません
既存の IP プールの IP アドレスを更新します。	拒否

VM グループのデイゼロ設定の更新

既存の展開でVMグループのデイゼロ設定を更新（追加、削除、または変更）するには、展開を編集して、config_dataで設定を更新します。新しいデイゼロ設定ファイルは、将来の展開にのみ適用されます。これは、VMリカバリ（展開解除/展開）またはスケールアウトによってトリガーされます。

（注） 既存のデイゼロ設定ファイルを変更するには、URLまたはパスを指定する必要があります。これにより、ESC は設定で発生した変更を検出できます。

次の例では、VMALIVEイベントが受信されない場合、自動回復のトリガーからイベントの単純なロギングにアクションを変更できます。

既存の設定：

```
<config_data>
  <configuration>
    <dst>WSA_config.txt</dst>

<file>https://172.16.73.167:4343/day0/cfg/vWSA/node/001-wsa/provider/Symphony_VNF_P-1B/file>

  </configuration>
  <configuration>
    <dst>license.txt</dst>

<file>https://172.16.73.167:4343/day0/cfg/vWSA/node/001-wsa/provider/Symphony_VNF_P-1B/wsa-license.txt</file>

  </configuration>
</config_data>
```

新しい設定：

```
<config_data>
  <configuration>
    <dst>WSA_config.txt</dst>

<file>https://172.16.73.167:4343/day0/cfg/vWSA/node/001-wsa/provider/Symphony_VNF_P-1B/file>

  </configuration>
  <configuration>
    <dst>license.txt</dst>

<file>https://172.16.73.167:4343/day0/cfg/vWSA/node/002-wsa/provider/Symphony_VNF_P-1B/wsa-license.txt</file>

  </configuration>
</config_data>
```

SERVICE_UPDATED 通知は、設定の更新後に送信されます。

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-05-05T00:35:15.359+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Service group update completed successfully</status_message>
    <depname>900cd7554d31-5454000964474c1cbc07256792e63240-cloudvpn</depname>
    <tenant>Symphony_VNF_P-1B</tenant>
    <tenant_id>3098b55808e84484a4f8bab2160a41a7</tenant_id>
    <depid>b7d566ce-1ee6-4147-8c23-c8bcb5d05fd4</depid>
```

```
    <vm_source/>
    <vm_target/>
    <event>
      <type>SERVICE_UPDATED</type>
    </event>
  </escEvent>
</notification>
```

デイゼロ設定の詳細については、デイゼロ設定 （173 ページ）を参照してください。

KPI とルールの更新

ESC では、既存の展開で VM の KPI とルールを更新できます。データモデルを編集して、KPI とルールのセクションを更新します。

たとえば、既存の展開でポーリング頻度を変更するには、データモデルの KPI セクションで *<poll_frequency>* 要素を更新します。

次のサンプルで、*<poll_frequency>3</poll_frequency>* を *<poll_frequency>20</poll_frequency>* に変更します。

```
      <kpi>
                  <event_name>VM_ALIVE</event_name>
                  <metric_value>1</metric_value>
                  <metric_cond>GT</metric_cond>
                  <metric_type>UINT32</metric_type>
                  <metric_collector>
                    <type>ICMPPing</type>
                    <nicid>0</nicid>
                    <poll_frequency>3</poll_frequency>
                    <polling_unit>seconds</polling_unit>
                    <continuous_alarm>false</continuous_alarm>
                  </metric_collector>
                </kpi>
```

同様に、VM の既存のルールを更新できます。たとえば、ブート障害時に自動リカバリをオフにし、アクションをログに記録するには、次のサンプルで *<action>FALSE recover autohealing</action>* を *<action>FALSE log</action>* に更新します。

```
   <rules>
               <admin_rules>
                 <rule>
                   <event_name>VM_ALIVE</event_name>
                   <action>ALWAYS log</action>
                   <action>FALSE recover autohealing</action>
                   <action>TRUE servicebooted.sh</action>
                 </rule>
                ...
     ...
             </rules>
```

（注）
- KPI またはルールの更新中は、モニタが設定解除されるため、自動回復は行われません。自動回復は、展開でモニタがリセットされると発生します。
- *event_name* は更新中に変更できません。追加または削除のみ可能です。

KPI とルールの詳細については、「KPIとルール」のセクションを参照してください。

展開内の VM 数の更新（手動スケールイン/スケールアウトの更新）

既存の展開から VM を追加および削除するには、データモデルのスケーリングセクションで min_active および max_active の値を変更します。これにより、初期展開のサイズが変更されます。

次の例では、導入の初期カウントは 2 VM で、5 VM にスケールアウトできます。

```
 <esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
  <version>1.0.0</version>
      . . .
     <vm_group>
        </interfaces>
          <interface>
            <network>1fbf9fc2-3074-4ae6-bb0a-09d526fbada6</network>
            <nicid>1</nicid>
            <ip_address>10.0.0.0</ip_address>
          </interface>
        </interfaces>
       <scaling>
          <min_active>2</min_active>
          <max_active>5</max_active>
          <elastic>true</elastic>

            . . .
```

次の例では、追加で 8 つの VM を作成し、アクティブな VM の数を 10 以上にします。その他のシナリオについては、次の表を参照してください。

```
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
  <version>1.0.0</version>
      . . .
     <vm_group>
        </interfaces>
          <interface>
            <network>1fbf9fc2-3074-4ae6-bb0a-09d526fbada6</network>
            <nicid>1</nicid>
            <ip_address>10.0.0.0</ip_address>
          </interface>
        </interfaces>
       <scaling>
          <min_active>10</min_active>
          <max_active>15</max_active>
          <elastic>true</elastic>
          <static_ip_address_pool>
            <network>1fbf9fc2-3074-4ae6-bb0a-09d526fbada6</network>
            <gateway>192.168.0.1</gateway> <!-- not used -->
            <netmask>255.255.255.0</netmask> <!-- not used -->
            <ip_address>10.0.0.0</ip_address>
          </static_ip_address_pool>
        </scaling>
```

次の表に、スケーリングセクションで最小値と最大値を更新するシナリオを示します。

表 *16* : 展開内の *VM* 数の更新

シナリオ	古い値	新しい値	アクティブ値
スケーリングセクションでVMの初期最小数が2、最大数が5の場合、VMの最小数を3に更新すると、追加のVMが1つ作成されます。これは、アクティブなVMの数が2のままであることを前提としています。	以前のVMの最小数は2です。	新しいVMの最小数は3です。	アクティブなVMの数は2です。
VMの数の初期最小値が2で最大値が5の場合、最小値を3に更新するとデータベースは更新されますが、展開には影響しません。このシナリオは、1つの追加VMを作成して元の展開が拡張された場合に発生します。	以前の最小値は2です。	新しい最小値は3です。	アクティブな数は3です。
VMの初期最小数が2、最大数が5の場合、最小値を1に更新するとデータベースは更新されますが、展開には影響しません。アクティブなVMの数が最小または最大値の範囲内にあるため、アクティブなVMの数が最小値よりも大きくても展開は有効です。	以前の最小値は2です。	新しい最小値は1です。	アクティブなVMの数は2です。

シナリオ	古い値	新しい値	アクティブ値
VM の初期最小数が 2、最大数が 5 の場合、最大値を 6 に更新するとデータベースは更新されますが、展開には影響しません。アクティブな VM の数が最小または最大値の範囲内にあるため、アクティブな VM の数が最大値よりも少なくても展開は有効です。	以前の最大値は 5 です。	新しい最大値は 6 です。	アクティブな VM の数は 2 です。
VM の初期最小数が 2、最大数が 5 の場合、最大値を 4 に更新するとデータベースは更新されますが、展開には影響しません。アクティブな VM の数が最小または最大値の範囲内にあるため、アクティブな VM の数が最大値よりも少なくても展開は有効です。	以前の最大値は 5 です。	新しい最大値は 4 です。	アクティブな VM の数は 2 です。
VM の初期最小数が 2、最大数が 5 の場合、VM の最大数を 4 に更新するとデータベースは更新され、展開から 1 つの VM が削除されます。最後に作成された VM が削除され、アクティブな最大数が 4 になります。	以前の最大値は 5 です。	新しい最大値は 4 です。	アクティブな VM の数は 4 です。

静的 IP が使用されている場合は、VM を展開に追加するには、プールの拡張セクションを更新する必要があります。

次に、展開データモデルを示します。

```
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
```

```
<version>1.0.0</version>
    . . .
   <vm_group>
      </interfaces>
        <interface>
          <network>1fbf9fc2-3074-4ae6-bb0a-09d526fbada6</network>
          <nicid>1</nicid>
          <ip_address>23.23.23.23</ip_address>
        </interface>
      </interfaces>
     <scaling>
        <min_active>1</min_active>
        <max_active>1</max_active>
        <elastic>true</elastic>
        <static_ip_address_pool>
          <network>1fbf9fc2-3074-4ae6-bb0a-09d526fbada6</network>
          <gateway>192.168.0.1</gateway> <!- not used ->
          <netmask>255.255.255.0</netmask> <!- not used ->
          <ip_address>23.23.23.23</ip_address>
        </static_ip_address_pool>
      </scaling>
```

プールは、ネットワーク ID を介してインターフェイスにリンクされます。更新されたデータモデルは次のとおりです。

```
Update payload
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
  <version>1.0.0</version>
    . . .
    <vm_group>
       <interfaces>
         <interface>
           <network>1fbf9fc2-3074-4ae6-bb0a-09d526fbada6</network>
           <nicid>1</nicid>
           <ip_address>23.23.23.23</ip_address>
         </interface>
       </interfaces>
       <scaling>
         <min_active>2</min_active>
         <max_active>2</max_active>
         <elastic>true</elastic>
         <static_ip_address_pool>
             <network>1fbf9fc2-3074-4ae6-bb0a-09d526fbada6</network>
             <gateway>192.168.0.1</gateway>
             <netmask>255.255.255.0</netmask>
             <ip_address>10.0.0.0</ip_address>
             <ip_address>10.0.0.24</ip_address>
         </static_ip_address_pool>
       </scaling>
```

最初の IP も更新データモデルに含まれています。値が更新リストにない場合は、プールから削除されます。これにより、IP アドレス 10.0.0.24 を使用する単一の VM が作成されます。

（注） 展開から特定の VM を削除することはできません。

リカバリ待機時間の更新

既存の展開でリカバリ待機時間を更新できるようになりました。次の例では、<recovery_wait_time> パラメータは、初期展開時に 60 秒に設定されます。

```
<vm_group>
 <name>CSR</name>
 <recovery_wait_time>60</recovery_wait_time>
```

既存の展開では、リカバリ待機時間が 100 秒に更新されます。

```
<vm_group>
 <name>CSR</name>
 <recovery_wait_time>100</recovery_wait_time>
```

リカバリ待機時間を更新すると、既存の展開で作成された VM に影響します。

VM_DOWN イベントを受信した後、リカバリ待機時間により、ESC は VM リカバリワークフローに進む前に一定時間待機できます。リカバリ待機時間に割り当てられた時間により、VM はネットワーク接続を復元したり、自身を修復したりできます。この時間内に VM_ALIVE がトリガーされると、VM リカバリはキャンセルされます。

リカバリポリシーの更新

展開の更新中に、リカバリポリシーを追加したり、既存のリカバリポリシーパラメータを更新したりできます。

自動リカバリは、通知なしで自動的にトリガーされます。手動リカバリの場合、VM_MANUAL_RECOVERY_NEEDED 通知が送信され、ユーザがコマンドを送信した場合にのみリカバリが開始されます。

リカバリタイプが自動に設定されている場合、リカバリは通知なしで自動的に開始されます。リカバリタイプを手動に設定すると、VM_MANUAL_RECOVERY_NEEDED 通知が送信され、ユーザがコマンドを送信した場合にのみリカバリが開始されます。

次の例では、初期展開時にリカバリアクションが REBOOT_THEN_REDEPLOY に設定されます。展開の更新中に REBOOT_ONLY に更新されます。リカバリが成功しない場合、最初の展開での最大再試行回数は 1 です。既存の展開でも、最大再試行回数を更新できます。次の例では、最大再試行回数が 3 に更新されます。

初期展開

```
<recovery_policy>
           <action_on_recovery>REBOOT_THEN_REDEPLOY</action_on_recovery>
              <max_retries>1</max_retries>
         </recovery_policy>
```

展開の更新

```
<recovery_policy>
           <action_on_recovery>REBOOT_ONLY</action_on_recovery>
            <max_retries>3</max_retries>
        </recovery_policy>
```

リカバリポリシー通知は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2017-06-21T12:35:12.354+00:00</eventTime>
```

```
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Service group update completed successfully</status_message>
    <depname>jenkins-update-recovery-success-dep-201102</depname>
    <tenant>jenkins-update-recovery-success-tenant-201102</tenant>
    <tenant_id>11ade63bac8a4010a969df0d0b91b9bf</tenant_id>
    <depid>574b2e11-61a9-4d9b-83b1-e95a3aa56fdd</depid>
    <event>
      <type>SERVICE_UPDATED</type>
    </event>
  </escEvent>
</notification>
```

展開の更新中は、リカバリポリシーを LCS で上書きすることはできません。たとえば、REBOOT_ONLY を使用したリカバリポリシーは、ライフサイクルステージ（LCS）で上書きできません。

イメージの更新

既存の展開内で VM のイメージ参照を更新できます。

次に、データモデルの更新を示します。

既存のデータモデル：

```
<recovery_wait_time>30</recovery_wait_time>
<flavor>Automation-Cirros-Flavor</flavor>
<image>Automation-Cirros-Image</image>
```

新しいデータモデル：

```
<recovery_wait_time>30</recovery_wait_time>
<flavor>Automation-Cirros-Flavor</flavor>
<image>Automation-CSR-Image-3_14</image>
```

イメージが更新された後、サービス更新通知が送信されます。

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
<eventTime>2018-05-10T17:34:00.605+00:00</eventTime>
<escEvent xmlns="http://www.cisco.com/esc/esc">
<status>SUCCESS</status>
<status_code>200</status_code>
<status_message>Service group update completed successfully</status_message>
<depname>ud-A</depname>
<tenant>ut-AM</tenant>
<tenant_id>24e21e581ad441ebbb3bd22e69c36322</tenant_id>
<depid>e009b1cc-0aa9-4abd-8aac-265be7f9a80d</depid>
<event>
<type>SERVICE_UPDATED</type>
</event>
</escEvent>
</notification>
```

新しいイメージ参照が opdata に表示されます。

```
<vm_group>
```

```
<name>ug-1</name>
<flavor>m1.large</flavor>
<image>cirror</image>
<vm_instance>
<vm_id>9a63afed-c70f-4827-91e2-72bdd86c5e39</vm_id>
```

誤ったイメージ名を指定すると、次のエラーが表示されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
<eventTime>2018-05-08T19:28:12.321+00:00</eventTime>
<escEvent xmlns="http://www.cisco.com/esc/esc">
<status>FAILURE</status>
<status_code>500</status_code>
<status_message>Error during service update: Failed to [Update] deployment: The image
Automation-1-Cirros-Image cannot be found on the virtual infrastructure
manager.</status_message>
<depname>ud-A</depname>
<tenant>ut-AL</tenant>
<tenant_id>4fb19d82c5b34b33aa6162c0b33f07d7</tenant_id>
<depid>6eed6eba-4f3f-401d-83be-91d703ee4946</depid>
<event>
<type>SERVICE_UPDATED</type>
</event>
</escEvent>
</notification>
```

イメージ更新のロールバックシナリオ

イメージ参照が後続の更新で更新されるように、サービスがエラー状態にある場合でも、イメージ参照を更新する必要があります。次の表に、イメージ更新のロールバック条件、予想される動作、および通知を示します。

ロールバック条件	予想される動作	通知
サービスはERROR状態であり、要求にはイメージの更新のみが含まれます	イメージは更新されますが、サービスはERROR状態のままです	?xml version="1.0" encoding="UTF-8"?> <notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"> <eventTime>2018-06-06T13:59:04.331+00:00</eventTime> <escEvent xmlns="http://www.cisco.com/esc/esc"> <status>SUCCESS</status> <status_code>200</status_code> <status_message>Deployment update successful. But one or more VMs are still in ERROR state.</status_message> <depname>ud-A</depname> <tenant>ut-JJ</tenant> <tenant_id>0dbb67d6457642f68520565ce785976a</tenant_id> <depid>0feea6bc-310c-49c8-8416-94f89a324bfb</depid> <event> <type>SERVICE_UPDATED</type> </event> </escEvent> </notification>

ロールバック条件	予想される動作	通知
サービスはERROR状態で、VMグループを削除する要求が送信されます（エラー）	VMグループが削除され、サービスがACTIVE状態です	
サービスがERROR状態です（エラー状態の）VMグループを削除する要求が、同じVMグループ内のイメージ更新要求とともに送信されます。	VMグループを削除する必要があります。イメージの更新による影響はありません。サービスはACTIVE状態に戻ります。	
サービスがERROR状態です。（アクティブな）VMグループを削除する要求が、（エラー状態の）別のVMグループのイメージ更新要求とともに送信されます。	（アクティブな）VMグループが削除されます。（エラー状態の）VMグループでイメージが更新されます。サービスはERROR状態のままです。	?xml version="1.0" encoding="UTF-8"?> <notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"> <eventTime>2018-06-06T13:59:04.331+00:00</eventTime> <escEvent xmlns="http://www.cisco.com/esc/esc"> <status>SUCCESS</status> <status_code>200</status_code> <status_message>Deployment update successful. But one or more VMs are still in ERROR state.</status_message> <depname>ud-A</depname> <tenant>ut-JJ</tenant> <tenant_id>0dbb67d6457642f68520565ce785976a</tenant_id> <depid>0feea6bc-310c-49c8-8416-94f89a324bfb</depid> <event> <type>SERVICE_UPDATED</type> </event> </escEvent> </notification>

ロールバック条件	予想される動作	通知
サービスは ERROR 状態です。単一の VM グループが存在します（エラー状態）。イメージ更新要求が送信されます。	イメージは更新されますが、サービスは ERROR 状態のままです。（エラー状態の）VM グループはサービス内の唯一のグループであるため、削除できません。ユーザは展開解除と再展開が必要です。	

VM グループ（vCloud Director）の追加

ESC は、vCD での VM グループの追加と削除のみをサポートします。サービスのアップデートでは、1 つまたは複数の VM グループを追加または削除できます。

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"
xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications">
  <tenants>
    <tenant>
      <!-- ESC scope tenant -->
      <name>vnf-dep</name>
      <vim_mapping>false</vim_mapping>
      <deployments>
        <deployment>
          <!-- vApp instance name -->
          <name>dep</name>
          <policies>
            <placement_group>
```

```
    <name>placement-affinity-1</name>
    <type>affinity</type>
    <enforcement>strict</enforcement>
    <vm_group>g1</vm_group>
    <vm_group>g2</vm_group>
    <vm_group>g3</vm_group>
  </placement_group>
</policies>
<extensions>
  <extension>
    <name>VMWARE_VCD_PARAMS</name>
    <properties>
      <property>
        <name>CATALOG_NAME</name>
        <value>catalog-1</value>
      </property>
      <property>
        <name>VAPP_TEMPLATE_NAME</name>
        <value>uLinux_vApp_Template</value>
      </property>
    </properties>
  </extension>
</extensions>
<vm_group>
  <name>g1</name>
  <locator>
    <!-- vCD vim connector id -->
    <vim_id>vcd</vim_id>
    <!-- vCD orgnization -->
    <vim_project>esc</vim_project>
    <!-- vDC name -->
    <vim_vdc>VDC-1</vim_vdc>
  </locator>
  <!-- VM name in vAppTemplate -->
  <image>vm-001</image>
  <bootup_time>120</bootup_time>
  <recovery_wait_time>5</recovery_wait_time>
  <interfaces>
    <interface>
      <nicid>0</nicid>
      <network>MgtNetwork</network>
      <ip_address>10.0.0.155</ip_address>
      <mac_address>00:1C:B3:09:85:15</mac_address>
    </interface>
  </interfaces>
  <scaling>
    <min_active>1</min_active>
    <max_active>1</max_active>
    <elastic>true</elastic>
    <static_ip_address_pool>
      <network>MgtNetwork</network>
      <ip_address>10.0.0.155</ip_address>
    </static_ip_address_pool>
    <static_mac_address_pool>
      <network>MgtNetwork</network>
      <mac_address>00:1C:B3:09:85:15</mac_address>
    </static_mac_address_pool>
  </scaling>
  <kpi_data>
    <kpi>
      <event_name>VM_ALIVE</event_name>
      <metric_value>1</metric_value>
      <metric_cond>GT</metric_cond>
      <metric_type>UINT32</metric_type>
```

```
        <metric_collector>
          <type>ICMPPing</type>
          <nicid>0</nicid>
          <poll_frequency>30</poll_frequency>
          <polling_unit>seconds</polling_unit>
          <continuous_alarm>false</continuous_alarm>
        </metric_collector>
      </kpi>
    </kpi_data>
    <rules>
      <admin_rules>
        <rule>
          <event_name>VM_ALIVE</event_name>
          <action>"ALWAYS log"</action>
          <action>"TRUE servicebooted.sh"</action>
          <action>"FALSE recover autohealing"</action>
        </rule>
      </admin_rules>
    </rules>
    <config_data>
      <configuration>
        <dst>ovfProperty:mgmt-ipv4-addr</dst>
        <data>$NICID_0_IP_ADDRESS/24</data>
      </configuration>
    </config_data>
    <recovery_policy>
      <action_on_recovery>REBOOT_ONLY</action_on_recovery>
    </recovery_policy>
  </vm_group>
  <vm_group>
    <name>g2</name>
    <locator>
      <!-- vCD vim connector id -->
      <vim_id>vcd</vim_id>
      <!-- vCD orgnization -->
      <vim_project>esc</vim_project>
      <!-- vDC name -->
      <vim_vdc>VDC-1</vim_vdc>
    </locator>
    <!-- VM name in vAppTemplate -->
    <image>vm-002</image>
    <bootup_time>120</bootup_time>
    <recovery_wait_time>5</recovery_wait_time>
    <interfaces>
      <interface>
        <nicid>0</nicid>
        <network>MgtNetwork</network>
        <ip_address>10.0.0.156</ip_address>
        <mac_address>00:1C:B3:09:85:16</mac_address>
      </interface>
    </interfaces>
    <scaling>
      <min_active>1</min_active>
      <max_active>1</max_active>
      <elastic>true</elastic>
      <static_ip_address_pool>
        <network>MgtNetwork</network>
        <ip_address>10.0.0.156</ip_address>
      </static_ip_address_pool>
      <static_mac_address_pool>
        <network>MgtNetwork</network>
        <mac_address>00:1C:B3:09:85:16</mac_address>
      </static_mac_address_pool>
    </scaling>
```

```
  <kpi_data>
    <kpi>
      <event_name>VM_ALIVE</event_name>
      <metric_value>1</metric_value>
      <metric_cond>GT</metric_cond>
      <metric_type>UINT32</metric_type>
      <metric_collector>
        <type>ICMPPing</type>
        <nicid>0</nicid>
        <poll_frequency>30</poll_frequency>
        <polling_unit>seconds</polling_unit>
        <continuous_alarm>false</continuous_alarm>
      </metric_collector>
    </kpi>
  </kpi_data>
  <rules>
    <admin_rules>
      <rule>
        <event_name>VM_ALIVE</event_name>
        <action>"ALWAYS log"</action>
        <action>"TRUE servicebooted.sh"</action>
        <action>"FALSE recover autohealing"</action>
      </rule>
    </admin_rules>
  </rules>
  <config_data>
    <configuration>
      <dst>ovfProperty:mgmt-ipv4-addr</dst>
      <data>$NICID_0_IP_ADDRESS/24</data>
    </configuration>
  </config_data>
  <recovery_policy>
    <action_on_recovery>REBOOT_ONLY</action_on_recovery>
  </recovery_policy>
</vm_group>
<vm_group>
  <name>g3</name>
  <locator>
    <!-- vCD vim connector id -->
    <vim_id>vcd</vim_id>
    <!-- vCD orgnization -->
    <vim_project>esc</vim_project>
    <!-- vDC name -->
    <vim_vdc>VDC-1</vim_vdc>
  </locator>
  <!-- VM name in vAppTemplate -->
  <image>vm-002</image>
  <bootup_time>120</bootup_time>
  <recovery_wait_time>5</recovery_wait_time>
  <interfaces>
    <interface>
      <nicid>0</nicid>
      <network>MgtNetwork</network>
      <ip_address>20.0.0.157</ip_address>
      <mac_address>00:1C:B3:09:85:17</mac_address>
    </interface>
  </interfaces>
  <scaling>
    <min_active>1</min_active>
    <max_active>1</max_active>
    <elastic>true</elastic>
    <static_ip_address_pool>
      <network>MgtNetwork</network>
      <ip_address>10.0.0.157</ip_address>
```

```
            </static_ip_address_pool>
            <static_mac_address_pool>
              <network>MgtNetwork</network>
              <mac_address>00:1C:B3:09:85:17</mac_address>
            </static_mac_address_pool>
          </scaling>
          <kpi_data>
            <kpi>
              <event_name>VM_ALIVE</event_name>
              <metric_value>1</metric_value>
              <metric_cond>GT</metric_cond>
              <metric_type>UINT32</metric_type>
              <metric_collector>
                <type>ICMPPing</type>
                <nicid>0</nicid>
                <poll_frequency>30</poll_frequency>
                <polling_unit>seconds</polling_unit>
                <continuous_alarm>false</continuous_alarm>
              </metric_collector>
            </kpi>
          </kpi_data>
          <rules>
            <admin_rules>
              <rule>
                <event_name>VM_ALIVE</event_name>
                <action>"ALWAYS log"</action>
                <action>"TRUE servicebooted.sh"</action>
                <action>"FALSE recover autohealing"</action>
              </rule>
            </admin_rules>
          </rules>
          <config_data>
            <configuration>
              <dst>ovfProperty:mgmt-ipv4-addr</dst>
              <data>$NICID_0_IP_ADDRESS/24</data>
            </configuration>
          </config_data>
          <recovery_policy>
            <action_on_recovery>REBOOT_ONLY</action_on_recovery>
          </recovery_policy>
        </vm_group>
      </deployment>
    </deployments>
  </tenant>
 </tenants>
</esc_datamodel>
```

VM グループの削除（vCloud Director）

ESC では、vCloud Director で VM グループを削除できます。

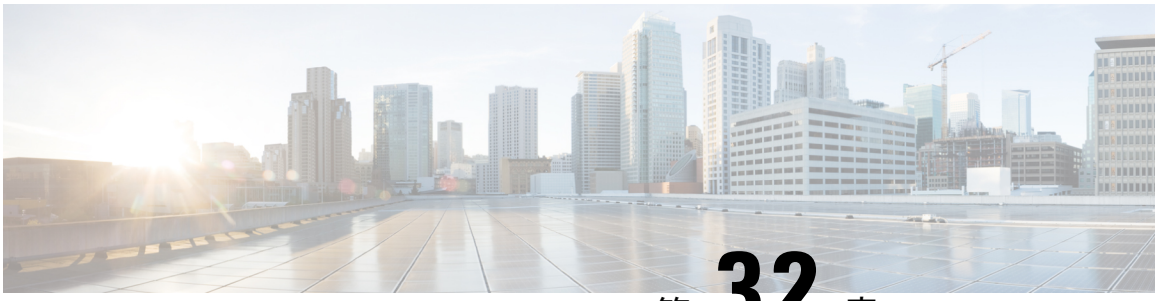
```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc" xmlns:nc="http://www.cisco.com/esc/esc"
 xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications">
  <tenants>
    <tenant>
      <!-- ESC scope tenant -->
      <name>vnf-dep</name>
      <vim_mapping>false</vim_mapping>
      <deployments>
        <deployment>
```

```
<!-- vApp instance name -->
<name>dep</name>
<policies>
  <placement_group>
    <name>placement-affinity-1</name>
    <type>affinity</type>
    <enforcement>strict</enforcement>
    <vm_group>g1</vm_group>
    <vm_group>g2</vm_group>
    <vm_group nc:operation="delete">g3</vm_group>
  </placement_group>
</policies>
<extensions>
  <extension>
    <name>VMWARE_VCD_PARAMS</name>
    <properties>
      <property>
        <name>CATALOG_NAME</name>
        <value>catalog-1</value>
      </property>
      <property>
        <name>VAPP_TEMPLATE_NAME</name>
        <value>uLinux_vApp_Template</value>
      </property>
    </properties>
  </extension>
</extensions>
<vm_group>
  <name>g1</name>
  <locator>
    <!-- vCD vim connector id -->
    <vim_id>vcd</vim_id>
    <!-- vCD orgnization -->
    <vim_project>esc</vim_project>
    <!-- vDC name -->
    <vim_vdc>VDC-1</vim_vdc>
  </locator>
  <!-- VM name in vAppTemplate -->
  <image>vm-001</image>
  <bootup_time>120</bootup_time>
  <recovery_wait_time>5</recovery_wait_time>
  <interfaces>
    <interface>
      <nicid>0</nicid>
      <network>MgtNetwork</network>
      <ip_address>10.0.0.155</ip_address>
      <mac_address>00:1C:B3:09:85:15</mac_address>
    </interface>
  </interfaces>
  <scaling>
    <min_active>1</min_active>
    <max_active>1</max_active>
    <elastic>true</elastic>
    <static_ip_address_pool>
      <network>MgtNetwork</network>
      <ip_address>10.0.0.155</ip_address>
    </static_ip_address_pool>
    <static_mac_address_pool>
      <network>MgtNetwork</network>
      <mac_address>00:1C:B3:09:85:15</mac_address>
    </static_mac_address_pool>
  </scaling>
  <kpi_data>
    <kpi>
```

```
        <event_name>VM_ALIVE</event_name>
        <metric_value>1</metric_value>
        <metric_cond>GT</metric_cond>
        <metric_type>UINT32</metric_type>
        <metric_collector>
          <type>ICMPPing</type>
          <nicid>0</nicid>
          <poll_frequency>30</poll_frequency>
          <polling_unit>seconds</polling_unit>
          <continuous_alarm>false</continuous_alarm>
        </metric_collector>
      </kpi>
    </kpi_data>
    <rules>
      <admin_rules>
        <rule>
          <event_name>VM_ALIVE</event_name>
          <action>"ALWAYS log"</action>
          <action>"TRUE servicebooted.sh"</action>
          <action>"FALSE recover autohealing"</action>
        </rule>
      </admin_rules>
    </rules>
    <config_data>
      <configuration>
        <dst>ovfProperty:mgmt-ipv4-addr</dst>
        <data>$NICID_0_IP_ADDRESS/24</data>
      </configuration>
    </config_data>
    <recovery_policy>
      <action_on_recovery>REBOOT_ONLY</action_on_recovery>
    </recovery_policy>
  </vm_group>
  <vm_group>
    <name>g2</name>
    <locator>
      <!-- vCD vim connector id -->
      <vim_id>vcd</vim_id>
      <!-- vCD orgnization -->
      <vim_project>esc</vim_project>
      <!-- vDC name -->
      <vim_vdc>VDC-1</vim_vdc>
    </locator>
    <!-- VM name in vAppTemplate -->
    <image>vm-002</image>
    <bootup_time>120</bootup_time>
    <recovery_wait_time>5</recovery_wait_time>
    <interfaces>
      <interface>
        <nicid>0</nicid>
        <network>MgtNetwork</network>
        <ip_address>10.0.0.156</ip_address>
        <mac_address>00:1C:B3:09:85:16</mac_address>
      </interface>
    </interfaces>
    <scaling>
      <min_active>1</min_active>
      <max_active>1</max_active>
      <elastic>true</elastic>
      <static_ip_address_pool>
        <network>MgtNetwork</network>
        <ip_address>10.0.0.156</ip_address>
      </static_ip_address_pool>
      <static_mac_address_pool>
```

```
        <network>MgtNetwork</network>
        <mac_address>00:1C:B3:09:85:16</mac_address>
      </static_mac_address_pool>
    </scaling>
    <kpi_data>
      <kpi>
        <event_name>VM_ALIVE</event_name>
        <metric_value>1</metric_value>
        <metric_cond>GT</metric_cond>
        <metric_type>UINT32</metric_type>
        <metric_collector>
          <type>ICMPPing</type>
          <nicid>0</nicid>
          <poll_frequency>30</poll_frequency>
          <polling_unit>seconds</polling_unit>
          <continuous_alarm>false</continuous_alarm>
        </metric_collector>
      </kpi>
    </kpi_data>
    <rules>
      <admin_rules>
        <rule>
          <event_name>VM_ALIVE</event_name>
          <action>"ALWAYS log"</action>
          <action>"TRUE servicebooted.sh"</action>
          <action>"FALSE recover autohealing"</action>
        </rule>
      </admin_rules>
    </rules>
    <config_data>
      <configuration>
        <dst>ovfProperty:mgmt-ipv4-addr</dst>
        <data>$NICID_0_IP_ADDRESS/24</data>
      </configuration>
    </config_data>
    <recovery_policy>
      <action_on_recovery>REBOOT_ONLY</action_on_recovery>
    </recovery_policy>
  </vm_group>
  <vm_group nc:operation="delete">
    <name>g3</name>
    <locator>
      <!-- vCD vim connector id -->
      <vim_id>vcd</vim_id>
      <!-- vCD orgnization -->
      <vim_project>esc</vim_project>
      <!-- vDC name -->
      <vim_vdc>VDC-1</vim_vdc>
    </locator>
    <!-- VM name in vAppTemplate -->
    <image>vm-002</image>
    <bootup_time>120</bootup_time>
    <recovery_wait_time>5</recovery_wait_time>
    <interfaces>
      <interface>
        <nicid>0</nicid>
        <network>MgtNetwork</network>
        <ip_address>10.0.0.157</ip_address>
        <mac_address>00:1C:B3:09:85:17</mac_address>
      </interface>
    </interfaces>
    <scaling>
      <min_active>1</min_active>
      <max_active>1</max_active>
```

```
            <elastic>true</elastic>
            <static_ip_address_pool>
              <network>MgtNetwork</network>
              <ip_address>10.0.0.157</ip_address>
            </static_ip_address_pool>
            <static_mac_address_pool>
              <network>MgtNetwork</network>
              <mac_address>00:1C:B3:09:85:17</mac_address>
            </static_mac_address_pool>
          </scaling>
          <kpi_data>
            <kpi>
              <event_name>VM_ALIVE</event_name>
              <metric_value>1</metric_value>
              <metric_cond>GT</metric_cond>
              <metric_type>UINT32</metric_type>
              <metric_collector>
                <type>ICMPPing</type>
                <nicid>0</nicid>
                <poll_frequency>30</poll_frequency>
                <polling_unit>seconds</polling_unit>
                <continuous_alarm>false</continuous_alarm>
              </metric_collector>
            </kpi>
          </kpi_data>
          <rules>
            <admin_rules>
              <rule>
                <event_name>VM_ALIVE</event_name>
                <action>"ALWAYS log"</action>
                <action>"TRUE servicebooted.sh"</action>
                <action>"FALSE recover autohealing"</action>
              </rule>
            </admin_rules>
          </rules>
          <config_data>
            <configuration>
              <dst>ovfProperty:mgmt-ipv4-addr</dst>
              <data>$NICID_0_IP_ADDRESS/24</data>
            </configuration>
          </config_data>
          <recovery_policy>
            <action_on_recovery>REBOOT_ONLY</action_on_recovery>
          </recovery_policy>
        </vm_group>
      </deployment>
    </deployments>
  </tenant>
 </tenants>
</esc_datamodel>
```



第 32 章

CSP クラスタでの VNF の移行

CSP クラスタで VNF/VM を展開、更新、または移行できます。CSP クラスタ内の VNF は、CSP クラスタ内のクラスタ間で移行できます。

CSP クラスタでの VNF の移行

シナリオ 1

CSP-1 が到達可能な場合、VM を CSP-1 から CSP-2 に移行します。

CSP-1 から CSP-2 に VM を移行するには、CSP-1 に到達可能なときにロケータ（vim_id、vim_project）を変更して、NB で ESC に更新を送信します。

次の例は、展開ペイロード/XML からの VM グループを示しています。

```
<vm_group>
            <name>Group1</name>
            <locator>
              <vim_id>CSP-1</vim_id>
              <vim_project>CSP-1</vim_project>
            </locator>
```

次の例は、CSP-2 での VM の移行成功通知を示しています。

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-09-03T05:41:16.299+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>VIM Locator Updated Successfully</status_message>
    <vm_update_type>LOCATOR_UPDATED</vm_update_type>
    <depname>dep</depname>
    <tenant>demo</tenant>
    <depid>06c94f58-b753-425b-b97c-f7adb9140ead</depid>
    <vm_group>group</vm_group>
    <vm_source>
      <vmid>6b0e7179-fd5e-487e-9570-e7ba98cce0ec</vmid>
      <vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</vmname>
     <generated_vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</generated_vmname>

      <vim_id>CSP-2</vim_id>
      <vim_project>CSP-2</vim_project>
```

```
      <interfaces>
        <interface>
          <nicid>0</nicid>
          <type>access</type>
          <port_id>539c6df4-4680-4bba-8a0d-d621947f2228</port_id>
          <admin_state_up>true</admin_state_up>
          <network>Eth0-2</network>
          <subnet/>
          <ip_address>192.168.23.62</ip_address>
          <netmask>255.255.255.0</netmask>
        </interface>
        <interface>
          <nicid>1</nicid>
          <type>trunk</type>
          <port_id>0adc3096-509c-49b7-9bd7-a25bbf2a9345</port_id>
          <admin_state_up>true</admin_state_up>
          <network>Eth0-2</network>
          <subnet/>
        </interface>
      </interfaces>
    </vm_source>
    <event>
      <type>VM_UPDATED</type>
    </event>
  </escEvent>
</notification>

<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-09-03T05:41:16.322+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Service group update completed successfully</status_message>
    <depname>dep</depname>
    <tenant>demo</tenant>
    <tenant_id>demo</tenant_id>
    <depid>06c94f58-b753-425b-b97c-f7adb9140ead</depid>
    <event>
      <type>SERVICE_UPDATED</type>
    </event>
  </escEvent>
```

シナリオ 2

CSP-1 が到達不可能な場合、VM を CSP-1 から CSP-2 に移行します。

初期展開時にリカバリモードが自動で、リカバリポリシーが REBOOT_ONLY であると仮定します。CSP-1 ホストに障害が発生し、CSP-1 の障害が原因で VM に障害が起きたことを ESC が検出したとします。ESC は VM の回復を試みますが、CSP-1 がダウンしているため失敗します。NB は CSP-1 から CSP-2 に VM を移動するための更新を送信します。

次の例は、CSP-1 上 の VM のリカバリ障害通知を示しています。

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-09-03T04:30:18.642+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Recovery event for VM Generated ID
[dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555] triggered.</status_message>
    <depname>dep</depname>
    <tenant>demo</tenant>
    <tenant_id>demo</tenant_id>
```

```
    <depid>06c94f58-b753-425b-b97c-f7adb9140ead</depid>
    <vm_group>group</vm_group>
    <vm_source>
      <vmid>6b0e7179-fd5e-487e-9570-e7ba98cce0ec</vmid>
      <vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</vmname>
     <generated_vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</generated_vmname>

      <vim_id>CSP-1</vim_id>
      <vim_project>CSP-1</vim_project>
    </vm_source>
    <event>
      <type>VM_RECOVERY_INIT</type>
    </event>
  </escEvent>
</notification>

<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-09-03T04:31:20.449+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>FAILURE</status>
    <status_code>500</status_code>
    <status_message> VM [dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555] failed to be
 rebooted.</status_message>
    <depname>dep</depname>
    <tenant>demo</tenant>
    <tenant_id>demo</tenant_id>
    <depid>06c94f58-b753-425b-b97c-f7adb9140ead</depid>
    <vm_group>group</vm_group>
    <vm_source>
      <vmid>6b0e7179-fd5e-487e-9570-e7ba98cce0ec</vmid>
      <vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</vmname>
     <generated_vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</generated_vmname>

      <vim_id>CSP-2</vim_id>
      <vim_project>CSP-2</vim_project>
    </vm_source>
    <event>
      <type>VM_RECOVERY_REBOOT</type>
    </event>
  </escEvent>
</notification>


<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-09-03T04:41:20.844+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>FAILURE</status>
    <status_code>500</status_code>
    <status_message>Recovery: Recovery completed with errors for VM:
[dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555]</status_message>
    <depname>dep</depname>
    <tenant>demo</tenant>
    <tenant_id>demo</tenant_id>
    <depid>06c94f58-b753-425b-b97c-f7adb9140ead</depid>
    <vm_group>group</vm_group>
    <vm_source>
      <vmid>6b0e7179-fd5e-487e-9570-e7ba98cce0ec</vmid>
      <vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</vmname>
     <generated_vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</generated_vmname>

      <vim_id>CSP-1</vim_id>
      <vim_project>CSP-1</vim_project>
    </vm_source>
    <vm_target>
```

```
      <vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</vmname>
     <generated_vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</generated_vmname>

    </vm_target>
    <event>
      <type>VM_RECOVERY_COMPLETE</type>
    </event>
  </escEvent>
</notification>
```

3 つの CSP（CSP-1、CSP-2、CSP-3）のクラスタがあるとします。VM が CSP-1 に展開されています。

始める前に

- VIM コネクタを作成する必要があります。詳細については、「CSP クラスタへの VIM コネクタの追加」の章を参照してください。
- VM は、基盤となる同じストレージで展開されます。詳細については、「CSP クラスタでの ESC を使用した VNF の展開」の章を参照してください。

次のシナリオは、VM の移行を示しています。

手順

ステップ 1 次の展開ペイロードでロケータの詳細を更新します。

```
(Update vim_id, vim_project to CSP-1 → CSP-2)
<locator>
<vim_id>CSP-2</vim_id>
<vim_project>CSP-2</vim_project>
</locator>
```

ステップ 2 次のコマンドを実行して VM を移行します。

```
esc_nc_cli --user <username> --password <password> edit-config deploy_csp_2.xml
```

ペイロードの例：

```
deploy_csp_1.xml
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <flavors>
    <flavor>
      <name>FLAVOR_2_4096_10000</name>
      <vcpus>2</vcpus>
      <memory_mb>4096</memory_mb>
      <root_disk_mb>10240</root_disk_mb>
    </flavor>
  </flavors>
  <tenants>
    <tenant>
      <name>name1</name>
      <vim_mapping>false</vim_mapping>
      <deployments>
        <deployment>
          <name>dep</name>
          <vm_group>
            <name>Group1</name>
            <locator>
```

```
      <vim_id>CSP-1</vim_id>
      <vim_project>CSP-1</vim_project>
    </locator>
    <image>csr1000v-universalk9.16.06.01.qcow2</image>
    <flavor>FLAVOR_2_4096_10000</flavor>
    <bootup_time>600</bootup_time>
    <recovery_wait_time>60</recovery_wait_time>
<recovery_policy>
        <recovery_type>AUTO</recovery_type>
        <action_on_recovery>REBOOT_ONLY</action_on_recovery>
        <max_retries>1</max_retries>
</recovery_policy>
    <interfaces>
       <interface>
        <nicid>0</nicid>
        <type>virtual</type>
        <model>virtio</model>
        <network>Eth0-2</network>
        <ip_address>192.168.23.61</ip_address>
        </interface>
     <interface>
        <nicid>1</nicid>
        <type>virtual</type>
        <model>virtio</model>
        <network>Eth0-2</network>
          <ip_address>192.168.23.61</ip_address>
        <admin_state_up>false</admin_state_up>
      </interface>
    </interfaces>
    <kpi_data>
      <kpi>
        <event_name>VM_ALIVE</event_name>
        <metric_value>50</metric_value>
        <metric_cond>GT</metric_cond>
        <metric_type>UINT32</metric_type>
        <metric_occurrences_true>3</metric_occurrences_true>
        <metric_occurrences_false>3</metric_occurrences_false>
        <metric_collector>
          <type>ICMPPing</type>
          <nicid>0</nicid>
          <poll_frequency>15</poll_frequency>
          <polling_unit>seconds</polling_unit>
          <continuous_alarm>false</continuous_alarm>
        </metric_collector>
      </kpi>
    </kpi_data>
    <rules>
      <admin_rules>
        <rule>
          <event_name>VM_ALIVE</event_name>
          <action>ALWAYS log</action>
          <action>FALSE recover autohealing</action>
          <action>TRUE servicebooted.sh</action>
        </rule>
      </admin_rules>
    </rules>
    <config_data>
      <configuration>
        <dst>iosxe_config.txt</dst>
        <file>file:///var/tmp/csr_config.sh</file>
      </configuration>
    </config_data>
    <scaling>
      <min_active>1</min_active>
```

```
  <max_active>1</max_active>
  <elastic>true</elastic>
  <static_ip_address_pool>
    <network>Eth0-2</network>
    <netmask>255.255.255.0</netmask>
     <gateway>192.168.23.1</gateway>
     <ip_address>192.168.23.61</ip_address>
  </static_ip_address_pool>
</scaling>
<extensions>
  <extension>
    <name>interfaces</name>
    <containers>
       <container>
        <name>0</name>
        <properties>
          <property>
            <name>passthroughMode</name>
            <value>none</value>
          </property>
          <property>
            <name>tagged</name>
            <value>false</value>
          </property>
          <property>
            <name>type</name>
            <value>access</value>
          </property>
          <property>
            <name>vlan</name>
            <value>1</value>
          </property>
        </properties>
      </container>
     <container>
        <name>1</name>
        <properties>
          <property>
            <name>passthroughMode</name>
            <value>none</value>
          </property>
          <property>
            <name>tagged</name>
            <value>false</value>
          </property>
          <property>
            <name>type</name>
            <value>access</value>
          </property>
           <property>
            <name>bandwidth</name>
            <value>160</value>
          </property>
          <property>
            <name>vlan</name>
           <value>2</value>
          </property>
        </properties>
      </container>
    </containers>
  </extension>
  <extension>
    <name>serial_ports</name>
    <containers>
```

```
                    <container>
                      <name>0</name>
                      <properties>
                        <property>
                          <name>serial_type</name>
                          <value>console</value>
                        </property>
                      </properties>
                    </container>
                  </containers>
                </extension>
                <extension>
                  <name>image</name>
                  <properties>
                    <property>
                      <name>disk-resize</name>
                      <value>true</value>
                    </property>
                    <property>
                      <name>disk_type</name>
                      <value>virtio</value>
                    </property>
                    <property>
                      <name>disk_storage_name</name>
                      <value>gluster</value>
                    </property>
                  </properties>
                </extension>
              </extensions>
            </vm_group>
          </deployment>
        </deployments>
      </tenant>
    </tenants>
</esc_datamodel>
```

ステップ 3 移行の成功または失敗の通知を確認します。

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-09-03T05:41:16.299+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>VIM Locator Updated Successfully</status_message>
    <vm_update_type>LOCATOR_UPDATED</vm_update_type>
    <depname>dep</depname>
    <tenant>demo</tenant>
    <depid>06c94f58-b753-425b-b97c-f7adb9140ead</depid>
    <vm_group>group</vm_group>
    <vm_source>
      <vmid>6b0e7179-fd5e-487e-9570-e7ba98cce0ec</vmid>
      <vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</vmname>
     <generated_vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</generated_vmname>

      <vim_id>CSP-2</vim_id>
      <vim_project>CSP-2</vim_project>
      <interfaces>
        <interface>
          <nicid>0</nicid>
          <type>access</type>
          <port_id>539c6df4-4680-4bba-8a0d-d621947f2228</port_id>
          <admin_state_up>true</admin_state_up>
          <network>Eth0-2</network>
          <subnet/>
```

```
          <ip_address>192.168.23.62</ip_address>
          <netmask>255.255.255.0</netmask>
        </interface>
        <interface>
          <nicid>1</nicid>
          <type>trunk</type>
          <port_id>0adc3096-509c-49b7-9bd7-a25bbf2a9345</port_id>
          <admin_state_up>true</admin_state_up>
          <network>Eth0-2</network>
          <subnet/>
        </interface>
      </interfaces>
    </vm_source>
    <event>
      <type>VM_UPDATED</type>
    </event>
  </escEvent>
</notification>

<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-09-03T05:41:16.322+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Service group update completed successfully</status_message>
    <depname>dep</depname>
    <tenant>demo</tenant>
    <tenant_id>demo</tenant_id>
    <depid>06c94f58-b753-425b-b97c-f7adb9140ead</depid>
    <event>
      <type>SERVICE_UPDATED</type>
    </event>
  </escEvent>
</notification>
```

シナリオ 3

同じ CSP のローカルストレージから Gluster ストレージに VM を移行します。

VM をローカルから Gluster に移行するために、NB で次のプロパティを含む更新を送信します。

a) クラスタ名を持つ新しい VIM コネクタを追加します。

```
<property>
          <name>cluster_name</name>
          <value>Cluster_Test</value>
          </property>
```

新しいVIMコネクタの追加の詳細については、「CSPクラスタへのVIMコネクタの追加」を参照してください。

b) 新しいコネクタを追加した後、ロケータと disk_storage_name を展開ペイロードで **Gluster** に更新し、設定の変更を有効にします。

次の例は、disk_storage_nameをイメージ拡張プロパティの下にGlusterとして追加し、クラスタ VIM コネクタで更新する方法を示しています。

```
<vm_group>
<name>Group1</name>
<locator>
<vim_id>CSP-1</vim_id>
<vim_project>CSP-1</vim_project>
```

```
</locator>
<extension>
                <name>image</name>
                <properties>
                  <property>
                    <name>disk-resize</name>
                    <value>true</value>
                  </property>
                  <property>
                    <name>disk_type</name>
                    <value>virtio</value>
                  </property>
                  <property>
                    <name>disk_storage_name</name>
                    <value>gluster</value>
                  </property>
                </properties>
              </extension>
```

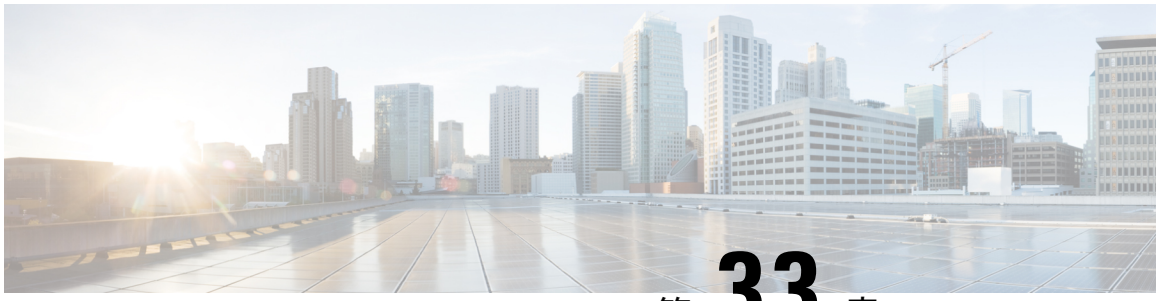
c) 次のコマンドを実行して VIM を展開します。

```
esc_nc_cli --user <username> --password <password> edit-config deploy.xml
```

次の通知で移行の成功/失敗を確認します。

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-09-03T05:41:16.299+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>VIM Locator Updated Successfully</status_message>
    <vm_update_type>LOCATOR_UPDATED</vm_update_type>
    <depname>dep</depname>
    <tenant>demo</tenant>
    <depid>06c94f58-b753-425b-b97c-f7adb9140ead</depid>
    <vm_group>group</vm_group>
    <vm_source>
      <vmid>6b0e7179-fd5e-487e-9570-e7ba98cce0ec</vmid>
      <vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</vmname>

<generated_vmname>dep_group_0_46e607a8-b797-4056-96f3-42a90a63b555</generated_vmname>

      <vim_id>CSP-2</vim_id>
      <vim_project>CSP-2</vim_project>
      <interfaces>
        <interface>
          <nicid>0</nicid>
          <type>access</type>
          <port_id>539c6df4-4680-4bba-8a0d-d621947f2228</port_id>
          <admin_state_up>true</admin_state_up>
          <network>Eth0-2</network>
          <subnet/>
          <ip_address>192.168.23.62</ip_address>
          <netmask>255.255.255.0</netmask>
        </interface>
        <interface>
          <nicid>1</nicid>
          <type>trunk</type>
          <port_id>0adc3096-509c-49b7-9bd7-a25bbf2a9345</port_id>
          <admin_state_up>true</admin_state_up>
          <network>Eth0-2</network>
          <subnet/>
        </interface>
      </interfaces>
    </vm_source>
```

```
    <event>
      <type>VM_UPDATED</type>
    </event>
  </escEvent>
</notification>

<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-09-03T05:41:16.322+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Service group update completed successfully</status_message>
    <depname>dep</depname>
    <tenant>demo</tenant>
    <tenant_id>demo</tenant_id>
    <depid>06c94f58-b753-425b-b97c-f7adb9140ead</depid>
    <event>
      <type>SERVICE_UPDATED</type>
    </event>
  </escEvent>
</notification>
```



第 33 章

展開状態とイベント

ESCの展開ライフサイクルは、さまざまな状態を使用して表されます。データモデルでは、展開ライフサイクル中におけるサービスとVNFのさまざまな状態を定義します。一般に、展開またはサービスライフサイクルは2つの段階で表されます。サービスには、1つ以上の異なるタイプのVMグループが含まれています。VMグループは、同じタイプのVMまたはVNFのグループを表します。展開またはサービスリクエストを受信すると、ESCはそのリクエストを検証し、リクエストの処理を受け入れます。処理中、ESCはデータモデルで定義されたリソースを使用して、基盤となるVIMにVMまたはVNFを展開します。ESCは、定義されたKPIとアクションに基づいてこれらのVMおよびVNFをモニタします。設定されたポリシーとアクションの定義に従い、ESCは自動修復、スケールイン、スケールアウト、およびその他のワークフローをトリガーします。

展開またはその他のワークフローの間、サービスまたは展開の状態、およびVMまたはVNFの状態が変化してイベントが送信されます。状態とイベントは、展開のステータスを識別する上で重要な役割を果たします。展開の現在の状態は、運用データに表示されます。ESCは、展開、VM、またはVNFの状態が通知が必要な状態に変わると、通知またはイベントを送信します。データモデルでは、すべての異なる状態とイベントが定義されます。

展開またはサービスの状態

サービス状態は、完全なサービスまたは展開の状態を表します。サービスの状態は、さまざまなVMまたはVNFの状態、VMグループ内のVMの状態、およびサービス、VM、またはVNFで実行されている現在のワークフローによっても異なります。サービスまたは展開の状態は、展開全体の集約サマリーです。

表 *17*：展開またはサービスの状態

サービス ステート	説明
SERVICE_UNDEF_STATE	初期のサービス状態。ESC が展開の処理を開始するまで、サービスはこの状態になります。

サービス ステート	説明
SERVICE_DEPLOYING_STATE	この状態では、サービスまたは展開のためにVM が展開されています。
SERVICE_INERT_STATE	この状態では、展開に含まれる VM は展開されますが、アクティブにはなっておらず、起動もされていません。
SERVICE_ACTIVE_STATE	この状態では、展開に含まれるすべての VM が展開され、稼働しています。
SERVICE_ERROR_STATE	展開、リカバリ、スケールインまたはスケールアウト、あるいはその他のワークフロー中にエラーが発生した場合、サービスはこの状態になります。
SERVICE_UNDEPLOYING_STATE	この状態では、サービスまたは展開に対する VM の展開が解除されています。
SERVICE_STOPPING_STATE	この状態では、サービスアクション要求により、サービス配下の VM または VNF が停止されています。
SERVICE_STOPPED_STATE	この状態では、サービスアクション要求により、サービス配下の VM または VNF が停止しています。
SERVICE_STARTING_STATE	この状態では、サービスアクション要求により、サービス配下の VM または VNF が開始されます。
SERVICE_REBOOTING_STATE	この状態では、サービスアクション要求により、サービス配下の VM または VNF が再起動されます。

VM または VNF の状態

VM または VNF の状態は、サービスまたは展開内の特定の VM または VNF の状態を表します。VM の状態は、特定の VNF の現在の状態と、この VM または VNF で実行されているワークフローを識別するための鍵となります。

表 *18 : VM* または *VNF* の状態

VM 状態	説明
VM_UNDEF_STATE	この VM を展開する前の VM または VNF の初期状態。

VM 状態	説明
VM_DEPLOYING_STATE	VM または VNF が VIM に展開されています。
VM_MONITOR_UNSET_STATE	VM または VNF は VIM に展開されますが、モニタリングルールは適用されません。
VM_MONITOR_DISABLED_STATE	VM アクション要求またはリカバリワークフローが原因で、VM または VNF に適用されたモニタリングまたは KPI ルールが有効になっていません。
VM_STOPPING_STATE	VM または VNF が停止しています。
VM_SHUTOFF_STATE	VM または VNF が停止またはシャットオフ状態です。
VM_STARTING_STATE	VM または VNF が開始されています。
VM_REBOOTING_STATE	VM または VNF が再起動されています。
VM_INERT_STATE	VM または VNF は展開されていますが、稼働していません。KPI モニタが適用され、VM が稼働するのを待機しています。
VM_ALIVE_STATE	VM または VNF が展開され、モニタまたは KPI メトリックに従って正常に起動または稼働しています。
VM_UNDEPLOYING_STATE	VM または VNF が展開解除または終了されています。
VM_ERROR_STATE	展開またはその他の操作が失敗した場合、VM または VNF はエラー状態になります。

ESC では、イベントは展開またはその他のワークフローの現在のステータスを提供する上で重要な役割を果たします。詳細については、「イベント通知またはコールバックイベント」を参照してください。

イベント通知またはコールバックイベント

ESC では、イベントは展開またはその他のワークフローの現在のステータスを提供する上で重要な役割を果たします。Netconf インターフェイスでは ESC が通知を送信し、REST インターフェイスでは ESC がコールバックイベントを送信します。ここでは、ESC によって送信されるすべての通知またはコールバックイベントについて説明します。

展開または VNF のイベント通知またはコールバック

以下に定義する通知またはコールバックイベントのタイプは、展開のライフサイクル中にノースバウンドに送信されるイベントです。これらのイベントは、展開要求を受信して処理が開始されたときに ESC から送信されます。ESC は、ステージの成功または失敗を示すステータスメッセージとともに、すべてのステージに関する通知を送信します。

表 *19 :* 展開または *VNF* のイベント通知またはコールバック

イベント ステート	ワークフロー	説明
VM_DEPLOYED	展開	VM または VNF が展開されている場合。VM または VNF の展開が成功した場合は成功、そうでない場合は失敗。VM または VNF ごとに送信されます。
VM_ALIVE	展開	展開された VM または VNF がモニタまたは KPI メトリックに従って正常に起動または稼働している場合。VM または VNF ごとに送信されます。
SERVICE_ALIVE	展開	展開またはサービスが完了し、すべての VM が動作している場合、またはいずれかが失敗した場合。
VM_UNDEPLOYED	展開解除	VM または VNF が展開解除されたとき。VM または VNF が正常に展開解除された場合は成功、そうでない場合は失敗。VM または VNF ごとに送信されます。
SERVICE_UNDEPLOYED	展開解除	すべての VM または VNF が展開解除されたとき。展開中のすべての VM とリソースが正常に削除された場合は成功、そうでない場合は失敗。
VM_UPDATED	展開の更新	展開が成功すると、各 VM グループの詳細が更新されます。更新が完了した場合は成功、そうでない場合は失敗。VM\VNF ごとに送信されます

イベント ステート	ワークフロー	説明
SERVICE_UPDATED	展開の更新	展開が成功し、すべての更新が完了した場合。更新が完了した場合は成功、そうでない場合は失敗。
SERVICE_UPDATING_STATE	展開の更新	この状態では、VM、VNF、およびこの展開のデータなどの一部のコンポーネントが更新されます。 。
VM_RECOVERY_INIT	リカバリ	リカバリワークフローがトリガーされると、リカバリ初期化通知が送信されます。
VM_RECOVERY_DEPLOYED	リカバリ	VM または VNF がリカバリワークフローの一部として展開されると、リカバリ展開通知が送信されます。
VM_RECOVERY_UNDEPLOYED	リカバリ	VM または VNF がリカバリワークフローの一部として展開解除されると、リカバリ展開解除通知が送信されます。
VM_RECOVERY_COMPLETE	リカバリ	VMのリカバリが完了すると、リカバリ完了通知が送信されます。VM が回復した場合は成功、そうでない場合は失敗。
VM_RECOVERY_REBOOT	リカバリ	VM または VN Fがリカバリの一部として再起動されると、リカバリ再起動通知が送信されます。再起動が成功した場合は成功、そうでない場合は失敗。
VM_RECOVERY_CANCELLED	リカバリ	リカバリがトリガーされたが、リカバリ待機時間の前に VM がアクティブ状態になったときに、リカバリキャンセル通知が送信されます。

イベント ステート	ワークフロー	説明
VM_MANUAL_RECOVERY_NEEDED	手動回復	リカバリがトリガーされたが、手動リカバリポリシーが設定されている場合に、手動リカバリ必要通知が送信されます。
VM_MANUAL_RECOVERY_NO_NEED	手動回復	手動回復ポリシーが設定された状態でリカバリがトリガーされ、VM が再びアクティブになると、手動リカバリ不要通知が送信されます。
VM_SCALE_OUT_INIT	スケール アウト	スケールアウトワークフローがトリガーされると、スケールアウト初期化通知が送信されます。
VM_SCALE_OUT_DEPLOYED	スケール アウト	VM がスケールアウトの一部として展開されると、スケールアウト展開通知が送信されます。
VM_SCALE_OUT_COMPLETE	スケール アウト	スケールアウトワークフローが完了すると、スケールアウト完了通知が送信されます。
VM_SCALE_IN_INIT	スケールイン	ワークフローのスケールが開始されると、スケールイン初期化通知が送信されます。
VM_SCALE_IN_COMPLETE	スケールイン	ワークフローのスケールが完了すると、スケールイン完了通知が送信されます。

展開または VNF 操作のイベント通知またはコールバックイベントタイプ

次に定義する通知またはコールバックイベントタイプは、ユーザが実行したさまざまな操作またはアクション中にノースバウンドに送信されるイベントです。これらのイベントは、アクション要求を受信して処理が開始されたときに ESC から送信されます。ESC は、ステージの成功または失敗を示すステータスメッセージとともに、すべてのステージに関する通知を送信します。

表 *20 :* 展開または *VNF* 操作のイベント通知またはコールバックイベントタイプ

イベント ステート	ワークフロー	説明
VM_REBOOTED	VM アクション	VM または VNF が再起動されると、イベントが送信されます。
VM_STOPPED	VM アクション	VM または VNF が停止すると、イベントが送信されます。
VM_STARTED	VM アクション	VM または VNF が開始されると、イベントが送信されます。
SERVICE_STOPPED	展開アクション	サービス停止イベントは、サービス内のすべての VM\VNF を停止する要求が完了すると送信されます。
SERVICE_STARTED	展開アクション	サービス開始イベントは、サービス内のすべての VM\VNF を開始する要求が完了すると送信されます。
SERVICE_REBOOTED	展開アクション	サービス再起動イベントは、サービス内のすべての VM または VNF を再起動する要求が完了すると送信されます。
HOST_DISABLE	ホストアクション/再展開	（OpenStack のみ）ホストを無効にする要求が完了すると、イベントが送信されます。
HOST_ENABLE	ホストアクション/ 再展開	（OpenStackのみ）ホストを有効にする要求が完了すると、イベントが送信されます。
VIM_OPERATIONAL_STATE	該当なし	このイベントは、ESC が VIM の動作状態が変更されたことを検出すると送信されます。

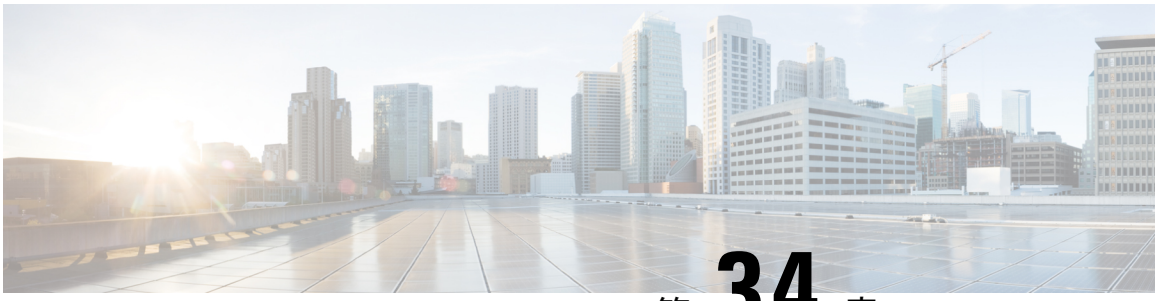
リソースのイベント通知またはコールバックイベントタイプ

次に定義する通知またはコールバックイベントタイプは、リソースの作成または削除中にノースバウンドに送信されるイベントです。これらのイベントは、要求を受信して処理が開始され

たときに ESC から送信されます。ESC は、ステージの成功または失敗を示すステータスメッセージとともに、すべてのステージに関する通知を送信します。

表 *21 :* リソースのイベント通知またはコールバックイベントタイプ

イベント ステート	ワークフロー	説明
CREATE_TENANT	テナント	テナントが作成されました
DELETE_TENANT	テナント	テナントが削除されました
CREATE_NETWORK	ネットワーク（Network）	ネットワークが作成されました
DELETE_NETWORK	ネットワーク（Network）	ネットワークが削除されました
CREATE_SUBNET	サブネット	サブネットが作成されました
DELETE_SUBNET	サブネット	サブネットが削除されました
CREATE_IMAGE	イメージ	イメージが作成されました
DELETE_IMAGE	イメージ	画像が削除されました
CREATE_FLAVOR	フレーバ	フレーバが作成されました
DELETE_FLAVOR	フレーバ	フレーバが削除されました



第 34 章

LCS を使用した VNF ソフトウェアのアップグレード

ESC は、展開の更新中の VNF ソフトウェア アプリケーションのアップグレードをサポートします。ポリシーデータモデルを使用して、VNF アップグレードをサポートする新しいライフサイクルステージ（条件）が導入されます。VNF アップグレードポリシーは、VM グループごとに異なる場合があります。これらのポリシーは VM のグループに適用され、展開全体ではなく <vm_group> の下で指定できます。

VNF ソフトウェアのアップグレード

ESC は、展開内の初期イメージまたは基本イメージのアップグレードをサポートします。ESC ポリシーフレームワークは、新規および既存の VM のソフトウェアをアップグレードするためのカスタムスクリプトを提供します。ESC ポリシーフレームワークが最新であれば、VM の増分更新がサポートされます。

- 既存の VM のアップグレード：次の ESC ポリシーフレームワークは、ソフトウェアバージョンの更新前にすでに展開されている既存の VM をアップグレードするためのスクリプトをトリガーします。

  ```
  LCS::DEPLOY_UPDATE::POST_VM_SOFTWARE_VERSION_UPDATED
  ```

- 新しい VM のアップグレード：次の ESC ポリシーフレームワークは、導入時、回復時、またはスケールアウト時に新しい VM をアップグレードするためのスクリプトをトリガーします。

  ```
  LCS::DEPLOY::POST_VM_ALIVE
  ```

ボリュームを使用した VNF アップグレードの詳細については、「ボリュームを使用した VNF ソフトウェアのアップグレード」を参照してください。

VNF ソフトウェアバージョンの更新とソフトウェアアップグレードのトリガー

このシナリオでは、カスタムスクリプトを使用してソフトウェアアップグレードをトリガーする手順について説明します。次の例では、CSR VM がアップグレードされます。`csr_dep2.xml` を使用したサービスの更新により、カスタム スクリプト アクション LCS::DEPLOY_UPDATE::POST_VM_SOFTWARE_VERSION_UPDATED がトリガーされます。LCS は最初にその VM のモニタリングを無効にしてから、`csr_upgrade.exp` スクリプトを呼び出します。スクリプトが CSR に接続し、指定された upgrade.bin を CSR のブートフラッシュに scp し、ブートローダに新しい bin ファイルを指定し、CSR VM を再起動します。その後、bootup_time をリセットして、モニタリングを有効にします。bootup_time を使用すると、CSR は ESC によって再展開されることなく再起動を完了できます。

手順

ステップ 1 ESC VM を展開します。

ステップ 2 デイゼロ設定を `/var/tmp/csp-csr-day0-config` として ESC VM にアップロードします。

ステップ 3 カスタム アップグレード スクリプトを ESC VM にアップロードします。たとえば、csr_upgrade.exp スクリプトを `/var/tmp/csr_upgrade.exp` として ESC VM にアップロードします。

ステップ 4 `chmod +x /var/tmp/csr_upgrade.exp` を実行します。

ステップ 5 初期展開データモデル（dep.xml など）を編集して、関連する IP、ユーザ名、パスワード、および CSR のアップグレードバージョンを含めます。

ステップ 6 展開データモデル（dep.xml）のソフトウェアバージョンを編集して、アップグレードされた CSR バージョンを反映させます。

ステップ 7 ESC ユーザのホームディレクトリに CSR アップグレードをアップロードします。

ステップ 8 展開された CSR VM をアップグレードします。Run the command: `esc_nc_cli --user <username> --password <password> edit-config csr_dep2.xml`

ボリュームを使用した VNF ソフトウェアのアップグレード

サービスの初回展開時に、データモデルには、将来のソフトウェアアップグレード用に設定されたポリシーがあります。展開の更新要求を受信すると、展開の更新の一部として VM のアップグレードが開始されます。LCS::DEPLOY_UPDATE::VM_PRE_VOLUME_DETACH は、ESC がボリュームをデタッチする前にトリガーされます。このライフサイクルステージでは、デタッチする前にボリュームをアンマウントするスクリプトがサポートされています。ESC は、古いバージョンのソフトウェアを含む古いボリュームをデタッチし、削除します。ボリューム

が正常にデタッチされると、LCS::DEPLOY_UPDATE::VM_POST_VOLUME_DETACHED がトリガーされます。さらなるクリーンアップのため、この LCS でスクリプトが実行されます。新しいソフトウェアバージョンの新しいボリュームがアタッチされると、LCS::DEPLOY_UPDATE::VM_VOLUME_ATTACHED がトリガーされます。ESC は、ソフトウェアの新しいバージョンを含む新しいボリュームを作成してアタッチします。ボリュームをマウントし、ソフトウェアのインストールをトリガーするスクリプトが実行されます。ボリュームがアタッチされると、ESC が VM のソフトウェアバージョンを更新した後に、LCS::DEPLOY_UPDATE::VM_SOFTWARE_VERSION_UPDATED がトリガーされます。この段階で、ソフトウェアアップグレードの設定を完了するためのスクリプトが実行されます。

VNF ソフトウェアアップグレードのデータモデル：

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>test</name>
      <deployments>
        <deployment>
          <name>dep</name>
          <vm_group>
            <name>Group1</name>
            <volumes>
              <volume nc:operation="delete">
                <name>v1.0</name>
                <volid>0</volid>
              </volume>
              <volume>
                <name>v2.0</name>
                <volid>1</volid>
                <sizeunit>GiB</sizeunit>
                <size>2</size>
                <bus>virtio</bus>
                <type>lvm</type>
                <image>Image-v2</image>
              </volume>
            </volumes>
            <software_version>2.0</software_version>
            <policies>
              <policy>
                <name>SVU1</name>
                <conditions>
                  <condition>
                    <name>LCS::DEPLOY_UPDATE::PRE_VM_VOLUME_DETACH</name>
                  </condition>
                </conditions>
                <actions>
                  <action>
                    <name>LOG</name>
                    <type>pre_defined</type>
                  </action>
                </actions>
              </policy>
              <policy>
                <name>SVU2</name>
                <conditions>
                  <condition>
                    <name>LCS::DEPLOY_UPDATE::POST_VM_VOLUME_ATTACHED</name>
                  </condition>
                </conditions>
                <actions>
```

```
                    <action>
                      <name>LOG</name>
                      <type>pre_defined</type>
                    </action>
                  </actions>
                </policy>
                <policy>
                  <name>SVU3</name>
                  <conditions>
                    <condition>
                      <name>LCS::DEPLOY_UPDATE::POST_VM_SOFTWARE_VERSION_UPDATED</name>
                    </condition>
                  </conditions>
                  <actions>
                    <action>
                      <name>LOG</name>
                      <type>pre_defined</type>
                    </action>
                  </actions>
                </policy>
              </policies>
            </vm_group>
          </deployment>
        </deployments>
      </tenant>
    </tenants>
</esc_datamodel>
```

このデータモデルでは、volid が 0 の既存のボリューム v1.0 が削除されます。volid が 1 の新しいボリューム v2.0 が追加されます。ソフトウェアバージョンである <software_version> 値が 1.0 から 2.0 に変更されます。VNF ソフトウェアアップグレード用に 3 つのポリシーが追加されました。

（注）

- 新しいボリュームを削除して作成する代わりに、ボリュームのプロパティを更新できます。name、vol_id、および image プロパティを保持できます。上記の 3 つのプロパティのいずれかが変更されると、ボリュームが削除され、再度作成されます。
- ボリュームサイズを拡張でき、ブート可能プロパティを変更できます。ボリュームタイプなどのその他のプロパティやイメージプロパティを変更すると、ボリュームが再度作成されます。
- ボリューム ID を更新するには、ボリュームを削除し、別のボリューム ID でボリュームを再度追加する必要があります。
- ESC によって作成されたボリュームは、同じボリューム ID のアウトオブバンドボリュームによって更新することはできません。その逆も同様です。

ボリュームを使用した VNF ソフトウェアアップグレードでサポートされるライフサイクルステージ（LCS）

各ライフサイクルステージには、条件とアクションがあります。条件に基づいて、アクションが実行されます。ポリシー主導型データモデルの詳細については、ポリシー駆動型データモデ

ル（199 ページ）を参照してください。VNF ソフトウェアアップグレードには、次の 3 つの条件が設定されています。

条件名	範囲	説明
LCS::DEPLOY_UPDATE::VM_PRE_VOLUME_DETACH	展開	ESC がボリュームをデタッチする直前にトリガーされます
LCS::DEPLOY_UPDATE::POST_VM_VOLUME_DETACHED	展開	ESC がボリュームをデタッチした直後にトリガーされます
LCS::DEPLOY_UPDATE::POST_VM_VOLUME_ATTACHED	展開	ESC が新しいボリュームをアタッチした直後にトリガーされます
LCS:DEPLOY_UPDATE:POST_VM_SOFTWARE_VERSION_UPDATED	展開	ESC が VM のソフトウェアバージョンを更新した直後にトリガーされます

LCS::DEPLOY_UPDATE::PRE_VM_VOLUME_DETACH

この LCS 条件は、ESC がボリュームをデタッチする前にトリガーされます。デタッチする前に、ボリュームをアンマウントするスクリプトが実行されます。

```
<policy>
    <name>SVU1</name>
    <conditions>
        <condition>
            <name>LCS::DEPLOY_UPDATE::PRE_VM_VOLUME_DETACH</name>
        </condition>
    </conditions>
    <actions>
        <action>
            <name>LOG</name>
            <type>pre_defined</type>
        </action>
    </actions>
</policy>
```

LCS::DEPLOY_UPDATE::POST_VM_VOLUME_ATTACHED

この LCS は、ESC が新しいボリュームをアタッチした後にトリガーされます。ボリュームをマウントし、新しいアプリケーションを新しいボリュームにインストールするスクリプトが実行されます。

```
<policy>
    <name>SVU2</name>
    <conditions>
        <condition>
            <name>LCS::DEPLOY_UPDATE::POST_VM_VOLUME_ATTACHED</name>
        </condition>
    </conditions>
    <actions>
        <action>
            <name>LOG</name>
```

```
            <type>pre_defined</type>
         </action>
    </actions>
</policy>
```

LCS::DEPLOY_UPDATE::POST_VM_SOFTWARE_VERSION_UPDATED

この LCS は、ESC が VM のソフトウェアバージョンを更新した後にトリガーされます。ソフトウェアのアップグレードを完了するための最終設定を実行するスクリプトが実行されます。

```
<policy>
   <name>SVU3</name>
   <conditions>
      <condition>
         <name>LCS::DEPLOY_UPDATE::POST_VM_SOFTWARE_VERSION_UPDATED</name>
      </condition>
   </conditions>
   <actions>
      <action>
         <name>LOG</name>
         <type>pre_defined</type>
      </action>
   </actions>
</policy>
```

（注） 上記の3つのポリシーはすべて、LOGアクションをデータモデルサンプルの定義済みアクションとして示しています。スクリプトの実行が必要な場合は、SCRIPT アクションを追加できます。サンプルスクリプトについては、以下の「スクリプトアクション」セクションを参照してください。

スクリプト アクション

上記の例では、すべてのアクションは事前定義されたログです。代わりにカスタムスクリプトを使用できます。

```
<action>
   <name>unmount_volume</name>
   <type>SCRIPT</type>
   <properties>
      <property>
         <name>script_filename</name>
         <value>/opt/cisco/esc/esc-scripts/unmount.sh</value>
      </property>
      <property>
         <name>user_param</name>
         <value>value</value>
      </property>
   </properties>
</action>
```

すべてのプロパティ名と値のペアは、スペースで区切られたパラメータとしてスクリプトに渡されます。上記の例では、unmount.sh 値は次のようにスクリプトによって呼び出されます。

```
/opt/cisco/esc/esc-scripts/unmount.sh user_param value
```

ESC 内部 ID を指定したスクリプトに渡すように、事前に作成されたプロパティ名を設定できます。事前に作成されたプロパティ名は次のとおりです。

```
<property>
  <name>internal_deployment_id</name>
</property>
<property>
  <name>external_deployment_id</name>
</property>
<property>
  <name>deployment_name</name>
</property>
<property>
  <name>internal_tenant_id</name>
</property>
<property>
  <name>external_tenant_id</name>
</property>
```

ESCが生成する、事前に作成されたプロパティ名と値を含むスクリプトの例を次に示します。

```
script_name.sh deployment_name my-deployment-name external_deployment_id
18fbcfd5-8b63-44e0-97ec-68de25902917
external_tenant_id my-tenant-id internal_deployment_id my-tenant-idmy-deployment-name
internal_tenant_id my-tenant-id
```

デフォルトで、ESC ではスクリプトの実行が完了するまでに 15 分かかります。一部のスクリプトは完了までにさらに時間がかかる場合があります。オプションのプロパティを指定して、タイムアウト値を秒単位で延長できます。次の例では、スクリプトのタイムアウトは 3600 秒に設定されています。

```
<property>
    <name>wait_max_timeout</name>
    <value>3600</value>
</property>
```

仮想ネットワーク機能ソフトウェアアップグレードの通知

通知は、VNF ソフトウェアアップグレードの各段階でトリガーされます。

デタッチされたボリューム

```
status SUCCESS
        status_code 200
        status_message Detached 1 volume: [Volume=test-esc-1,volid=1]
        depname dep
        tenant test
        tenant_id 9132cc90b8324a1c95a6c00975af6206
        depid eb4fe3b5-138d-41a3-b6ff-d6fa9035ca6c
        vm_group Group1
        vm_source {
            vmid cd4eeb61-61db-45a6-9da1-793be08c4de6
            hostid 8e96b8830d7bfbb337ce665586210fcca9644cbe238240e207350735
            hostname my-server-5
            software_version 1.0
            interfaces {
                interface {
                    nicid 0
                    type virtual
```

```
                    port_id 26412180-45cf-4f0b-ab45-d05bb7ca7091
                    network 943fda9e-79f8-400c-b442-3506f102721a
                    subnet e313b95c-ca1f-4c81-8d60-c9e721a85d0b
                    ip_address 192.168.0.56
                    mac_address fa:16:3e:18:90:1e
                    netmask 255.255.255.0
                    gateway 192.168.0.1
                }
            }
            volumes {
                volume {
                    display_name test-esc-1__v0_0_0_1
                    external_id 5d008a12-6fb1-492a-b648-4cf7fc8c68b1
                    bus virtio
                    type lvm
                    size 2
                }
            }
        }
        vm_target {
        }
        event {
            type VM_UPDATED
        }
    }
}
```

削除されたボリューム

```
notification {
    eventTime 2016-11-24T00:27:25.457+00:00
    escEvent {
        status SUCCESS
        status_code 200
        status_message Removed 1 volume: [Volume=test-esc-3,volid=1]
        depname dep
        tenant test
        tenant_id 9132cc90b8324a1c95a6c00975af6206
        depid f938ca24-d0c2-42b3-a757-66b0543fe0a6
        vm_group Group1
        vm_source {
            vmid 91379ad1-1cfc-4a10-abaf-068d01ae92b9
            hostid 101f55110748903af4844a2517e854f64843b9ac8d880ad68be8af59
            hostname my-server-4
            software_version 1.0
            interfaces {
                interface {
                    nicid 0
                    type virtual
                    port_id a8201c3e-2c6e-4313-94d0-1b4eee14f08a
                    network 943fda9e-79f8-400c-b442-3506f102721a
                    subnet e313b95c-ca1f-4c81-8d60-c9e721a85d0b
                    ip_address 192.168.0.220
                    mac_address fa:16:3e:eb:bd:77
                    netmask 255.255.255.0
                    gateway 192.168.0.1
                }
            }
        }
        vm_target {
        }
        event {
            type VM_UPDATED
        }
```

```
    }
}
```

アタッチされたボリューム

```
notification {
    eventTime 2016-11-23T19:54:48.105+00:00
        status_message Attached 1 volume: [Volume=test-esc-2,volid=0]
        depname dep
        tenant test
        tenant_id 9132cc90b8324a1c95a6c00975af6206
        depid eb4fe3b5-138d-41a3-b6ff-d6fa9035ca6c
        vm_group Group1
        vm_source {
            vmid cd4eeb61-61db-45a6-9da1-793be08c4de6
            hostid 8e96b8830d7bfbb337ce665586210fcca9644cbe238240e207350735
            hostname my-server-5
            software_version 1.1
            interfaces {
                interface {
                    nicid 0
                    type virtual
                    port_id 26412180-45cf-4f0b-ab45-d05bb7ca7091
                    network 943fda9e-79f8-400c-b442-3506f102721a
                    subnet e313b95c-ca1f-4c81-8d60-c9e721a85d0b
                    ip_address 192.168.0.56
                    mac_address fa:16:3e:18:90:1e
                    netmask 255.255.255.0
                    gateway 192.168.0.1
                }
            }
            volumes {
                volume {
                    display_name test-esc-2__v0_0_0_1
                    external_id bf5c9a01-e9fb-42fa-89ee-73699d6c519c
                    bus virtio
                    type lvm
                    size 2
                }
            }
        }
        vm_target {
        }
        event {
            type VM_UPDATED
        }
    }
}
```

更新されたソフトウェアバージョン

```
notification {
    eventTime 2016-11-23T20:06:56.75+00:00
    escEvent {
        status SUCCESS
        status_code 200
        status_message VM Software Updated. VM name:
[dep_Group1_0_c9edef63-4d9d-43ea-af1b-16527ed2edae], previous version: [1.0], current
version: [1.1]
        depname dep
        tenant test
        tenant_id 9132cc90b8324a1c95a6c00975af6206
        depid eb4fe3b5-138d-41a3-b6ff-d6fa9035ca6c
```

```
        vm_group Group1
        vm_source {
            vmid cd4eeb61-61db-45a6-9da1-793be08c4de6
            hostid 8e96b8830d7bfbb337ce665586210fcca9644cbe238240e207350735
            hostname my-server-5
            software_version 1.1
            interfaces {
                interface {
                    nicid 0
                    type virtual
                    port_id 26412180-45cf-4f0b-ab45-d05bb7ca7091
                    network 943fda9e-79f8-400c-b442-3506f102721a
                    subnet e313b95c-ca1f-4c81-8d60-c9e721a85d0b
                    ip_address 192.168.0.56
                    mac_address fa:16:3e:18:90:1e
                    netmask 255.255.255.0
                    gateway 192.168.0.1
                }
            }
            volumes {
                volume {
                    display_name test-esc-2__v0_0_0_1
                    external_id bf5c9a01-e9fb-42fa-89ee-73699d6c519c
                    bus virtio
                    type lvm
                    size 2
                }
            }
        }
        vm_target {
        }
        event {
            type VM_SOFTWARE_VERSION_UPDATED
        }
    }
}
```

更新されたサービス

```
notification {
    eventTime 2016-11-23T20:06:56.768+00:00
    escEvent {
        status SUCCESS
        status_code 200
        status_message Service group update completed successfully
        depname dep
        tenant test
        tenant_id 9132cc90b8324a1c95a6c00975af6206
        depid eb4fe3b5-138d-41a3-b6ff-d6fa9035ca6c
        vm_source {
        }
        vm_target {
        }
        event {
            type SERVICE_UPDATED
        }
    }
}
```

展開内の VNF のアップグレード

ESC では、次のライフサイクルステージのいずれかで、既存の展開の VNF ソフトウェアをアップグレードできます。

- LCS：PRE SOFTWARE UPGRADE-SCRIPT ACTION
- LCS：POST SOFTWARE UPGRADE-SCRIPT ACTION

NB は、PRE、POST、または BOTH を使用してカスタム アクション スクリプトを実行することを選択できます。

カスタムスクリプトの詳細については、スクリプトアクション（188 ページ）のカスタムスクリプトを参照してください。ライフサイクルステージについては、さまざまなステージで定義されているライフサイクルステージ（LCS）ポリシーの条件 （203 ページ）を参照してください。

LCS_NOTIFY 通知は、ライフサイクルの各ステージでオンまたはオフにできます。software_version の変更では、各 VM の最終通知は VM_SOFTWARE_VERSION_UPDATED となります。ESC は、展開の更新ごとに SERVICE_UPDATED 通知を受信します。

ESC は、既存の展開で次の VNF ソフトウェア アップグレード シナリオをサポートします。

- 展開後の VNF のアップグレード
- 既存の展開での VNF 展開とアプリケーションのアップグレード

既存の展開内で他のリソースを更新する方法の詳細については、既存の展開の更新（225 ページ）を参照してください。

展開後の VNF のアップグレード

VNF のアップグレードは、単一または段階的なトランザクションで実行できます。

ESC は、単一のトランザクションで LCS ポリシーを追加し、ソフトウェアバージョンを変更します。

2 段階のトランザクションでは、ESC は最初のトランザクションで LCS ポリシーを追加し、2 番目のトランザクションでソフトウェアバージョンの変更を伴うソフトウェアアップグレードをトリガーします。

通知

- LCS_NOTIFY—LCS::DEPLOY_UPDATE::PRE_VM_SOFTWARE_VERSION_UPDATE
- LCS_NOTIFY—LCS::DEPLOY_UPDATE::POST_VM_SOFTWARE_VERSION_UPDATED
- VM_SOFTWARE_VERSION_UPDATED
- SERVICE_UPDATED

エラー

ESC は、VNF アップグレードプロセスの早期検証を実行します。カスタムスクリプトファイルが存在しない場合は、エラーが発生します。トランザクションは拒否され、通知はNFVOに送信されません。

カスタムスクリプトがタイムアウトすると、エラーが発生します。次の通知がNFVOに送信されます。

- LCS::DEPLOY_UPDATE::PRE_VM_SOFTWARE_VERSION_UPDATE
- LCS::DEPLOY_UPDATE::PRE_VM_SOFTWARE_VERSION_UPDATE
- VM_SOFTWARE_VERSION_UPDATED
- SERVICE_UPDATED

既存の展開での VNF 展開とアプリケーションのアップグレード

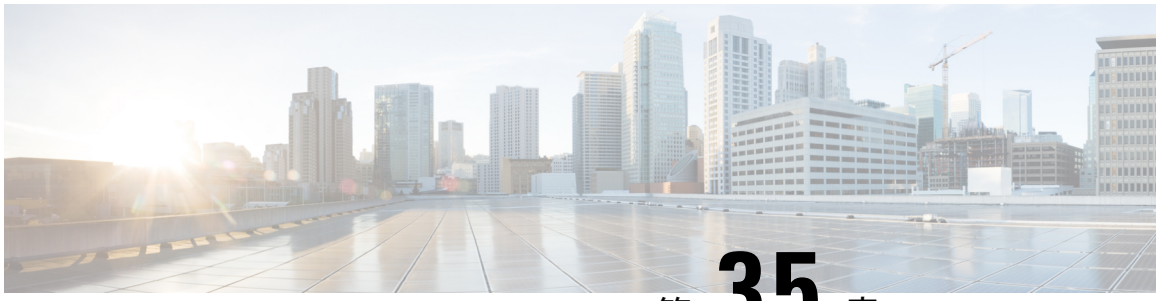
VNF の展開およびアプリケーションのアップグレード中に、ESC は次の通知を NFVO に送信します。

- VM_DEPLOYED
- LCS_NOTIFY-LCS::DEPLOY::POST_VM_ALIVE
- VM_ALIVE
- SERVICE_ALIVE

エラー

カスタムスクリプトがタイムアウトすると、エラーが発生します。次の通知がNFVOに送信されます。

- VM_DEPLOYED
- LCS::VM::POST_VM_ALIVE
- VM_DEPLOYED
- SERVICE_ALIVE



第 35 章

仮想ネットワーク機能の操作

VNF 操作

VNF を起動、停止、および再起動できます。起動、停止、および再起動の操作は、RESTful インターフェイスを使用して実行されます。

VNF 操作にはペイロードが必要です。

```
POST ESCManager/v0/{internal_tenant_id}/deployments/service/{internal_deployment_id}
```

例、

```
<?xml version='1.0' encoding='UTF-8'?>
<service_operation xmlns='urn:ietf:params:xml:ns:netconf:base:1.0'>
   <operation>stop</operation>
</service_operation>
```

操作フィールドで起動、停止、または再起動を指定する必要があります。

- VNF の起動：すべての VM が起動し、モニタリングが有効になり、KPI の詳細に従ってしきい値が再割り当てされます。VM の実行が開始され、VM_ALIVE_STATE に移動します。サービスは service_active_state になります。VNF の起動ワークフローを中断できるのは展開解除のみです。
- VNF の停止：サービスが停止すると、モニタリングが無効になり、すべての VM サービスが停止します。VM は使用できなくなります。サービスは service_stopped_state になります。VM は shutoff_state になります。リカバリ、スケールアウト、スケールインを実行することはできません。VNF の展開解除のみ可能です。
- VNF の再起動：モニタリングが無効になり、すべての VM が再起動します。つまり、VM が停止してから OpenStack で起動し、モニタリングが有効になり、KPI の詳細に従ってしきい値が再割り当てされます。VM は VM_ALIVE_STATE となり、サービスは service_alive_state です。再起動操作を中断できるのは展開解除のみです。

すでに実行中の VNF のモニタリングを開始することはできません。再起動後、VM に再度ログインすると、再起動、更新、およびモニタリングの詳細が示されます。また、リカバリも示す必要があります。

VM の操作

VNF 操作と同様に、個々の VM を起動、停止、および再起動できます。

VM 操作にはペイロードが必要です。

```
POST ESCManager/v0/{internal_tenant_id}/deployments/vm/{vm_name}
```

例、

```
<?xml version='1.0' encoding='UTF-8'?>
<vm_operation xmlns='urn:ietf:params:xml:ns:netconf:base:1.0'>
   <operation>stop</operation>
   <force>true/false</force>
</vm_operation>
```

操作フィールドで起動、停止、または再起動を指定する必要があります。

VNF バックアップおよび復元操作

ここでは、VM スナップショットを使用した VNF のバックアップおよび復元操作について説明します。

VNF バックアップ操作

VM スナップショットの管理

ESC は、OpenStack VIM 上にイメージ（特定の状況ではボリューム）であるスナップショットを作成します。ESC API は、ESC によって管理される VNF のスナップショットを管理します。ESC は、次の 3 つの主なスナップショット操作をサポートします。

- VM スナップショットの作成
- VM スナップショットの一覧表示
- VM スナップショットの削除

ESC は、HTTP および HTTPS プロトコルで ESC REST API を使用して VM スナップショット操作を実行します。VM スナップショットの作成および削除操作は、`esc_nc_cli` スクリプトによってサポートされます。NETCONF 通知と REST API 通知はどちらも、作成および削除操作に対するスナップショット操作のさまざまな段階で生成されます。

（注） VM スナップショット操作は、OpenStack VIM でのみサポートされています。

VM スナップショットの作成

ESC VM によって管理される任意の VNF から（REST API または esc_nc_cli スクリプトを使用して）スナップショットを作成できます。スナップショットは、（ESC VM ステータスが VM_ALIVE または VM_STOPPED に変換される）アクティブな VNF または停止した VNFに対してのみ作成できます。API 呼び出しのペイロードでスナップショット名を指定できます。スナップショット名が一意でない場合、ESC スナップショット操作ペイロードを指定するときに参照として使用されるスナップショット名とともに一意の ID が生成されます。OpenStack 上にスナップショット（イメージ）が作成されます。ブート可能ボリュームを使用する VNF の場合、ボリュームスナップショットも OpenStack に作成されます。

REST API を使用したスナップショットの作成

スナップショットを作成するには、ESCManager API に HTTP POST 操作を指定します。

```
POST: /ESCManager/v0/<tenant-id>/deployments/snapshot-vm/<generated-vm-name>
```

ペイロードには operation と name の値が含まれている必要があり、操作の値は **snapshot** である必要があります。

```
operation: snapshot
name: <snapshot-name>
```

成功すると、HTTP 200 コードが返され、ペイロードはありません。

失敗した場合（検証エラーまたは OpenStack API エラー）、適切な HTTP エラーコードとエラーメッセージが返されます。

以下は、スナップショットを作成するための API 呼び出しを示しています。

```
[admin@localhost]$ cat snapshot.json
{
    "operation": "snapshot",
    "name": "my-snapshot-name"
}

[admin@localhost]$ curl -X POST -d @snapshot.json -H 'Content-Type: application/json'
-H 'callback: http://localhost:9009' -H 'Callback-ESC-Events: http://localhost:9009'
"http://localhost:8080/ESCManager/v0/snapshot-tenant/deployments/snapshot-vm/new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba"
```

esc_nc_cli スクリプトを使用したスナップショットの作成

esc_nc_cli スクリプトを使用してスナップショットを作成するには、生成された VM の名前と操作を指定する固定パラメータを渡します。

```
VM Backup Action : vm-backup-action vm-name backup-name [<action-type>] [<xmlfile>]
    action-type := SNAPSHOT|EXPORT
```

オプションの action-type パラメータは、指定されていない場合、デフォルトで SNAPSHOT になります。以下は、スナップショットを作成するためのスクリプトの呼び出しを示しています。

```
[admin@localhost]$ esc_nc_cli vm-backup-action
new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba my-snapshot-name SNAPSHOT
VM Backup Action
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=esc-nc-admin
 --privKeyFile=/home/admin/.ssh/confd_id_rsa --privKeyType=rsa
--rpc=/tmp/tmp_esc_nc_cli.c8d9kAjcGf
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <ok/>
</rpc-reply>
```

成功した場合、単一の <ok/> 要素を含む XML ペイロードが返されます。失敗した場合（検証エラーまたは OpenStack API エラー）、適切なエラーメッセージが返されます。

注記

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

- スナップショット名の長さは 255 文字以下にする必要があります。
- 生成された VM 名は有効である必要があります。
- action-type は SNAPSHOT または EXPORT（`esc_nc_cli` のみ）である必要があります。
- xmlfile：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）。

[Notifications]

スナップショットの作成操作中に、NETCONF 通知と ESC REST コールバックメッセージの両方が送信されます。

表 22：

通知（**NETCONF** または **ESC** コールバック）	通知が送信された場合
VM_BACKUP_INIT	API が呼び出され、検証に合格した場合。
VM_BACKUP_CREATED	OpenStack がスナップショットの作成要求を正常に受信して検証した場合。
VM_BACKUP_COMPLETE	OpenStack がスナップショットの作成要求操作を完了し、成功したか、エラーが発生した場合。

次に、VM_BACKUP_CREATED の成功した NETCONF 通知の例を示します（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>2021-09-14T12:18:39.836+00:00</eventTime>
    <escEvent xmlns="http://www.cisco.com/esc/esc">
        <status>SUCCESS</status>
        <status_code>202</status_code>
        <status_message>Snapshot is now active.</status_message>
        <depname>snapshot-deployment-name</depname>
        <tenant>snapshot-tenant</tenant>
        <tenant_id>7d61b5de73874f88a458d486759a9b83</tenant_id>
        <depid>ae0bea05-9630-4d17-a9e7-926f1f625dc7</depid>
        <vm_group>snapshot-group</vm_group>
        <vm_source>
            <vmid>1773914c-20cd-4f50-b337-1e46be2cf295</vmid>
```

```
            <vmname>new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba</vmname>


<generated_vmname>new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba</generated_vmname>

            <vim_id>default_openstack_vim</vim_id>
            <vim_project>snapshot-tenant</vim_project>
            <vim_project_id>7d61b5de73874f88a458d486759a9b83</vim_project_id>
            <hostid>95503baadeccce2d33e5d924322390aee9d30c6ed24043284bf46984</hostid>
            <hostname>pf-ucs-27</hostname>
        </vm_source>
        <event>
            <type>VM_BACKUP_CREATED</type>
        </event>
    </escEvent>
</notification>
```

失敗した場合、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 FAILURE になります。
- <status_code> は 500 になります。
- <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になります。

スナップショットの一覧表示

ESC REST API を使用してスナップショットを一覧表示できます。ESC で管理しているスナップショットのみ一覧表示できます。スナップショットデータのサブセットをクエリパラメータとして指定して、返されるスナップショットの数を減らすことができます。返されるスナップショットデータは、HTTP Accept ヘッダーによって制御される XML または JSON 形式にできます。Accept ヘッダーの値は、指定されていない場合、デフォルトで XML になります。

ESC REST API のみがスナップショットの一覧表示をサポートしています。`esc_nc_cli` は、ESC 管理対象エンティティの一覧表示をサポートしていません。

ESC REST API を使用したスナップショットの一覧表示

スナップショットを一覧表示するには、ESCManager API に対して HTTP GET 操作を指定できます。

```
GET: /ESCManager/v0/snapshots
```

オプションのクエリパラメータ（internalTenantId、 generatedVMName）も指定できます。

返されるスナップショットの数に関係なく、常に HTTP 200 コードが返されます。

以下は、特定の内部テナント ID と生成された VM 名のスナップショットを一覧表示する API 呼び出しを示しています。

```
[admin@localhost]$ curl -X GET --header "Accept: application/xml"
"http://localhost:808/ESCManager/v0/snapshots?internalTenantId=snapshot-tenant&generatedVMName=new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba"
 | xmllint --format -

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<snapshots>
    <snapshot xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
        <id>7813c20b-94b6-492b-ae74-0bd36c1168dc</id>
        <name>my-snapshot-name</name>
        <creation_start_date>2021-07-20T11:26:47.532Z</creation_start_date>
        <creation_end_date>2021-07-20T11:27:53.139Z</creation_end_date>
        <status>available</status>
        <status_message>Snapshot image for VM [gen_vm_name] is active.</status_message>

<gen_vm_name>new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba</created_from_generated_vm_name>

        <vim_id>default_openstack_vim</vim_id>
        <tenant>snapshot-tenant</tenant>
        <bootable_volume_snapshot_id:c3cd5d13-63bf-49f0-b864-df3bc024d5e4/>
    </snapshot>
</snapshot>
```

（注） この API 呼び出し時に生成される通知はありません。

スナップショットの削除

ESC で作成したスナップショットは、REST API または esc_nc_cli スクリプトを使用して削除できます。削除できるのは現在の ESC VM によって管理されているスナップショットのみで、一度に削除できるスナップショットは 1 つだけです。成功した場合、スナップショットは ESC から削除され、OpenStack 内の関連するイメージとボリュームのスナップショット（ある場合）も削除されます。

REST API を使用したスナップショットの削除

ESC を介して以前に作成されたスナップショットを削除するには、ESCManager API に対して HTTP DELETE 操作を指定できます。

```
DELETE: /ESCManager/v0/snapshots/<snapshot-id|snapshot-name>
```

スナップショット ID またはスナップショット名を渡すことができます。成功すると、HTTP 200 コードが返され、ペイロードはありません。失敗した場合（検証エラーまたは OpenStack API エラー）、適切な HTTP エラーコードとエラーメッセージが返されます。以下は、スナップショットを削除するための API 呼び出しを示しています。

```
[admin@localhost]$ curl -X DELETE -H 'callback: http://localhost:9009' -H
'Callback-ESC-Events: http://localhost:9009'
"http://localhost:8080/ESCManager/v0/snapshots/7813c20b-94b6-492b-ae74-0bd36c1168dc"
```

esc_nc_cli を使用したスナップショットの削除

esc_nc_cli スクリプトを使用してスナップショットを削除するには、スナップショット ID またはスナップショット名のみを単一のパラメータとして渡す必要があります。

```
Snapshot Action : snapshot-action <snapshot-id|snapshot-name>
```

以下は、スナップショットを作成するためのスクリプトの呼び出しを示しています。

```
[admin@localhost]$ esc_nc_cli snapshot-action delete my-snapshot-name
```

または

```
[admin@localhost]$ esc_nc_cli snapshot-action delete my-snapshot-name-1
<?xml version="1.0" encoding="UTF-8"?>
<error xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
    <error_code>404</error_code>
    <error_message>Snapshot image [my-snapshot-name-1] not found.</error_message>
</error>
```

成功した場合、単一の <ok/> 要素を含む XML ペイロードが返されます。失敗した場合（検証エラーまたは OpenStack API エラー）、適切なエラーメッセージが返されます。

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

- スナップショット ID またはスナップショット名は有効である必要があります。
- スナップショット名を指定する場合は、一意の名前にする必要があります。

通知

スナップショットの削除操作中に、NETCONF 通知と ESC REST コールバックメッセージの両方が送信されます。

通知は次のとおりです。

表 *23：*

通知（**NETCONF** や **ESC** コールバック）	通知の送信タイミング
VM_SNAPSHOT_DELETING	送信と検証が成功した場合。
VM_SNAPSHOT_DELETED	OpenStack がスナップショットの削除操作を完了し、成功したか、エラーが発生した場合。 スナップショットにイメージスナップショットとともにボリュームスナップショットがある場合、両方が削除されるまで通知は送信されません。

次に、VM_SNAPSHOT_DELETED の成功した NETCONF 通知の例を示します（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>2021-09-14T12:18:39.836+00:00</eventTime>
    <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Snapshot image [2ffadd36-3b41-4c13-a9d6-a48c07764d1a] has been
deleted.</status_message>
    <depname>snapshot-deployment-name</depname>
    <tenant>snapshot-tenant</tenant>
    <tenant_id>7d61b5de73874f88a458d486759a9b83</tenant_id>
    <depid>ae0bea05-9630-4d17-a9e7-926f1f625dc7</depid>
    <vm_group>snapshot-group</vm_group>
    <vm_source>
        <vmid>1773914c-20cd-4f50-b337-1e46be2cf295</vmid>
        <vmname>new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba</vmname>

<gen_vm_name>new-deployment-n_new-gr_0_af0148e2-e74c-4be7-b8c1-49bd53def6ba</generated_vmname>

        <vim_id>default_openstack_vim</vim_id>
```

```
        <vim_project>snapshot-tenant</vim_project>
        <vim_project_id>7d61b5de73874f88a458d486759a9b83</vim_project_id>
        <hostid>95503baadeccce2d33e5d924322390aee9d30c6ed24043284bf46984</hostid>
        <hostname>pf-ucs-27</hostname>
    </vm_source>
    <event>
        <type>VM_SNAPSHOT_DELETED</type>
    </event>
    </escEvent>
</notification>
```

失敗した場合、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 FAILURE になります。
- <status_code> は 500 になります。
- <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になります。

VM スナップショット ポーリング パラメータ

VIM Manager の構成プロパティを使用して、OpenStack への呼び出しの間隔を制御して、作成操作と削除操作のステータスを確認するとともに、操作がエラーでタイムアウトするまでの最大許容時間（分）を制御できます。

次のプロパティがこれに該当します。

- vim.asyncpoller.snapshot.create.poll.secs # default 15、ポーリング間の秒数
- vim.asyncpoller.snapshot.create.timeout.mins # default 20、スナップショット作成操作の最大分数
- vim.asyncpoller.snapshot.delete.poll.secs # default 15、ポーリング間の秒数
- vim.asyncpoller.snapshot.delete.timeout.mins # default 20、スナップショット削除操作の最大分数

デフォルト値は、以下に示すように、/opt/cisco/esc/vimmanager/application.properties の下にある application.properties ファイルに設定し、VIM Manager サービスを再起動することでオーバーライドできます。

```
[admin@localhost]$ sudo cat /opt/cisco/esc/vimmanager/application.properties
vim.asyncpoller.snapshot.create.poll.secs=5
vim.asyncpoller.snapshot.create.timeout.mins=10
vim.asyncpoller.snapshot.delete.poll.secs=10
vim.asyncpoller.snapshot.delete.timeout.mins=60

[admin@localhost]$ sudo escadm vimmanager restart
Stopping vimmanager service: [OK]
Starting vimmanager service: [OK]

[admin@localhost]$ sudo escadm vimmanager show
VimManager System Configurations.
{
    "ccp.pollRetries": "200",
    "ccp.pollRetryDelaySecs": "15",
```

```
    . . .

    "vim.asyncpoller.snapshot.create.poll.secs": "5",
    "vim.asyncpoller.snapshot.create.mins": "10",
    "vim.asyncpoller.snapshot.delete.poll.secs": "10",
    "vim.asyncpoller.snapshot.delete.timeout.mins": "60",

    . . .

    "vmware.ovftool.params": "--acceptAllEulas --disableVerification --noSSLVerify
--allowExtraConfig",
    "vmware.powerOnRetry": "8"
}
```

HAセットアップでは、アプリケーションプロパティファイルを両方のノードにコピーする必要があります。

または、escadmスクリプトを使用して、値を動的に設定できます（ただし、それらの値は再起動後は保持されません）。

```
[admin@loclhost] sudo escadm vimmanager set --config
vim.asyncpoller.snapshot.create.poll.secs=200
vim.asyncpoller.snapshot.create.timeout.mins=1
VimManager configuration [vim.asyncpoller.snapshot.create.poll.secs] has updated to
[200].
VimManager configuration [vim.asyncpoller.snapshot.create.timeout.mins] has updated to
[1].
```

ブート可能ボリュームのある VNF のスナップショット

ブートボリュームのある ESC 管理対象 VNF のスナップショットが作成された場合、イメージスナップショットとボリュームスナップショットの両方が OpenStack 内に作成されます。

（注） イメージスナップショット名は、スナップショットペイロードで指定されたスナップショット名になります。ボリュームのスナップショット名（該当する場合）は、スナップショットの前に付加されます。

たとえば、ブート可能ボリュームのある VNF の ESC VM でスナップショットが作成され、そのスナップショットにmy-snapshot-nameという名前が付けられた場合、次のことが当てはまります。

```
[admin@localhost]$ openstack volume snapshot list | grep my-snapshot-name
| 52a96891-f22d-4863-bb47-bd9442ca0cb1 | snapshot for my-snapshot-name | None | available
 | 2 |

[admin@localhost]$ openstack image list | grep my-snapshot-name
| c8846c14-48e4-45db-88a0-f838fc3ac29d | my-snapshot-name | active |
```

ボリュームスナップショットは、ESC 内で直接使用することも、OpenStack でネイティブに復元操作で使用することもできません。最初にスナップショットからブート可能ボリュームを作成する必要があります。ESCは、ボリュームスナップショットからのブート可能ボリュームの作成をサポートしています。詳細については、「VNF 復元操作」を参照してください。

VNF 復元操作

ブート可能ボリュームのない VNF のスナップショット

ESC がブート不能ボリュームを使用して VNF のスナップショットを取得した場合、そのスナップショットは OpenStack にスナップショットイメージとして保存されます。そのスナップショットイメージは、ESC のサービス更新機能を介して復元するために使用できます。サービス更新 XML は、元の展開 XML と同じように作成できますが、スナップショットイメージ名を使用します。このサービス更新 XML が REST または `esc_nc_cli` インターフェイスを使用して ESC に展開されると、ESC は VNF のイメージ名を内部的に更新し、次の再展開時に（通常は REST または `esc_nc_cli` インターフェイスを使用して手動でトリガーされます）、新しいイメージを使用して新しい展開が作成されます。

ブート可能ボリュームのある VNF のスナップショット

ESC がブート可能ボリュームを使用して VNF のスナップショットを作成すると、そのスナップショットは、スナップショットイメージとボリュームスナップショットの両方として OpenStack に保存されます。ボリュームスナップショットは、ESC または OpenStack では、VNF の復元プロセス内で直接使用できません。ブート可能ボリュームは、ESC の観点からアウトオブバンドと見なされる（つまり、ESC によって直接管理されない）ボリュームスナップショットから最初に作成する必要があります。ブート可能ボリュームが作成されて利用可能になると、ESC のサービス更新機能を介して復元するために使用できます。サービス更新 XML は、元の展開 XML と同じように作成できますが、指定された元のボリュームに対する削除操作と、新しいブート可能ボリュームを指定する作成操作を実行します。このサービス更新 XML が REST または `esc_nc_cli` インターフェイスを使用して ESC に展開されると、ESC は、VNF を再展開することなく、元のボリュームを新しいボリュームに自動的にスワップアウトするため、すべての OpenStack UUID とリソースが保持されます。

元のボリューム（VNF から切り離されたボリューム）は引き続き OpenStack に残るため、手動でクリーンアップする必要があります。

ボリュームスナップショットからのブート可能ボリュームの作成

ESC がブート可能ボリュームのある VNF のスナップショットを取得すると、OpenStack でボリュームのスナップショットが作成され、ESC を使用してボリュームスナップショットからブート可能ボリュームを作成できます。ESC 内で、または OpenStack API を直接使用して、復元操作を実行するには、ボリュームスナップショットからブート可能ボリュームを作成する必要があります。ボリューム名は、ESC REST API 呼び出しのペイロードで指定できます。`esc_nc_cli` スクリプトは、ボリュームスナップショットからのブート可能ボリュームの作成をサポートしていません。ボリューム名は一意である必要はありません。ESC 復元操作のペイロードを指定するときに参照として使用できる名前と一緒に一意の ID が生成されます。ボリュームスナップショットから正常に作成されたボリュームは最終的に、OpenStack で新しいブート可能ボリュームになります。

（注） OpenStack 上の新しいブート可能ボリュームは、ESC によって管理されません。このボリュームはアウトオブバンドであり、Orchestrator で直接管理する必要があります。

REST API を使用したボリュームスナップショットからのブート可能ボリュームの作成

ボリュームスナップショットからボリュームを作成するには、ESCManager API のスナップショット エンドポイントに対して HTTP POST 操作を指定できます。

```
POST: /ESCManager/v0/snapshots/<snapshot-id>/volumes
```

ペイロードには、新しいボリュームの名前となる name の値が含まれている必要がありますが、オプションで volume_type、multiattach、および bootable を指定できます。

```
operation: snapshot
name: <snapshot-name>
volume_type: <valid-volume-type>   # defaults to the OpenStack default volume type
multiattach: <true|false>          # defaults to false
bootable: <true|false>             # defaults to true
```

成功すると、操作が正常に OpenStack に送信されたことを示す HTTP 202 コードが返され、ペイロードはありません。失敗した場合（検証エラーまたは OpenStack API エラー）、適切な HTTP エラーコードとエラーメッセージが返されます。

以下は、スナップショット ID を決定するために、最初に ESC 内で一覧表示した後、スナップショットを作成する API 呼び出しを示しています。

```
[admin@localhost]$ curl -s
"http://localhost:8080/ESCManager/v0/snapshots?internalTenantId=dave-2000" | xmllint
--format -
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<snapshots>
    <snapshot xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <id>171ffa7d-8318-47d1-acab-b01db4501a39</id>
        <name>my-snapshot-name</name>
        <creation_start_date>2021-09-20T08:29:22.074+01:00</creation_start_date>
        <creation_end_date>2021-09-20T08:33:41.193+01:00</creation_end_date>
        <status>active</status>
        <status_message>Snapshot image for
[new-dep-4_new-gr_0_34b9da8a-af64-4452-a8a3-8972e23e4e98] is active.</status_message>

<created_from_generated_vm_name>new-dep-4_new-gr_0_34b9da8a-af64-4452-a8a3-8972e23e4e98</created_from_generated_vm_name>

        <vim_id>my-snapshot-vim</vim_id>
        <tenant>dave-2000</tenant>
        <volume_snapshot_id>c4548ba4-0480-4b42-8229-ad98de44b3ea</volume_snapshot_id>
    </snapshot>
</snapshots>

[admin@localhost]$ cat volume_from_volume_snapshot.json
{
    "name": "my-bootable-volume-from-snapshot-volume",
    "multiattach": true
}

[admin@localhost]$ curl -X POST -d @volume_from_volume_snapshot.json -H 'Content-Type:
application/json' -H 'callback: http://localhost:9009' -H 'Callback-ESC-Events:
http://localhost:9009'
"http://localhost:8080/ESCManager/v0/snapshots/171ffa7d-8318-47d1-acab-b01db4501a39/volumes"
```

```
[admin@localhost]$ openstack volume list | grep my-bootable-volume
| c4548ba4-0480-4b42-8229-ad98de44b3ea | my-bootable-volume-from-snapshot-volume |
available | 2 | |
```

ESC REST API では、次の内容が検証されます。

- ボリューム名の長さは 255 文字以下にする必要があります。
- スナップショット ID は、ESC 管理対象スナップショットの ID である必要があります（つまり、list snapshot 操作が返すスナップショットの 1 つである必要があります）。
- name はペイロードで指定する必要があります。他の属性はすべてオプションで設定できます。
- ペイロードでサポートされていない属性名は無視されます。
- スナップショットは、ブート可能ボリュームがある VNF に対して取得されている必要があります。

[Notifications]

この操作では、ESC REST コールバックメッセージのみが生成されます。

VM_VOLUME_ACCEPTED_EVENT と VM_VOLUME_CREATED_EVENT の 2 つのコールバックメッセージが生成されます。

VM_VOLUME_CREATED_EVENT ESC REST コールバックメッセージの例を次に示します。

```
{
    "escTransactionId": "5acac790-9213-45c0-8fde-9dd7d3111fdb",
    "eventType": "VM_VOLUME_CREATED_EVENT",
    "eventSourceContext": null,
    "eventTargetContext": null,
    "message": "Create volume snaphot request completed",
    "stateMachineEventNBInfo": {
        "id": "de9440c0-1342-441d-a16e-5c8267231ae5",
        "message": {},
        "logNames": [],
        "keywords": {},
        "actionInfo": {},
        "stackTrace": ""
    },
    "escParameter": {
        "external_volume_id": "c3cd5d13-63bf-49f0-b864-df3bc024d5e4",
        "size": "2",
        "sizeunit": null,
        "bus": "virtio",
        "type": "LVM",
        "outOfBand": "false",
        "bootIndex": null,
        "name": "daves-ooband-bootable-volume-for-restore",
        "format": null,
        "deviceType": null,
        "storageLocation": null,
        "external_tenant_id": null,
        "internal_tenant_id": null,
        "internal_volume_id": null,
        "volid": null,
        "event_type": null,
        "image": null
    },
```

```
    "vmUpdateType": null,
    "requestDetails": null,
    "statusCode": "201",
    "notificationOnlyEvent": false
}
```

ポーリング設定パラメータ

VIM Manager の構成プロパティを使用して、OpenStack への呼び出しの間隔を制御して、ボリューム作成操作のステータスを確認するとともに、操作がエラーでタイムアウトするまでの最大許容時間（分）を制御できます。

次のプロパティがこれに該当します。

- vim.asyncpoller.volume.create.poll.secs # default 15、ポーリング間の秒数
- vim.asyncpoller.volume.create.timeout.mins # default 20、ボリューム作成操作の最大分数

```
[admin@localhost]$ sudo cat /opt/cisco/esc/vimmanager/application.properties
vim.asyncpoller.volume.create.poll.secs=5
vim.asyncpoller.volume.create.timeout.mins=10

[admin@localhost]$ sudo escadm vimmanager restart
Stopping vimmanager service: [OK]
Starting vimmanager service: [OK]

[admin@localhost]$ sudo escadm vimmanager show
VimManager System Configurations.
{
    "ccp.pollRetries": "200",
    "ccp.pollRetryDelaySecs": "15",

    . . .

    "vim.asyncpoller.volume.create.poll.secs": "5",
    "vim.asyncpoller.volume.create.mins": "10",

    . . .

    "vmware.ovftool.params": "--acceptAllEulas --disableVerification --noSSLVerify
--allowExtraConfig",
    "vmware.powerOnRetry": "8"
}
```

（注） HAセットアップでは、アプリケーションプロパティファイルを両方のノードにコピーする必要があります。

個々の VNF と複合 VNF の管理

個々のサービスは単一の VNF で構成されます。連携サービスまたは複合 VNF は、異なるタイプの複数の VM で構成されます。ESC インターフェイスは、ノースバウンドシステムから VM 相互依存情報を受信し、VM および VNF の作成中、およびライフサイクル管理中にこの情報

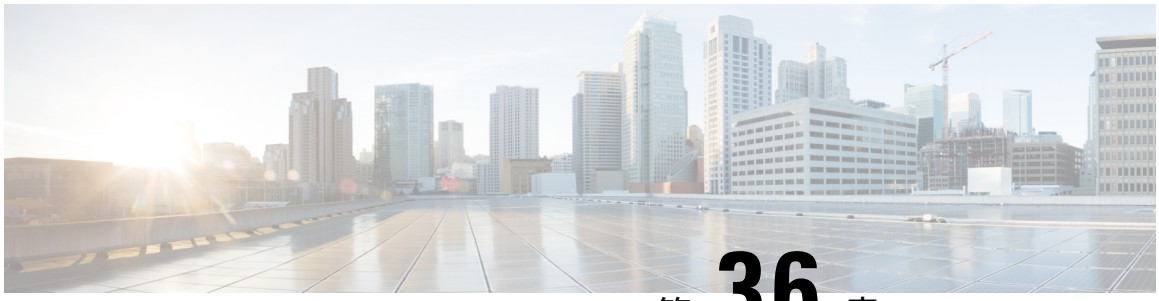
を使用します。相互依存性には、単一 VNF 内の VM グループ、VNF モニタリング、拡張性などの VM 固有のワークフローが含まれます。

VM では、作成、読み取り、更新、および削除の操作が許可されます。静的IPを使用して展開済み VNF に VM インスタンスを追加するには、追加の IP アドレスを静的 IP プールに提供する必要があります。既存の静的IP展開を使用している場合は、VMの最小数が変更されます。

新しい最小値（VMの数）がアクティブなVMの数より大きい場合、新しいVMがサービスに追加されます。値が最大値よりも大きい場合、更新は拒否されます。

第 V 部

モニタリング、スケーリング、および修復



第 36 章

仮想ネットワーク機能のモニタリング

VNF のモニタリング

VNF は展開後、正常性とワークロードを確認するために定期的にモニタされます。モニタリングは、展開データモデルの KPI セクション内のメトリックの定義に基づいています。KPI セクションで説明されているように、メトリックタイプによって、モニタする変数だけでなく、実行するコレクタアクションも決まります。ESC では、モニタ対象のメトリックと、条件を満たしたときに実行する必要があるアクションを定義できます。それらのメトリックとアクションは、展開データモデルで定義されます。VNF をモニタするために、複数のモニタリング方法が使用されます。次の内容をモニタできます。

- VM の稼働状態
- ディスク使用率、メモリ、CPU、ネットワークスループットの VM 変数
- VM モニタリング インターフェイスの ICMP メッセージ

モニタリングの前提条件

ESC で VM をモニタするには、次の前提条件を満たしている必要があります。

- 正常に展開された VM のモニタリングが有効になっている。展開された VM は稼働している必要があります。
- KPI は、モニタリングパラメータを使用してデータモデル内で設定されている必要がある。

モニタリングおよびアクション実行エンジン

モニタリングは、展開データモデルの KPI セクション内のメトリックの定義に基づいています。KPIセクションで説明されているように、メトリックタイプによって、モニタする変数だけでなく、実行するコレクタアクションも決まります。モニタリングエンジンは、メトリックとアクションで構成されます。

1. メトリック
2. アクション

メトリックとアクション<metadata>セクションでは、エンジンのプログラム可能な側面を制御するプロパティまたはエントリについて記述します。

メトリックセクション

メトリックセクションは次のとおりです。

```
<metrics>
     <metric>
     <name>{metric name}name>
           <type>{metric type}type>
           <metaData>
           <type>{monitoring engine action type}</type>
         <properties>
         <property>
         <name></name>
        <value></value>
       </property>
       : : : : : : : :
      <properties/>
     </metaData>
    </metric or action>
     : : : : : : : : :
<metrics>
```

表 *24 :* メトリックセクションの説明

タグ名	説明	値
名前	ユーザ定義のメトリック名。メトリック名は一意である必要があります。	
タイプ	ダイナミックマッピングがサポートされるタイプ。	MONITOR_SUCCESS_FAILURE MONITOR_THRESHOLD MONITOR_COMPUTE_THRESHOLD

メトリック メタデータ セクション

メタデータセクションの目的は、モニタリングソリューションに固有の情報を提供することです。

表 *25 :* メトリック メタデータ セクション

タグ名	説明	値
タイプ	アクションタイプ、値は、MONA でサポートされるアクションと 1 対 1 でマッピングされます。	custom_script custom_script_threshold snmp_get_threshold
プロパティ	選択したアクションに渡されるプロパティ（名前/値）のリストのコンテナ。プロパティは、予期されるモニタリングおよびアクション属性のリストによって定義されます。	プロパティは、選択したアクションタイプに基づきます。

アクションセクション

アクションセクションは次のとおりです。

```
<actions>
           <action>
           <name>{action name}name>
           <type>{action type}type>
           <metaData>
           <type>{monitoring engine action type}</type>
           <properties>
           <property>
           <name></name>
           <value></value>
          </property>
           : : : : : : : :
           <properties/>
          </metaData>
         </action>
           : : : : : : : : :
          <actions>
```

表 *26 :* アクション

タグ名	説明	値
名前	ユーザ定義のアクション名。アクション名は一意である必要があります。	主な要件の 1 つは、選択された名前の先頭に TRUE または FALSE を付けて、MONITOR_SUCCESS_FAILURE のためだけに、ESC データモデルルールと動的アクション間のマッピングを許可することです。

タグ名	説明	値
タイプ	サポート対象タイプ。	ESC _POST_EVENT スクリプト CUSTOM_SCRIPT

アクション メタデータ セクション

メタデータセクションの目的は、モニタリングソリューションに固有の情報を提供することです。

表 *27*: アクション メタデータ セクション

タグ名	説明	値
タイプ	アクションタイプ、値は、モニタリングおよびアクションエンジンでサポートされるアクションと 1 対 1 でマッピングされます。	icmp_ping icmp4_ping icmp6_ping esc_post_event スクリプト custom_script snmp_get snmp_get_threshold
プロパティ	選択したアクションに渡されるプロパティ（名前/値）のリストのコンテナ。プロパティは、予期されるモニタリングおよびアクション属性のリストによって定義されます。	プロパティは、選択したアクションタイプに基づきます。

詳細については、「KPI、ルール、およびダイナミックマッピング API 」セクションを参照してください。

表 *28 :* サポートされているアクションタイプ

タイプ	プロパティと各プロパティの説明
icmp_ping	• ip_address • enable_events_after_success：MONA がイベント通知の転送を開始するタイミングを制御するブール値。true に設定すると、初めて成功に移行した後にのみ通知が転送されます。 • timeOut：デフォルトで 5 秒に設定
icmpv4_ping	• ip_address • enable_events_after_success：MONA がイベント通知の転送を開始するタイミングを制御するブール値。true に設定すると、初めて成功に移行した後にのみ通知が転送されます。 • timeOut：デフォルトで 5 秒に設定
icmpv6_ping	• ip_address • enable_events_after_success：MONA がイベント通知の転送を開始するタイミングを制御するブール値。true に設定すると、初めて成功に移行した後にのみ通知が転送されます。 • timeOut：デフォルトで 5 秒に設定
スクリプト	• script_filename：実行するスクリプトへのフルパス（スクリプトは ESC VM に配置する必要があります）。 • wait_for_script：アクションがスクリプトの完了を待機しているかどうかを制御するブール値（実際には実行されません）。
custom_script	script_filename：実行するスクリプトへのフルパス（スクリプトは ESC Manager VM に配置する必要があります）。

タイプ	プロパティと各プロパティの説明
custom_script_threshold	• script_filename：実行するスクリプトへのフルパス（スクリプトはESC Manager VMに配置する必要があります）。 • しきい値
post_esc_event	• esc_url • vm_external_id • vm_name • esc_event • event_name
snmp_get	• target_oid、agent_address、SNMP エージェントの IP アドレス（IPV4/IPV6 がサポートされます）。 • agent_port：SNMP エージェントで使用されるポート。 • agent_protocol：SNMP エージェントで使用されるプロトコル（tcp/udp）。 • Community：SNMP エージェントで使用される SNMP v2c コミュニティストリング。
snmp_get_threshold	• target_oid：しきい値の比較に使用されるオブジェクト識別子。 • agent_address：SNMP エージェントの IP アドレス（IPV4/IPV6 がサポートされます）。 • agent_port：SNMP エージェントで使用されるポート。 • agent_protocol：SNMP エージェントで使用されるプロトコル（tcp/udp）。 • community：SNMP エージェントで使用される SNMP v2c コミュニティストリング。

タイプ	プロパティと各プロパティの説明
snmp_get_threshold_ratio	• oid_total_value：比率/パーセンテージ計算の経過を表すために使用されるオブジェクト識別子。 • oid_current_value：比率/パーセンテージ計算の経過を表すために使用されるオブジェクト識別子。パーセンテージ/比率の計算に使用されるアルゴリズム。現在、COMPUTE_TOTAL_CURRENT_BASED と COMPUTE_TOTAL_AVAILABILITY_BASED の 2 つのアルゴリズムがサポートされています。 • agent_address：SNMP エージェントの IP アドレス（IPV4/IPV6 がサポートされます）。 • agent_port：SNMP エージェントで使用されるポート。 • agent_protocol：SNMP エージェントで使用されるプロトコル（tcp/udp）。 • community：SNMP エージェントで使用される SNMP v2c コミュニティストリング。

プロパティとランタイム パラメータ インジェクション

選択したアクションタイプに渡されるプロパティリストは、選択した一部のパラメータのランタイム値を自動的に挿入する機能をサポートします。たとえば、仮想マシン ip_address のランタイム値や仮想マシンの名前を、選択したアクションに引数として自動的に渡すことができます。

次に、実行時にスクリプトに渡すことができるパラメータの一部を示します。パラメータ値は、次の場合にのみ実行時に設定されます。

- パラメータがサポート対象のパラメータである。
- パラメータの値が、dynamic-mappings.xml ファイル内で空である。

それ以外の場合、スクリプト内で定義された値がそのまま渡されます。

次の表に、実行時に渡されるパラメータを示します。

esc_url	Elastic Services Controller の URL。
vm_external_id	管理対象 VM の外部 ID。

vm_name	管理対象 VM の名前。
vm_mac_address	管理対象 VM の MAC アドレス。
vm_external_host_id	VM 外部ホスト識別子。
vm_external_host_name	VM 外部ホスト名。
vm_group_name	VM グループ名。
ip_address	VM IP アドレス。
event_name	ESC イベント名。

（注） 選択したアクションに渡されるプロパティリストは、アクションタイプのパラメータによってバインドされていません。スクリプト設計者は、独自のパラメータを定義できます。ただし、値を指定する必要があります。

モニタリング方式

ESC は、いくつかのモニタリング方法を使用して VNF をモニタします。モニタリング方式の KPI データモデルを設定する必要があります。

ICMP ping モニタリング

ping モニタリングは、VNFの動作状態または到達可能性を評価します。

VM が到達不能の場合、VM の修復がトリガーされます。定義された間隔ごとに、ESC はメトリック値をポーリングし、必要に応じてアラームを送信します。ポーリング数、メトリック値、およびその他の設定は、KPI データモデルで設定されます。

SNMP モニタリング

SNMP モニタリングでは、特定の期間におけるメモリ使用率や CPU などの VM の負荷がモニタされます。SNMP Get 操作は、VNF の動作状態または到達可能性を評価するために使用されます。このモニタリング方式では、成功または失敗のみがモニタされます。

SNMP しきい値モニタリング

SNMP しきい値モニタリングでは、データモデルの KPI セクションで上限および下限しきい値レベルを設定できます。アクションは、しきい値の上限と下限に基づいて実行されます。

カスタムモニタリング

ESC 2.1 以前では、データモデルで定義されたアクションとメトリックをモニタリングエージェントで使用可能な有効なアクションとメトリックにマッピングするために、ダイナミックマッ

ピング XML が必要です。ファイルは ESC VM に保存され、テキストエディタを使用して変更されます。この方法はエラーが発生しやすく、アクティブ VM とスタンバイ VM の両方で HA ペアを変更する必要があります。ESC 2.2 以降には、esc-dynamic-mapping ディレクトリと dynamic_mappings.xml ファイルはありません。アクションとメトリックをマッピングするための CRUD 操作が、ESC の REST API を介して実行できるようになりました。詳細については、KPI、ルール、およびメトリック （181 ページ）を参照してください。

VM のモニタリング

Cisco Elastic Services Controller は、VM をモニタしてエラー状態を検出します。ESC はそのモニタリング方式の 1 つを使用して VM のアクションを検出し、この情報をルールサービスに渡して処理します。モニタリング要求は、VNF 展開要求とともにノースバウンドクライアントから送信されます。

データモデル xml ファイルには、イベントとルール（KPI とルール）を定義する 2 つのセクションがあります。

モニタとアクションに基づいて、ルールがトリガーされます。

```
<kpi>
   <event_name>VM_ALIVE</event_name>
   <metric_value>50</metric_value>
   <metric_cond>GT</metric_cond>
   <metric_type>UINT32</metric_type>
   <metric_occurrences_true>3</metric_occurrences_true>
   <metric_occurrences_false>3</metric_occurrences_false>
   <metric_collector>
      <type>ICMPPing</type>
      <nicid>0</nicid>
      <poll_frequency>15</poll_frequency>
      <polling_unit>seconds</polling_unit>
      <continuous_alarm>false</continuous_alarm>
   </metric_collector>
</kpi>
```

上記の例では、VM が動作しているかどうかを確認するイベントが送信されます。VM は定期的に ping され、その結果に基づいて VM_ALIVE イベントが VM の詳細とともにルールエンジンに送信されます。

ルールエンジンは、モニタリングエンジンからイベントを受信します。ルールエンジンは、単純なイベントから複雑なイベントまで処理できます。受信したイベントに基づいて、アクションがトリガーされます。

VM が動作していない場合、イベントに基づいて、<rule> セクションで定義されたアクションがトリガーされます。これは dep.xml データモデルで確認できます。

```
<rules>
   <admin_rules>
      <rule>
         <event_name>VM_ALIVE</event_name>
         <action>ALWAYS log</action>
         <action>FALSE recover autohealing</action>
         <action>TRUE servicebooted.sh</action>
```

```
      </rule>
   </admin_rules>
</rules>
```

ルールセクションでは、モニタリングイベントが検出されたときに実行されるアクションについて説明します。ダイナミックマッピング API は、キーワードに基づいてルールを駆動します。

上記の例では、指定された条件に基づいて次のアクションが実行されます。

- ALWAYS log：イベントが ping 可能かどうかにかかわらず、詳細がログに記録されます。
- TRUE servicebooted.sh：ダイナミックマッピング API でこのキーワードによって識別されるアクションは、VM が ping 不能状態から ping 可能状態に移行したときにトリガーされます。serviceboot スクリプトは、VM が動作していることを ESC に通知し、VM の状態を移行できるようにします。
- FALSE recover autohealing：このキーワードによって識別されるアクションがトリガーされ、管理者の介入なしで VM が回復されます。

トラブルシューティング用のモニタリングログファイルは、`/var/log/mona` にあります。

VM ネットワークステータスのモニタリング

ICMP ping モニタリングを使用する場合、ESC が VM ダウンイベントを受信すると、修復ワークフローは回復ポリシーを使用して VM の回復を試みます。ESC から VNF へのネットワークインターフェイスまたは IP ルートに問題がある場合。たとえば、ゲートウェイがダウンしている場合は、VM ダウンイベントが誤ってトリガーされ、不要なリカバリが発生する可能性があります。

インターフェイスチェック機能は、すべてのネットワークインターフェイスのヘルスステータスとゲートウェイの動作状態をチェックすることによって、ネットワークルートをさらにスキャンします。ネットワーク環境に問題がある場合は、VNF が動作していると見なされます。

ESC がネットワークの問題を検出した場合、または既存の問題が修正された場合（自動修復）、VM_NETWORK_STATE イベントがノースバウンドに送信されます。

次の障害通知がノースバウンドに送信されます。

```
16:13:15,567 14-Mar-2018 WARN  ===== SEND NOTIFICATION STARTS =====
16:13:15,567 14-Mar-2018 WARN  Type: VM_NETWORK_STATE
16:13:15,567 14-Mar-2018 WARN  Status: FAILURE
16:13:15,567 14-Mar-2018 WARN  Status Code: 500
16:13:15,567 14-Mar-2018 WARN  Status Msg: Warning: VM
[NG_G1_0_46fdcf70-f4ea-4289-ae79-08674e7d6f42] has a network problem: Network interface
 not healthy, please check.
16:13:15,567 14-Mar-2018 WARN  Tenant: tenant2
16:13:15,567 14-Mar-2018 WARN  Deployment ID: 455d2407-9dda-4203-95b0-724c4a651720
16:13:15,567 14-Mar-2018 WARN  Deployment name: NG
16:13:15,567 14-Mar-2018 WARN  VM group name: G1
16:13:15,567 14-Mar-2018 WARN  VM Source:
16:13:15,567 14-Mar-2018 WARN      VM ID: 4bee016a-6b30-43ff-a249-157a07d9b4db
16:13:15,567 14-Mar-2018 WARN     VM Name: NG_G1_0_46fdcf70-f4ea-4289-ae79-08674e7d6f42
16:13:15,568 14-Mar-2018 WARN      VM Name (Generated):
NG_G1_0_46fdcf70-f4ea-4289-ae79-08674e7d6f42
```

```
16:13:15,568 14-Mar-2018 WARN      VIM ID: default_openstack_vim
16:13:15,568 14-Mar-2018 WARN      VIM Project: tenant2
16:13:15,568 14-Mar-2018 WARN      VIM Project ID: 62afb63cd28647a7b526123cac1ba605
16:13:15,568 14-Mar-2018 WARN      Host ID:
b83004159a46c20bc8383927c2231067bb0c1905b4b4c28475653190
16:13:15,568 14-Mar-2018 WARN      Host Name: my-server-50
16:13:15,568 14-Mar-2018 WARN  =====  SEND NOTIFICATION ENDS  =====
```

ネットワークの問題が修正されると、次の成功通知がノースバウンドに送信されます。

```
16:13:19,141 14-Mar-2018 INFO  ===== SEND NOTIFICATION STARTS =====
16:13:19,141 14-Mar-2018 INFO  Type: VM_NETWORK_STATE
16:13:19,142 14-Mar-2018 INFO  Status: SUCCESS
16:13:19,142 14-Mar-2018 INFO  Status Code: 200
16:13:19,142 14-Mar-2018 INFO  Status Msg: Network of VM
[NG_G1_0_46fdcf70-f4ea-4289-ae79-08674e7d6f42] has been restored.
16:13:19,142 14-Mar-2018 INFO  Tenant: tenant2
16:13:19,142 14-Mar-2018 INFO  Deployment ID: 455d2407-9dda-4203-95b0-724c4a651720
16:13:19,142 14-Mar-2018 INFO  Deployment name: NG
16:13:19,142 14-Mar-2018 INFO  VM group name: G1
16:13:19,142 14-Mar-2018 INFO  VM Source:
16:13:19,142 14-Mar-2018 INFO      VM ID: 4bee016a-6b30-43ff-a249-157a07d9b4db
16:13:19,142 14-Mar-2018 INFO     VM Name: NG_G1_0_46fdcf70-f4ea-4289-ae79-08674e7d6f42
16:13:19,142 14-Mar-2018 INFO      VM Name (Generated):
NG_G1_0_46fdcf70-f4ea-4289-ae79-08674e7d6f42
16:13:19,142 14-Mar-2018 INFO      VIM ID: default_openstack_vim
16:13:19,143 14-Mar-2018 INFO      VIM Project: tenant2
16:13:19,143 14-Mar-2018 INFO      VIM Project ID: 62afb63cd28647a7b526123cac1ba605
16:13:19,143 14-Mar-2018 INFO      Host ID:
b83004159a46c20bc8383927c2231067bb0c1905b4b4c28475653190
16:13:19,143 14-Mar-2018 INFO      Host Name: my-server-50
16:13:19,143 14-Mar-2018 INFO  =====  SEND NOTIFICATION ENDS  =====
```

ETSI API を使用した VNF のモニタリングについては、『*Cisco Elastic Services Controller ETSI NFV MANO Guide*』を参照してください。

VM モニタリングステータスの通知

ESC は、次の条件で VM_MONITORING_STATUS を送信します。

モニタリングスクリプトが欠落している場合、または ESC スイッチオーバー後にモニタが停止しているときにモニタリングタイマーが期限切れになった場合、モニタリングの設定または設定解除操作中にエラーが発生します。

VM_MONITOR_STATUS 通知が NB に送信されます。ESC は VM をモニタせず、リカバリプロセスを開始できません。障害発生後にモニタリングを有効にするには、モニタリングを無効にしてから有効にする必要があります。

通知

```
WARN ===== SEND NOTIFICATION STARTS =====
WARN Type: VM_MONITORING_STATUS
WARN Status: FAILURE
WARN Status Code: 500
WARN Status Msg: No response from the monitor
WARN Tenant: tenant
WARN Deployment ID: 02cc4018-e4e3-4974-884a-f9fee17d7040
WARN Deployment name: dep
```

```
WARN VM group name: g1
WARN VM Source:
WARN VM ID: 6aa98b79-9d35-442a-9abb-f611e6316083
WARN VM Name: dep_g1_0_7fdae2a6-5095-4071-9c50-fb80c0e6b80e
WARN VM Name (Generated): dep_g1_0_7fdae2a6-5095-4071-9c50-fb80c0e6b80e
WARN VIM ID: default_openstack_vim
WARN VIM Project: tenant
WARN VIM Project ID: 33bf6768e45445da87feed838b248849
WARN Host ID: 79e4104d1d33de80aab13205b1e3c61d64aa4b61230c8b7b064b2891
WARN Host Name: my-ucs-62
WARN ===== SEND NOTIFICATION ENDS =====
```

モニタリング操作

RESTful インターフェイスを使用して VM のモニタリングを設定および設定解除できます。

VM のモニタリングにはペイロードが必要です。

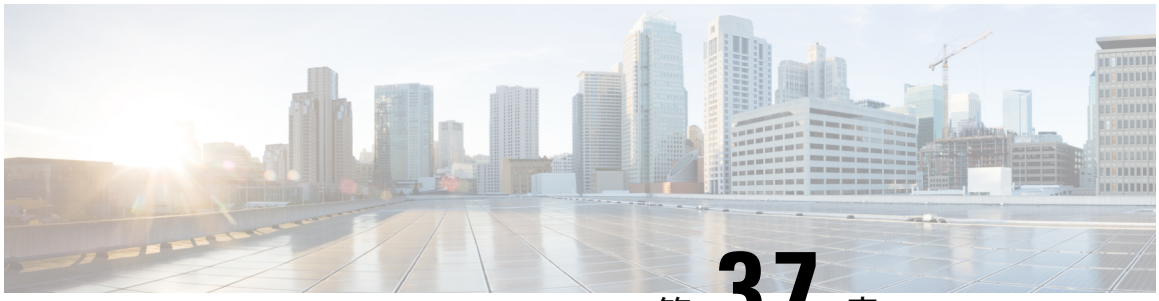
```
POST ESCManager/v0/{internal_tenant_id}/deployments/vm/{vm_name}
```

例：

```
<?xml version="1.0" encoding="UTF-8"?>
<vm_operation xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <operation>enable_monitoring</operation>
   <force>false</force>
</vm_operation>
```

VM モニタリングを設定するには enable_monitoring を、VM モニタリングを設定解除するには disable_monitoring を、それぞれ operation フィールドに指定する必要があります。

（注） ユーザーが ESC ポータルから VM を再起動すると、モニタリングが自動的に有効になります。



第 37 章

D-MONA を使用した VNF のモニタリング

ESC モニタリングおよびアクション（MONA）は、ESC によって展開される VNF をモニタします。精度を維持するために、ping、custom_scripts などのアクションを特定の間隔で実行します。

D-MONA のオンボーディング

D-MONA を展開する前に、次の前提条件を満たしている必要があります。

前提条件

- ESC と D-MONA の間に接続が存在することを確認します。
- D-MONA と展開された VNF 間に接続が存在することを確認します。

展開が成功すると、D-MONA は ESC VM で実行されているローカル MONA によって監視されます。

（注） 別の D-MONA による D-MONA のモニタリングはサポートされていません。

D-MONA の展開

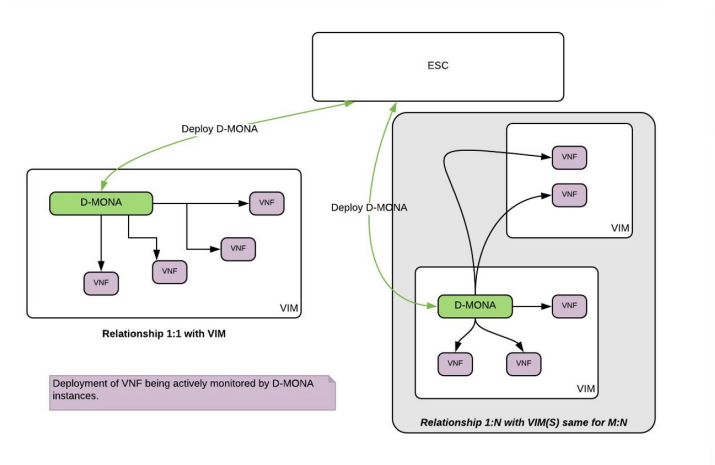
ESC 5.3 以降では、1：1 マッピングは不要です。明示的な D-MONA 展開をサポートします。

- このシナリオでは、複数の D-MONA インスタンスを導入できます。
- VNF は、指定したモニタリングエージェントの下に展開するか、または指定したモニタリングエージェントに移行できます。

インフラストラクチャで D-MONA を使用するには、次の手順を実行する必要があります。

1. モニタリング インフラストラクチャで D-MONA を展開します。
2. D-MONA を使用して VNF を展開し、モニタリングします。

図 *3 : D-MONA* の展開タイプ



モニタリングに D-MONA を使用しない場合は、「仮想ネットワーク機能のモニタリング」を参照してください。

次の表に、大規模導入の D-MONA VM フレーバーを示します。

展開	VM の数	VM ごとの仮想 CPU	VM ごとの仮想メモリ（GB）	VM ごとの仮想ディスク（GB）	サポートされる VM の合計数
D-MONA	1	4	8	40	1500

D-MONA の設定

D-MONA の設定中、2 種類のランタイム動作を表示できます。1 つは一般的な ESC 展開で予想されるすべての動作を表示でき、もう 1 つは D-MONA が提供する機能を示します。

D-MONA のランタイム動作は、展開時に VM に指定されるデイゼロ設定によって制御されます。デイゼロ設定の詳細については、「デイゼロ設定」を参照してください。

HA アクティブ/スタンバイおよびスタンドアロンの通知 URL を指定する必要があります。ただし、アクティブ/アクティブ HA の場合、URL は展開時に自動生成または計算されます。

D-MONA Day 0 設定

次の例は、D-MONA の SSH VM アクセス設定を示しています。

```
<configuration>
 <dst>--user-data</dst>
 <file>file:///opt/cisco/esc/esc-config/dmona/user-data.template</file>
 <variable>
  <name>vm_credentials</name>
  <val>REPLACED_WITH_GENERATED_PWD</val>
 </variable>
</configuration>
```

次の例は、HA アクティブ/スタンバイおよびスタンドアロンの通知 URL を示しています。

```
<variable>
  <name>notification.url</name>
  <val>
    http(s)://xxx.xx.x.xx:xxxx/ESCManager/dmona/api/events/notif
  </val>
</variable>
```

vm_credentials は、D-MONA への SSH アクセスのために暗号化されたパスワードを管理者に渡します。

次の例は、D-MONA ESC 証明書の設定を示しています。

```
<configuration>
 <dst>/opt/cisco/esc/moan/dmona.crt</dst>
 <data>$DMONA_CERT</data>
</configuration>
```

次の例は、D-MONA アプリケーションのユーザデータ設定を示しています。

```
<configuration>
<dst/opt/cisco/esc/mona/config/application-dmona.properties</dst>
 <file>file:///opt/cisco/esc/esc-config/dmona/application-dmona.template</file>
 <variable>
  <name>monitoring.agent</name>
  <val>true</val>
 </variable>
 <variable>
  <name>monitoring.agent.vim.mapping</name>
  <val>true</val>
 </variable>
 <!—Used to enable Basic Authentication for communication with the D-MONA Application.->

 <variable>
  <name>security_basic_enabled</name>
  <val>true</val>
 </variable>
 <variable>
  <name>security_user_name</name>
  <val>REPLACED_WITH_USER_NAME</val>
 </variable>
 <variable>
  <name>security_user_password</name>
```

```
  <val>REPLACED_WITH_USER_PASSWORD</val>
 </variable>

</configuration>
```

次に、CSP の D-MONA のデイゼロ テンプレート ファイルの例を示します。

展開前に、適切なアクセス権限を持つすべてのESCインスタンスの /var/tmp/ ディレクトリに D-MONA のデイゼロテンプレートをアップロードします。

```
#cloud-config
users:
 - name: admin          # The user's login name
   gecos: admin         # The user name's real name
   groups: esc-user     # add admin to group esc-user
   passwd: $vm_credentials
                        # The hash -- not the password itself -- of the password you
want
                        #           to use for this user. You can generate a safe hash
via:
                        #               mkpasswd --method=SHA-512 --rounds=4096
   lock-passwd: false   # Defaults to true. Lock the password to disable password login
                        # Set to false if you want to password login
   homedir: /home/admin # Optional. Set to the local path you want to use. Defaults to
/home/<username>
   sudo: ALL=(ALL) ALL  # Defaults to none. Set to the sudo string you want to use

ssh_pwauth: True        # Defaults to False. Set to True if you want to enable password
 authentication for sshd.
write_files:
# ESC Configuration
- path: /opt/cisco/esc/esc-config/esc-config.yaml
  content: |
    resources:
      mona:
        dmona: true
- path: /etc/sysconfig/network-scripts/ifcfg-eth0
  content: |
    DEVICE="eth0"
    BOOTPROTO="none"
    ONBOOT="yes"
    TYPE="Ethernet"
    USERCTL="yes"
    IPADDR="${NICID_0_IP_ADDRESS}"
    NETMASK="${NICID_0_NETMASK}"
    GATEWAY="${NICID_0_GATEWAY}"
    DEFROUTE="yes"
    NM_CONTROLLED="no"
    IPV6INIT="no"
    IPV4_FAILURE_FATAL="yes"
- path: /etc/sysconfig/network-scripts/ifcfg-eth1
  content: |
    DEVICE="eth1"
    BOOTPROTO="none"
    ONBOOT="yes"
    TYPE="Ethernet"
    USERCTL="yes"
    IPADDR="${NICID_1_IP_ADDRESS}"
    NETMASK="${NICID_1_NETMASK}"
    GATEWAY="${NICID_1_GATEWAY}"
    DEFROUTE="yes"
    NM_CONTROLLED="no"
```

```
    IPV6INIT="no"
    IPV4_FAILURE_FATAL="yes"
runcmd:
- [ cloud-init-per, once, apply_network_config, sh, -c, "systemctl restart network"]
- [ cloud-init-per, once, copy_dmona_config, sh, -c, "cp -RT
/media/cdrom/opt/cisco/esc/mona/ /opt/cisco/esc/mona/"]
- [ cloud-init-per, once, esc_service_start, sh, -c, "chkconfig esc_service on && service
 esc_service start"] # You must include this line
```

明示的な D-MONA モニタリングエージェントを使用した VNF の展開

ESC 5.3 以降、ESC では VNF をモニタするために D-MONA 識別子を明示的に指定できます。次に、VNF を明示的に使用して VNF を D-MONA のモニタリングエージェントに展開する手順を示します。

手順

ステップ 1 D-MONA の Day 0 設定の monitoring.agent.vim.mapping プロパティを省略または False に設定して D-MONA を展開します。

次の例は、monitoring.agent.vim.mapping が False に設定されている D-MONA データモデルの Day 0 設定を示しています。

```
<configuration>
        <dst>/opt/cisco/esc/mona/config/application-dmona.properties</dst>
        <file>file:///opt/cisco/esc/esc-config/dmona/application-dmona.template</file>
        <variable>
            <name>monitoring.agent</name>
            <val>true</val>
        </variable>
        <!-- property for one to one mapping - omit or set to false for explicit VNF to
 D-MONA mapping-->
        <variable>
            <name>monitoring.agent.vim.mapping</name>
            <val>false</val>
        </variable>
        <!-- property to enable basic auth in dmona. Not to be confused with basic auth
 for esc -->
        <variable>
            <name>security_basic_enabled</name>
            <val>true</val>
        </variable>
        <variable>
            <name>security_user_name</name>
            <val>REPLACE_WITH_USER_NAME</val>
        </variable>
        <variable>
            <name>security_user_password</name>
            <val>REPLACE_WITH_USER_PASSWORD</val>
        </variable>
</configuration>
```

ステップ 2 展開データモデルの KPI 設定で monitoring_agent パラメータを指定して、VNF を展開します。

タグ <*monitoring_agent*> は、VNF をモニタする分散型 MONA 展開の明示的な識別子として使用されます。タグが存在する場合、ESC はその正確な展開名を持つ分散型 MONA 展開を探します。D-MONA 識別子は、以前に展開された D-MONA VNF を表す特定のスキームを使用して URI で指定されます。

たとえば、`dmonaName://<D_MONA_DEP_NAME>` は `<D_MONA_DEP_NAME>` を分散型 MONA インスタンスの展開名に置き換えます。

次の例は、モニタリングエージェントが指定された VNF データモデルの KPI 設定を示しています。

```
<kpi>
    <event_name>VM_ALIVE</event_name>
    <!-- specify dmona deployment name using dmonaName:// URI format-->
    <monitoring_agent>dmonaName://D-MONA-OTTAWA</monitoring_agent>
    <metric_value>1</metric_value>
    <metric_cond>GT</metric_cond>
    <metric_type>UINT32</metric_type>
    <metric_collector>
    <type>ICMPPing</type>
    <nicid>0</nicid>
    <poll_frequency>3</poll_frequency>
    <polling_unit>seconds</polling_unit>
    <continuous_alarm>false</continuous_alarm>
    <monitoring_public_ip>true</monitoring_public_ip>
    </metric_collector>
</kpi>
```

（注） ESC では、VNF ごとに 1 つのモニタリングエージェントしか許可されません。

トラブルシューティングのモニタリングステータス

VNF が D-MONA のモニタリングエージェントによってモニタされているかどうかを確認するには、次のコマンドを実行します。

```
curl -u username:pwd -H 'Accept:application/json'
http://localhost:8080/ESCManager/v0/api/monitoring/agents/config
```

次の例に結果が示されます。

```
{
    "a8345881-adc8-4d16-8741-9d105592c676": {
        "monitoringAgents": [
            {
                "name": "sample-dmona-10",
                "notificationUrl":
"https://172.16.235.73:8443/ESCManager/dmona/api/events/notif",
                "oneToOneMapping": false,
                "state": "ACTIVE",
                "uri": "https://172.16.235.81:8443/mona/v1/rules",
                "vimId": "OPENSTACK_VIMCONN_pf-ucs-20",
                "vnfData": [
                    {
                        "deploymentExternalId": "785e170c-55b5-4df7-929f-d34f052e4616",
                        "deploymentName": "dmona-10-vnf-121-f2b1df6d",
```

```
                        "state": "MONITORED",  <============ Monitoring state for DMONA
                        "vmGroupName": "vm1"
                    },
                    {
                        "deploymentExternalId": "2e42c8d9-51fa-4de8-a260-d3a3429be7d4",
                        "deploymentName": "dmona-10-vnf-442-faa43053",
                        "state": "MONITORED",  <============ Monitoring state for DMONA
                        "vmGroupName": "vm1"
                    }
                ]
            },
            {
                "name": "local_mona",
                "notificationUrl": "",
                "oneToOneMapping": false,
                "state": "ACTIVE",
                "uri": "http://localhost:8090/mona/v1/rules",
                "vimId": "N/A",
                "vnfData": [
                    {
                        "deploymentExternalId": "9501376e-e29e-4c99-b5fb-66ab66de45b7",
                        "deploymentName": "sample-dmona-2",
                        "state": "N/A",   <============ Local Mona monitoring state is
not available
                        "vmGroupName": "g1"
                    }
                ]
            }
        ]
    }
}
```

VIM インスタンス間での D-MONA のリカバリ

D-MONA エージェントに障害が発生した場合、ESC は障害を迅速にリカバリできるため、ダウンタイムを最小限に抑え、展開された VNF のモニタリングを最短で再開できます。ただし、エージェントがリカバリされている間、監視対象の VNF は監視されません。D-MONA によってモニタされる VNF は、D-MONA が再びアクティブになるまで、最後の既知の状態のままです。

VNF モニタリングステータスは、各 VNF モニタリングルールの再プログラミング中に D-MONA が ESC で正常にリカバリした場合にのみ復元されます。復元されるまでは、D-MONA エージェントの状態は UNKNOWN のままで、VNF の状態は UNMONITORED のままです。

次の例は、D-MONA がダウンしている場合のモニタリングエージェント API の D-MONA と VNF の状態を示しています。

```
{
    "name": "Test-dmona-dep-1",
    "notificationUrl": "",
    "oneToOneMapping": false,
    "state": "UNKNOWN",
    "uri": "https://172.29.0.15:8443/mona/v1/rules",
    "vimId": "default_openstack_vim",
    "vnfData": [
        {
            "deploymentExternalId": "70d7f1f0-362e-4d2b-a89b-4877d8bfabf4",
```

```
            "deploymentName": "Test-dep-2",
            "state": "UNMONITORED",
            "vmGroupName": "g1"
        }
    ]
}
```

モニタリングエージェントのディザスタリカバリ

VIM が使用できないために、展開された分散 D-MONA に到達できない場合、`VIM_FAILURE` の理由で HealVnfRequest を送信することにより、別の VIM で D-MONA をリカバリできます。

次の手順を使用して、別の VIM インスタンスで D-MONA をリカバリします。

- 次の SOL003 の例に従って、手動で HealVnfRequest を開始します。

メソッドタイプ：

```
POST
```

VNFM エンドポイント：

```
/vnf_instances/{vnfInstanceId}/heal
```

HTTP 要求ヘッダー：

```
Content-Type:application/json
```

要求ペイロード（ETSI データ構造：HealVnfRequest）：

```
{
  "cause":"VIM_FAILURE"
}
```

- NFVO からの許可に、D-MONA VNF を再展開する VIM を識別する新しい vimConnectionInfo が含まれていない場合、リカバリリクエストは拒否されます。

- HealVnfRequest が正常に完了すると、D-MONA VNF は新しい VIM で再作成され、以前に保持していたすべての VNF を引き続き監視します。

（注） 元の展開は、古い VIM から削除されません。古い VIM に到達できたら、以前の D-MONA を手動で削除します。

アクティブ/アクティブ HA の フェールオーバー

ESC アクティブ/アクティブ HA 展開のフェールオーバーでは、障害が発生した ESC インスタンスが所有する VNF がクラスタ内の他の ESC インスタンスに転送されます。

障害が発生した ESC インスタンスから D-MONA 展開が転送されると、モニタリングエージェント API で状態が UNKNOWN に更新されます。転送された D-MONA によってモニターされる VNF は、D-MONA モニタリングエージェントの状態が ACTIVE になると調整されます。

他の D-MONA 展開のように、転送された D-MONA によってモニターされる VNF は、D-MONA が再びアクティブになるまで、最後の既知の状態のままです。

D-MONA ログの取得

D-MONAデイゼロ設定の一部として提供されたvm_credentialsパスワードを使用してD-MONAにアクセスします。

D-MONA ログを取得するには、次のコマンドを使用します。

```
<security_user_name>:<security_user_password>
```

ip-address はターゲットの D-MONA の IP アドレス、username、password は D-MONA の展開時にデイゼロ設定として指定されたユーザ名とパスワードです。

すべての ESC ログの完全なリストについては、ESC アドミニストレーション ガイド [英語] の「ESC Logs」を参照してください。

ETSI 関連の情報については、Cisco Elastic Services Controller ETSI NFV MANO ユーザーガイド [英語] の「Monitoring VNF Using D-MONA」の章を参照してください。

D-MONA のモニタリングルールのリセット

精度を維持するために、Monitoring and Action（MONA）はping、custom_scriptsなどのアクションを特定の間隔でモニタリングおよび実行します。

ローカルMONAは、ポーリングされたD-MONAプロセスの最後の既知の起動時間を追跡します。ステータスコード200は、要求が成功したことを示します。要求が成功すると、ローカルMONA は最後の既知の起動時間を、ポーリングされたアプリケーションから返された起動時間と比較します。DMONA の再起動時に、リカバリセットアップが自動的に開始されます。

開始時刻チェックを有効にするには、dep.xml で application_startup_time を設定する必要があります。

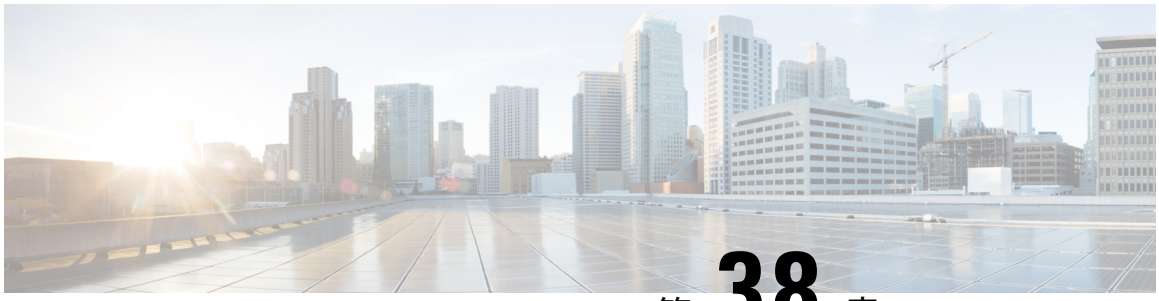
ただし、application_startup_time が存在しないか、または false に設定されている場合、DMONA リブートチェックは無効になります。D-MONA を展開するには、このプロパティを設定する必要があります。

（注） 下位互換性はサポートされていません。バージョン 5.3 以降 でのみ設定する必要があります。

次に、D-MONA の導入モデルの例を示します。

```
<?xml version="1.0"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>A_tenant_name</name>
      <deployments>
        <deployment>
          <name>dmona_deployment</name>
          <vm_group>
            <name>g1</name>
            <image>ESC-5_3_0_31</image>
```

```
                <flavor>m1.large</flavor>
                <bootup_time>120</bootup_time>
                <recovery_wait_time>0</recovery_wait_time>
                <interfaces>
                  <interface>
                    <nicid>0</nicid>
                    <network>esc-net</network>
                  </interface>
                </interfaces>
                <kpi_data>
                  <kpi>
                    <event_name>VM_ALIVE</event_name>
                    <metric_value>1</metric_value>
                    <metric_cond>GT</metric_cond>
                    <metric_type>UINT32</metric_type>
                    <metric_occurrences_true>1</metric_occurrences_true>
                    <metric_occurrences_false>5</metric_occurrences_false>
                    <metric_collector>
                      <type>HTTPGET</type>
                      <nicid>0</nicid>
                      <poll_frequency>3</poll_frequency>
                      <polling_unit>seconds</polling_unit>
                      <continuous_alarm>false</continuous_alarm>
                      <properties>
                     <!-- Set to true to enable start time check -->
                    <property>
                    <name>application_startup_time</name>
                    <value>true</value>
                    </property>
                     <property>
                          <name>protocol</name>
                          <value>https</value>
                       </property>
                       <property>
                         <name>port</name>
                         <value>8443</value>
                       </property>
                       <property>
                         <name>path</name>
                         <value>mona/v1/health/status</value>
                       </property>
                      </properties>
                    </metric_collector>
                  </kpi>

                </kpi_data>
[...]
          </vm_group>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```



第 38 章

モニタリングエージェントの移行

モニタリングエージェントの移行

各 ESC インスタンスには、ESC がリカバリおよびスケーリング操作を制御できるように、それをモニタするエージェントがあります。モニタリングエージェントの移行が必要なさまざまなシナリオを次に示します。

1. **ローカル**から**分散型**への移行

   次に例を示します。

   新しい D-MONA をデータセンターに導入する場合。

2. **分散型**から**ローカル**への移行

   次に例を示します。

   ソフトウェアアップグレードを実行する場合。

3. **分散型**から**分散型**への移行

   次に例を示します。

   ロードバランシングを実行する場合。

4. **分散型**から**分散型**への多数のインスタンスの迅速な移行

   次に例を示します。

   ディザスタ リカバリ

モニタリングエージェントを移行するには、次の手順に従います。

手順

ステップ 1 展開データモデルの KPI 設定セクションで、<monitoring_agent> のタグ値を追加/編集します。

a) D-MONA に移行するには、次の手順を実行します。

<monitoring_agent>dmonaName://dmona-dep-name</monitoring_agent> を設定します。dmona-dep-name は D-MONA の展開名です。

b) ローカルの MONA に移行するには、次の手順を実行します。

<monitoring_agent>dmonaName://local_mona</monitoring_agent> を設定します。local_monaは ESC 5.3 でローカル MONA 用に導入された特別な識別子です。

ステップ 2 更新された展開データモデルを使用した、サービス更新の実行：

サービスの更新を実行すると、現在のモニタリングエージェントでモニタの設定が解除され、新しいモニタリングエージェントで VNF が更新され、新しいモニタリングエージェントでモニタが設定されます。

monitoring_agent パラメータの詳細については、「明示的な D-MONA モニタリングエージェントを使用した VNF の展開」の章を参照してください。

移行後の通知

ESC は移行後に 3 つの通知を NorthBound に送信します。

1. SERVICE_UPDATED 通知：

   更新が成功したかどうかを示すために送信されます。

2. VM_SET_MONITOR_STATUS 通知：

   この通知は、VNF 内の各 VM の新しいモニタリングエージェント上の監視設定ステータスを示すために送信されます。

3. SVC_SET_MONITOR_STATUS 通知

   展開の監視設定のサービスレベルステータスを示すために送信されます。

NorthBound が正常な SERVICE_UPDATED および SVC_SET_MONITOR_STATUS 通知を受信すると、モニタリングエージェントの移行は成功したと見なされます。

次の例は、VM_SET_MONITOR_STATUS 通知を示しています。

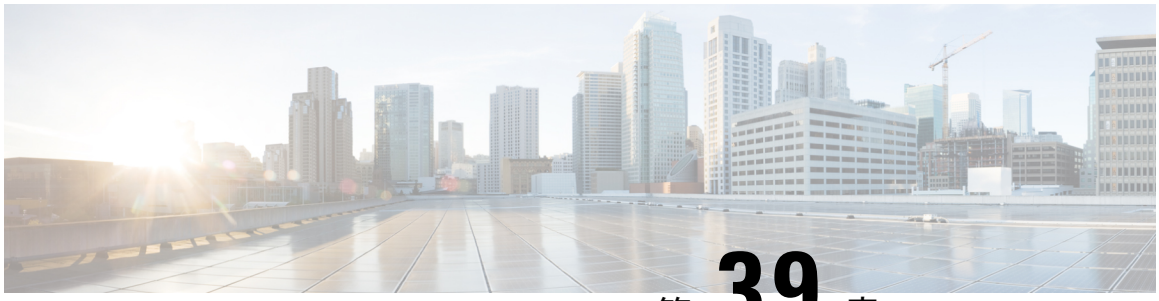
```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-08-06T14:04:47.124+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>VM monitor setting completed successfully.</status_message>
    <depname>test-dep</depname>
    <tenant_id>563fba7044c847a6a370cc10d5ef7d57</tenant_id>
    <depid>995f6849-0599-4287-bc3b-fca6de7bfcb2</depid>
    <vm_group>g1</vm_group>
    <vm_source>
      <vmid>ca40ccb1-fe21-4846-a15f-79900e7e3baa</vmid>
```

```
      <vmname>test-dep_g1_0_88e9b2af-aef2-472c-84c1-1dbbf96df31f</vmname>

<generated_vmname>test-dep_g1_0_88e9b2af-aef2-472c-84c1-1dbbf96df31f</generated_vmname>
      <hostid>16e897fa14b3d1ecee0f7489a7a9ac7902f66c1f017437f27474a4c5</hostid>
      <hostname>my-ucs-3</hostname>
      <interfaces>
        <interface>
          <nicid>0</nicid>
          <type>virtual</type>

<vim_interface_name>test-dep_g1_0_88e9b2af-aef2-472c-84c1-1dbbf96df31f</vim_interface_name>

          <port_id>f8cc9d5b-6bb0-4050-98bd-8aa25d71a68c</port_id>
          <network>3d8a4b3d-6ced-4733-8143-6cea6da85411</network>
          <subnet>e0f2da9e-0c8d-4351-847a-1bf36cc3ffdc</subnet>
          <ip_address>172.29.0.9</ip_address>
          <mac_address>fa:16:3e:f6:3b:b7</mac_address>
          <netmask>255.255.240.0</netmask>
          <gateway>172.29.0.1</gateway>
        </interface>
      </interfaces>
      <properties>
        <property>
          <name>monitoring_agent</name>
          <value>dmonaName://test-dmona-dep-1</value>
        </property>
      </properties>
    </vm_source>
    <event>
      <type>VM_SET_MONITOR_STATUS</type>
    </event>
  </escEvent>
</notification>
```

次の例は、SVC_SET_MONITOR_STATUS 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>2020-08-06T14:04:47.132+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Service monitor setting completed successfully.</status_message>
    <depname>test-dep</depname>
    <tenant>admin</tenant>
    <tenant_id>563fba7044c847a6a370cc10d5ef7d57</tenant_id>
    <depid>995f6849-0599-4287-bc3b-fca6de7bfcb2</depid>
    <monitoring>
      <vm_group>
        <name>g1</name>
        <monitoring_agent>dmonaName://test-dmona-dep-1</monitoring_agent>
       <status_message>VM group setting monitor completed successfully.</status_message>

      </vm_group>
    </monitoring>
    <event>
      <type>SVC_SET_MONITOR_STATUS</type>
    </event>
  </escEvent>
</notification>
```

ETSI API を使用した VNF のモニタリングエージェントの移行については、『Cisco Elastic Services Controller ETSI NFV MANO ユーザーガイド』の「モニタリングエージェントの移行」の章を参照してください。



第 39 章

仮想ネットワーク機能のスケーリング

スケーリングの概要

ESC では、サービスを柔軟に拡張でき、スケールインとスケールアウトの両方を自動的に実行するように設定できます。スケーリングは、KPI、ルール、およびアクションを使用して実現されます。これらは展開時に設定されます。KPI では、イベント名としきい値を定義します。ルールでは、スケールアウトとスケールインをトリガーするアクションを定義します。

KPI、ルール、およびメトリックの詳細については、KPI、ルール、およびメトリック （181 ページ）を参照してください。

VM のスケールインとスケールアウト

スケーリングワークフローは、VNF の展開が成功した後に開始されます。VM は、データモデルの KPI データを形成する CPU 負荷、メモリ使用率などの属性をモニタするように設定されます。いずれかの属性について、定義されたアクションに基づいて KPI がしきい値に達すると、スケールインとスケールアウトが実行されます。

- スケールアウト中に、VM の数がアクティブな最大数を下回ると、新しい VM の展開がトリガーされます。
- スケールイン中に、VM の数がアクティブな最小数を超えると、VM は展開解除されます。

（注） VM が展開され、VM Alive イベントを受信しなかった場合、リカバリがトリガーされます。展開解除中のエラーは、ノースバウンドユーザに通知されます。

データモデルのスケーリングセクションでは、最小値と最大値が設定されます。min_active では、展開された VM の数を定義します。max_active では、展開可能な VM の最大数を定義します。たとえば、最小で 2、最大で 100 の VM を指定して VNF を展開する場合、以下の XML で各 VM グループのスケーリングを定義します。

スタティック IP アドレスを使用してアクティブ VM を設定した場合、スケールアウトされた VM にスタティック IP アドレスを割り当てる必要があります。展開時に、スタティック IP アドレスのリストを指定する必要があります。次に、スタティック IP プールを作成する例を示します。

```
 <scaling>
   <min_active>1</min_active>
   <max_active>2</max_active>
   <elastic>true</elastic>
   <static_ip_address_pool>
      <network>1234-5678-9123</network>
      <gateway>10.86.22.1</gateway>
      <netmask>255.255.255.0</netmask>
      <ip_address>10.86.22.227</ip_address>
      <ip_address>10.86.22.228</ip_address>
   </static_ip_address_pool>
</scaling>
```

次に、KPI データセクションで CPU 負荷を検出する方法の例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<kpi>
  <event_name>VM_OVERLOADED</event_name>
  <metric_value>70</metric_value>
  <metric_cond>GT</metric_cond>
  <metric_type>UINT32</metric_type>
  <metric_occurrences_true>2</metric_occurrences_true>
  <metric_occurrences_false>4</metric_occurrences_false>
  <metric_collector>
    <type>CPU_LOAD_1</type>
    <nicid>0</nicid>
    <poll_frequency>3</poll_frequency>
    <polling_unit>seconds</polling_unit>
    <continuous_alarm>false</continuous_alarm>
  </metric_collector>
</kpi>
<kpi>
   <event_name>VM_UNDERLOADED</event_name>
   <metric_value>40</metric_value>
   <metric_cond>LT</metric_cond>
   <metric_type>UINT32</metric_type>
   <metric_occurrences_true>2</metric_occurrences_true>
   <metric_occurrences_false>4</metric_occurrences_false>
   <metric_collector>
      <type>CPU_LOAD_1</type>
      <nicid>0</nicid>
      <poll_frequency>3</poll_frequency>
      <polling_unit>seconds</polling_unit>
      <continuous_alarm>false</continuous_alarm>
   </metric_collector>
</kpi>
```

KPI ルールは次のとおりです。

```
<rule>
   <event_name>VM_OVERLOADED</event_name>
   <action>ALWAYS log</action>
   <action>TRUE servicescaleup.sh</action>
</rule>
<rule>
   <event_name>VM_UNDERLOADED</event_name>
   <action>ALWAYS log</action>
   <action>TRUE servicescaledown.sh</action>
</rule>
```

ETSI API を使用した VNF のスケーリングについては、Cisco Elastic Services Controller NFV MANO ガイド [英語] を参照してください。

スケーリングのためのリソースの一貫した順序付け

ESC では、IP アドレス、MAC アドレス、デイゼロ設定変数などのリソースを展開データモデルに一貫した方法で指定できます。

ESC は、手動および自動スケーリング中に、一貫した方法で展開データモデル内の静的 IP アドレスプールを割り当ておよび割り当て解除します。

次に例を示します。

```
<scaling>
  <min_active>3</min_active>
  <max_active>6</max_active>
  <static_ip_address_pool>
    <network>jenkins-internal-vnf-net-1</network>
    <ip_address>192.168.15.3</ip_address>
    <ip_address>192.168.15.111</ip_address>
    <ip_address>192.168.15.22</ip_address>
    <ip_address>192.168.15.5</ip_address>
    <ip_address>192.168.15.4</ip_address>
    <ip_address>192.168.15.222</ip_address>
  </static_ip_address_pool>
</scaling>
```

- 手動スケーリング：ESC は、スケールアウト時に静的 IP プールで使用可能な順序で IP アドレスを割り当てます。スケールインの間、IP アドレスはラストインファーストアウトの順序でリリースされます。
- 自動スケーリング：自動スケーリングは、SNMP イベントを使用して VNF のオーバーロードとアンダーロードを示します。オーバーロードイベントによって ESC がスケールアウトし、展開データモデルにリストされている順序から、静的 IP プールの最初の空き IP アドレスを割り当てます。スケールインの間、ESC は IP アドレスの割り当てを解除し、IP アドレスは将来のスケーリングイベントのために解放されます。

デイゼロ設定、展開データモデルの IP アドレスの詳細については、導入パラメータ （169 ページ）を参照してください。

スケーリング通知とイベント

スケーリング通知は、ノースバウンドユーザに送信されます。通知には、スケーリング中のサービスを特定するためのステータスメッセージとその他の詳細情報が含まれます。通知のリストは次のとおりです。

```
VM_SCALE_OUT_INIT
VM_SCALE_OUT_DEPLOYED
VM_SCALE_OUT_COMPLETE
VM_SCALE_IN_INIT
VM_SCALE_IN_COMPLETE
```

次の表に、スケーリングシナリオと生成される通知を示します。

シナリオ	通知

スケール アウト	ESC が VM を展開し、KPI\Monitor と受信したすべての VM Alive が設定されます。次の NETCONF 通知がトリガーされます。
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	ESC が VM_OVERLOADED イベントを受信すると、次の NETCONF 通知がトリガーされます。
	`<type> VM_SCALE_OUT_INIT</type>`
	`<status>SUCCESS</status>`
	ESC が最大値の制限に達したかチェックし、達していない場合は新しい VM が展開されます。
	`<type> VM_SCALE_OUT_DEPLOYED</type>`
	`<status>SUCCESS</status>`
	展開が完了すると、次の NETCONF 通知が送信されます。
	`<type>VM_SCALE_OUT_COMPLETE</type>`
	`<status>SUCCESS</status>`

シナリオ	通知

スケールイン	ESC が VM を展開し、KPI\Monitor と受信したすべての VM Alive が設定されます。
	NETCONF 通知が送信されます。
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	ESC が VM_UNDERLOADED イベントを受信すると、次の NETCONF 通知がトリガーされます。
	`<type> VM_SCALE_IN_INIT</type>`
	`<status>SUCCESS</status>`
	ESC が VM の数が最小アクティブ制限を超えているかチェックします。超えている場合は、展開解除の完了後に、いずれかの VM の展開が解除され、NETCONF 通知が送信されます。
	`<type>VM_SCALE_IN_COMPLETE</type>`
	`<status>SUCCESS</status>`

すべてのエラーシナリオで、通知は FAILURE ステータスで送信されます。また、ステータスメッセージには、対応する障害の詳細が表示されます。

第 40 章

仮想ネットワーク機能の修復

修復の概要

ライフサイクル管理の一環として、ESC は障害発生時に VNF を修復します。修復パラメータは、データモデルの KPI セクションで設定されます。展開中に指定したリカバリポリシーがリカバリを制御します。ESC はポリシー駆動型フレームワークを使用したリカバリをサポートしています。

ESC は KPI を使用して VM をモニターします。イベントは KPI 条件に基づいてトリガーされます。トリガーされるすべてのイベントに対して実行されるアクションは、展開時にルールセクションで設定されます。

VM の修復

各 VM グループは、修復を有効にするように設定されます。修復は、データモデルで定義されたリカバリポリシーを使用して、サービスの動作前と動作後の 2 つの段階で実行されます（VNF レベル、必要に応じて、VM レベルでオーバーライドされます）。

VM は展開され、モニタされています。ESC が VM Alive イベントを受信後、VM Down イベントを受信すると、設定済みのリカバリポリシーを使用して、修復ワークフローによる VM のリカバリが試みられます。

ESC は展開後に VM Alive を受信しない場合、タイムアウト発生時にリカバリポリシーを使用して VM を回復します。リカバリ手順はすべて、リカバリポリシーの設定、および以下で説明されているポリシー名によって異なります。追加のポリシーを使用して、VM を手動でリカバリするときの動作をオーバーライドできます。

ESC は、YANG ベースのデータモデルに、修復を定義するために必要なすべてのパラメータと説明の包括的な詳細情報を提供します。ESC は、イベントとルールを定義するデータモデル XML ファイル内の 2 つのセクションを使用します。

- <kpi> セクションでは、モニタリングのタイプ、イベント、ポーリング間隔、およびその他のパラメータを定義します。
- <rule> セクションでは、KPI モニタリングイベントがトリガーされたときのアクションを定義します。

KPI、ルール、およびデータモデルの詳細については、KPI、ルール、およびメトリック （181 ページ）を参照してください。

設定には、次の手順が含まれます。

1. KPI の定義
2. ルールの定義

次に、データモデルで KPI を設定する例を示します。

```
<kpi>
<event_name>VM_ALIVE</event_name>
<metric_value>1</metric_value>
<metric_cond>GT</metric_cond>
<metric_type>UINT32</metric_type>
<metric_collector>
<type>ICMPPing</type>
<nicid>0</nicid>
<poll_frequency>3</poll_frequency>
<polling_unit>seconds</polling_unit>
<continuous_alarm>false</continuous_alarm>
</metric_collector>
</kpi>
```

次の例は、すべてのイベントのルールを設定する方法を示しています。

```
<rules>
<admin_rules>
<rule>
<event_name>VM_ALIVE</event_name>
<action>ALWAYS log</action>
<action>FALSE recover autohealing</action>
<action>TRUE servicebooted.sh</action>
</rule>
</admin_rules>
</rules>
```

前述の例では、nicid 0 で ICMP Ping をモニタする KPI を定義しています。また、属性メトリック条件とポーリングを定義しています。KPI に基づいて、VM_ALIVE イベントが適切な値でトリガーされます。対応するルールのアクションでは、次のステップを定義します。

- FALSE：VM のリカバリをトリガーします。
- TRUE：定義されたアクションをトリガーします。

リカバリポリシーで設定された再起動、および再展開オプションを使用してVMでリカバリがトリガーされた場合、ESC は VM リカバリの最初のステップとして VM を再起動します。失敗した場合、VM は展開解除され、同じデイゼロ設定の新しい VM が展開されます。ESC は、以前の VM と同じネットワーク設定（MAC や IP アドレスなど）を再利用しようとします。

通常、VM が到達不能な場合、ESC は到達不能なすべての VM で VM リカバリを開始します。ESC はネットワークの停止中は VM リカバリを一時停止するため、ネットワークの停止中は VM リカバリが遅延します。ESC は到達不能な VM を検出し、最初にゲートウェイの到達可能性を評価して、ネットワーク障害の存在を検出します。

ESC がゲートウェイに ping を実行できない場合、VM の回復アクションは実行されません。ゲートウェイが到達可能になると、VM リカバリが再開されます。

二重障害状態の場合、つまり、ネットワークゲートウェイと VM の障害が同時に発生した場合、ゲートウェイが再度到達可能になった後、ESC は自動的に VM モニタリングを実行します。

ETSI API を使用した VNF の修復の詳細については、Cisco Elastic Services Controller NFV MANO ガイド [英語] を参照してください。

リカバリポリシー

ESC には、VNF の展開時に指定できる次の VM リカバリタイプがあります。

- **自動回復**
- **手動回復**

ESCは、ポリシー主導型フレームワークを使用したリカバリをサポートしています。詳細については、「リカバリポリシー（ポリシーフレームワークを使用）」を参照してください。

展開データモデルで指定できる VM リカバリには、次の 3 種類のアクションがあります。

- **REBOOT_THEN_REDEPLOY（デフォルト）**：VM ダウンイベントを受信するか、タイマーが期限切れになると、修復ワークフローは最初にVMの再起動を試行し、再起動に失敗すると、同じホストで VM の再展開を試行します。
- **REBOOT_ONLY**：VM ダウンイベントを受信するか、タイマーが期限切れになると、修復ワークフローは VM の再起動のみを試行します。
- **REDEPLOY_ONLY**：VM ダウンイベントを受信するか、タイマーが期限切れになると、修復ワークフローは VM の再展開のみを試行します。

（注） VM を再展開する REBOOT_THEN_REDEPLOY および REDEPLOY_ONLY がポリシーに含まれ、配置ポリシーが適用されていない場合、VIM は VM を再展開するホストを決定します。

（注） ESC は、vCloud Director の手動リカバリと自動リカバリの両方をサポートしています。3 種類のリカバリアクションはすべて、vCloud Director に適用されます。REBOOT_THEN_REDEPLOY がデフォルトのリカバリアクションです。vCD の展開については、VMware vCloud Director（vCD）での仮想ネットワーク機能の展開 （152 ページ）を参照してください。

VM の再展開を伴うリカバリアクションでは、障害があるか、削除された ESC 管理対象のエフェメラルポートとボリュームが自動的に再作成および接続され、リカバリが成功することを確認します。

自動回復

自動回復では、リカバリタイプパラメータは[自動（Auto）]に設定されます。ESC は、リカバリポリシーで指定された<action-on-recovery>値により、VM を自動的に回復させます。ユーザがリカバリタイプを選択しない場合、リカバリタイプはデフォルトで自動になります。

```
<recovery_policy>
 <recovery_type>AUTO</recovery_type>
 <action_on_recovery>REBOOT_THEN_REDEPLOY</action_on_recovery>
 <max_retries>3</max_retries>
</recovery_policy>
```

手動回復

VM の手動回復

手動回復では、ESC は VM_MANUAL_RECOVERY_NEEDED 通知をノースバウンド（NB）に送信し、NB から回復のための指示を待ちます。ESC は、NB からリカバリ指示を受信すると、リカバリを実行します。展開全体の手動回復については、以下を参照してください。展開の手動回復 （348 ページ）

ESC は、リカバリポリシーの action-on-recovery パラメータを使用して、単一のリクエストベースでのアクションのオーバーライドもサポートします。前述の 3 つのリカバリアクションに加えて、次の 2 つのリカバリアクションを利用できます。

- **RESET_STATE_THEN_REBOOT**：VM を再起動する前に、VM の状態がリセットされ、リカバリのために VIM が VM を再起動できるようになります。これは OpenStack にのみ適用されます。
- **DISASTER_RECOVERY**：VNF が展開されている VIM が使用できなくなり、サービスを継続するために VNF を新しい VIM に移動する必要がある場合、このアクションを呼び出して VNF（個々の VM ではなくサービス全体）を新しい VIM に再展開できます。

  このアクションを使用するには、VIM ロケータを更新するモデル専用サービス更新を先に実行する必要があります。この手順を実行しないと、リカバリリクエストが失敗します。このタイプのサービス更新を実行する方法の詳細については、以下を参照してください（REST API 経由のみ）。

元の VNF は削除されません。このリカバリアクションの使用は、オーケストレーションスタックから VNF に到達できないことを意味し、VIM 自体がリカバリされたときに、古い展開を手動でクリーンアップする**必要がある**と想定されるためです。

手動リカバリポリシーのデータモデルは次のとおりです。

```
<vm_group>
        ...
        <recovery_policy>
         <recovery_type>MANUAL</recovery_type>
         <action_on_recovery>REBOOT_THEN_REDEPLOY</action_on_recovery>
         <max_retries>3</max_retries>
        </recovery_policy>
        ...
  </vm_group
```

データモデルのリカバリポリシーパラメータの詳細については、Elastic Services Controller 展開属性 [英語] を参照してください。ESC ポータル（VMware のみ）でのリカバリポリシーの設定の詳細については、「ESCポータルを使用した VMware vCenter での VNF の展開」を参照してください。

VM_MANUAL_RECOVERY_NEEDED 通知は次のとおりです。

```
===== SEND NOTIFICATION STARTS =====
 WARN  Type: VM_MANUAL_RECOVERY_NEEDED
 WARN  Status: SUCCESS
 WARN  Status Code: 200
 WARN  Status Msg: Recovery event for VM
[manual-recover_error-g1_0_7d96ad0b-4f27-4a5a-bdf7-ec830e93d07e] triggered.
 WARN  Tenant: manual-recovery-tenant
 WARN  Service ID: NULL
 WARN  Deployment ID: 08491863-846a-4294-b305-c0002b9e8daf
 WARN  Deployment name: dep-error
 WARN  VM group name: error-g1
 WARN  VM Source:
 WARN      VM ID: ffea079d-0ea2-4d47-ba31-26a08e6dff22
 WARN      Host ID: 3a5351dc4bb7df0ee25e238a8ebbd6c6fcdf225aebcb9dff6ba10249
 WARN      Host Name: my-server-27
 WARN      [DEBUG-ONLY] VM IP: 192.168.0.3;
 WARN  =====  SEND NOTIFICATION ENDS  =====
```

VM の手動回復用 API

Confd API と REST API を使用して手動リカバリを実行できます。手動回復要求は、事前定義されたリカバリアクションを任意のアクションに上書きするように設定できます。

Netconf API `recovery-vm-action DO generated vm name [xmlfile]`

API を使用してリカバリを実行するには、esc_nc_cli にログインし、次のコマンドを実行します。

```
$ esc_nc_cli --user <username> --password <password> recovery-vm-action DO [xmlfile]
```

リカバリが実行され、リカバリ通知が NB に送信されます。

（注） リカバリ（recovery-vm-action DO <VM-NAME>）は、VM が動作し、サービスがアクティブになった後に実行できます。展開が不完全な場合は、リカバリを実行する前に展開を完了する必要があります。

設定可能な手動回復中にフェールオーバーが発生した場合、手動回復は事前定義されたリカバリアクションで再開されます。

展開の移行では、常にデフォルトのリカバリポリシーを使用する必要があります。LCS ベースのリカバリでは、VM / VNF 手動回復のリカバリアクションを指定しないでください。モニタの有効化オプションと設定可能な手動回復オプションを同時に使用することはできません。

REST API

```
http://ip:8080/ESCAPI/#!/Recovery_VM_Operations/handleOperation
```

```
POST /v0/{internal_tenant_id}/deployments/recovery-vm/{vm_name}
```

リカバリ VM 操作ペイロード：

```
{
"operation":"recovery_do",
"properties":{
"property":[
{
"name":"action",
"value":"REDEPLOY_ONLY"
}
]
}
}
```

モデル専用サービス更新を実行するには、新しいパラメータを edit-config API に提供して、VIM でアクションが実行されないようにし、更新を ESC データモデルのみに制限します。これにより、VIM で展開を更新する準備が整うまでに、データモデルの準備を完了できます。

```
http://ip:8080/ESCManager/v0/conf/edit-config?modelOnly=true
```

たとえば、<action-on-recovery> として `DISASTER_RECOVERY` を使用してリカバリ API を呼び出す前に、VIM ロケータを更新します。

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <tenants>
      <tenant>
         <name>admin-tenant</name>
         <deployments>
            <deployment>
               <name>test-deploy</name>
               <networks>
                  <network>
                     <name>test-network</name>
                     <locator>
                        <vim_id>my-ucs-59</vim_id>
                        <vim_project>admin</vim_project>
                     </locator>
```

```
                </network>
             </networks>
             <vm_group>
                <name>g1</name>
                <locator>
                   <vim_id>my-ucs-59</vim_id>
                   <vim_project>admin</vim_project>
                </locator>
                <bootup_time>120</bootup_time>
             </vm_group>
          </deployment>
       </deployments>
    </tenant>
 </tenants>
</esc_datamodel>
```

（注） VIMが再び使用可能になったら、ディザスタリカバリシナリオで古い展開を削除する必要があることを忘れないでください。

この API のさらなる用途は、前述のリカバリ API を介した VM の再起動前に実行する永続ボリュームの UUID の更新です。これには、以前のバージョンの ESC のように、VM グループを削除して再度追加する必要がないという利点があります。ペイロードの例を次に示します。

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <tenants>
    <tenant>
       <name>my-tenant</name>
       <deployments>
          <deployment>
             <name>my-dep</name>
             <vm_group>
                <name>my-vm</name>
                <bootup_time>1800</bootup_time>
                <volumes>
                   <volume>
                      <name>new-volume</name>
                      <volid>1</volid>
                      <bus>ide</bus>
                      <type>lvm</type>
                   </volume>
                   <volume nc:operation="delete">
                      <name>old-volume</name>
                      <volid>1</volid>
                   </volume>
                </volumes>
             </vm_group>
          </deployment>
       </deployments>
    </tenant>
 </tenants>
</esc_datamodel>
```

VM の手動回復でサポートされる VM の状態とサービスの組み合わせ

recovery-vm-action の API は、自動と手動の両方のリカバリタイプに適用されますが、特定の VM の状態とサービスに限ります。次のテーブルに詳細を示します。一般に、展開、サービス更新、展開解除、およびリカバリの間は、ESC は手動リカバリアクションを拒否します。

VM 状態	サービス ステート	recovery-vm-action
動作中（Alive）	ACTIVE	サポート対象
接続中（Alive）	ERROR	サポート対象
ERROR	ERROR	サポート対象

展開の手動回復

モニタリングパラメータを使用しないリカバリ

ESCは、サービスレベルでのVMの手動回復、つまり展開全体の回復をサポートします。サービスが正常に展開された後、VMの障害が原因でサービスがエラー状態に移行することがあります。ESCは、障害が発生したこれらのVMを手動で回復することも、展開回復要求によって展開全体を回復することもできます。VM単独の手動回復については、手動回復（344ページ）を参照してください。

展開の手動回復用 API

NETCONF API と REST API を使用して手動リカバリを実行できます。

手動回復要求は、事前定義されたリカバリアクションを任意のアクションに上書きするように設定できます。

（注） 展開回復後のサービスアクティブ通知はありません。展開のサービス状態がアクティブかどうかを確認するには、*esc_nc_cli --user <username> --password <password> get esc_datamodel* などのクエリを実行する必要があります。

設定可能な手動回復中にフェールオーバーが発生した場合、手動回復は事前定義されたリカバリアクションで再開されます。

展開の移行では、常にデフォルトのリカバリポリシーを使用する必要があります。LCSベースのリカバリでは、VM/VNF手動リカバリのリカバリアクションを指定しないでください。モニターの有効化オプションと設定可能な手動リカバリオプションは同時に使用できません。

NETCONF API

```
svc-action RECOVER tenant-name deployment-name [xmlfile]
```

API を使用してリカバリを実行するには、`esc_nc_cli` にログインします。

REST API

```
POST /v0/{internal_tenant_id}/deployments/service/{internal_deployment_id}
Content-Type: application/xml
Accept: application/json
Callback: http://172.16.0.1:9010/
Callback-ESC-Events: http://172.16.0.1:9010/
<service_operation xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
   <operation>recover</operation>
</service_operation>
```

値は次のとおりです。

internal_tenant_id：システム管理者のテナント ID またはテナント名。

internal_deployment_id：展開名。

展開の手動回復でサポートされる VM の状態とサービスの組み合わせ

svc-action RECOVER の API は、自動と手動の両方のリカバリタイプに適用されますが、特定の VM の状態とサービスに限ります。次のテーブルに詳細を示します。一般に、展開、サービスの更新、展開解除、およびリカバリの間は、ESC は手動回復アクションを拒否します。

（注） サービスがアクティブまたはエラー状態の場合、ESC は VM レベルのリカバリ要求を受け入れます。

サービスリカバリ要求後にすべての VM が動作状態になっている場合、NB に通知は送信されません。

VM 状態	サービス ステート	svc-action RECOVER
ERROR	ERROR	サポート対象
ERROR	ERROR	サポート対象

モニタリングパラメータによるリカバリの有効化

手動回復では、モニタリングパラメータに応じて VM をリカバリできます。VM がエラー状態の場合は、エラー状態の VM を動作状態に戻すためのモニタリングパラメータを設定します。VM が回復すると、ESC は RECOVERY_CANCELLED 通知を送信します。VM が動作状態に復帰しない場合、リカバリプロセスがトリガーされます。詳細については、「手動回復」を参照してください。

NETCONF API

```
svc-action SET_MONITOR_AND_RECOVER <tenant-name> <dep-name>
```

リカバリ通知：

```
===== SEND NOTIFICATION STARTS =====
 WARN  Type: VM_RECOVERY_INIT
 WARN  Status: SUCCESS
 WARN  Status Code: 200
 WARN  Status Msg: Recovery with enabling monitor first event for VM Generated ID
[dep-resource_g1_0_74132737-d0a4-4ef0-bd9e-86465c1017bf] triggered.
```

（注） モニタリングパラメータで有効化されるリカバリは、サービスレベルでの手動回復専用です。

monitor_on_error パラメータにより、エラー状態にある VM の継続的なモニタリングが設定されます。

```
<recovery_policy>
                        <recovery_type>AUTO</recovery_type>
                        <action_on_recovery>REBOOT_ONLY</action_on_recovery>
                        <max_retries>1</max_retries>
                        <monitor_on_error>true</monitor_on_error>
                    </recovery_policy>
```

デフォルト値は false です。

false を指定すると、エラー状態にある VM のモニタリングは**設定解除**されます。

true を指定すると、エラー状態にある VM のモニタリングは**設定**されます。後から VM 稼働イベントが発生した場合（VM_RECOVERY_COMPLETE の後）、VM は稼働状態に戻ります。

リカバリポリシーと再展開ポリシー

ESC は、ポリシー駆動型フレームワークを使用して、展開のライフサイクルステージに基づいてアクションを実行します。展開は、そのライフサイクルを通じて複数のステージで構成されます。各ライフサイクルステージ（LCS）は、条件に関連付けられています。条件は、定義済みのアクションまたはカスタムスクリプトに関連付けられています。それらの条件とアクションは、データモデルの policy タグ内で指定されます。ポリシー駆動型フレームワークの詳細については、ポリシー駆動型データモデル （199 ページ）を参照してください。

ESC のリカバリおよび再展開のワークフローはポリシー駆動型です。VNF が展開されると、リカバリおよび再展開のポリシーが展開データモデルで指定されます。これらのポリシーは、VM または VNF のライフサイクルステージに基づいており、アクションが関連付けられています。

展開データモデルの作成時に、次のポリシーを指定できます。

- リカバリポリシー：リカバリポリシーは、VM ライフサイクル、つまり単一の VM のリカバリ用です。事前定義されたアクションに基づいて、VM が再起動または再展開されます。ユーザは、ポリシーフレームワークを使用せずにリカバリを実行できます。リカバリポリシー （343 ページ）を参照してください。
- 再展開ポリシー：再展開ポリシーは、展開ライフサイクル全体、つまり展開内のすべての VM グループに適用されます。事前定義された一連のアクションに基づいて、ホストが無効になり、VM が展開内で回復されます。

最大試行回数の後に VM リカバリが失敗すると、ESC はホストを無効にし、展開内のすべての VM の再展開をトリガーします。すべての VM が古いホストから展開解除され、新しいホストに再展開されます。

ESC は、最初に障害が発生した VM の再展開をサポートします。再展開中は、障害が発生した VM が最初に回復され、障害が発生していない VM は再展開のためにキューに入れられます。

リカバリポリシー（ポリシーフレームワークを使用）

ESCはポリシー主導型フレームワークのデータモデルを使用したVMのリカバリをサポートしています。リカバリは、VM展開のライフサイクルステージと事前定義されたアクションに基づいています。

自動回復および手動回復については、リカバリポリシー（343ページ）を参照してください。

次の表に、さまざまなライフサイクルステージで実行される事前定義されたアクションを示します。

事前定義されたアクション名	範囲	説明
SET_RECOVERY::REBOOT_ONLY	展開	すべてのVMグループ（展開内）またはVM（VMグループ内）のリカバリアクションをREBOOT_ONLYに設定します。
SET_RECOVERY::REBOOT_THEN_REDEPLOY	展開	すべてのVMグループ（展開内）またはVM（VMグループ内）のリカバリアクションをREBOOT_THEN_REDEPLOYに設定します。
SET_RECOVERY::REDEPLOY_ONLY	展開	すべてのVMグループ（展開内）またはVM（VMグループ内）のリカバリアクションをREDEPLOY_ONLYに設定します。
SET_RECOVERY::RESET_STATE_THEN_REBOOT	配置	すべてのVMグループ（展開内）またはVM（VMグループ内）のリカバリアクションをRESET_STATE_THEN_REBOOTに設定します（OpenStackのみ）。

サポートされる条件と事前定義されたアクションの組み合わせ

次の表に、ポリシーフレームワークを使用したリカバリおよび再展開ポリシーでサポートされるLCS条件とそのアクションを示します。ポリシー主導型フレームワークの詳細については、リカバリポリシーと再展開ポリシー（350ページ）を参照してください。

条件	事前定義されたアクション	説明
LCS::PRE_DEPLOY ：展開で VM を展開する直前に発生します。 LCS:: POST_DEPLOY_ALIVE ：展開がアクティブになった直後に発生します。	• SET_RECOVERY::REBOOT_ONLY ：すべての VM グループ（展開内）または VM（VM グループ内）のリカバリアクションを REBOOT_ONLY に設定します。 • SET_RECOVERY:: REBOOT_THEN_REDEPLOY ：すべての VM グループ（展開内）または VM（VM グループ内）のリカバリアクションを REBOOT_THEN_REDEPLOY に設定します。 • SET_RECOVERY:: REDEPLOY_ONLY ：すべての VM グループ（展開内）または VM（VM グループ内）のリカバリアクションを REDEPLOY_ONLY に設定します。 • SET_RECOVERY_ REDEPLOY:: SERIALIZED ：展開のリカバリをキューに入れます。つまり、現在進行中のリカバリが完了するまで、新しいリカバリは開始されません。	リカバリ用に事前定義されたアクションのいずれかを選択します。 SET_RECOVERY_ REDEPLOY::SERIALIZED は、 DROP_RECOVERIES アクションが使用される場合に選択します。これは、再展開が失敗した場合、元のホストに VM を保持する必要があることを意味します。選択しない場合、DROP_RECOVERIES アクションは使用できません。
LCS::DEPLOY_ERR ：展開が失敗した直後に発生します。	DISABLE_HOST ：展開または VM が使用しているホストを無効にします。	-

条件	事前定義されたアクション	説明
LCS::POST_ DEPLOY::VM_RECOVERY _ERR ：1つのVMのリカバリが失敗した直後に発生します	DISABLE_HOST ：展開またはVMが使用しているホストを無効にします。 REDEPLOY_ALL:: DISABLE_HOST ：VMが使用しているホストを無効にしてから、（展開内の）すべてのVMまたはそのホスト上のすべてのVMの再展開をトリガーします。	必要に応じて、DISABLE_HOSTを選択します。 REDEPLOY_ALL:: DISABLE_HOST ホストの無効化後に再展開が必要かどうかを選択します。 DISABLE_HOSTとREDEPLOY_ALL::DISABLE_HOSTは重複するため、一緒にすることはできません。
LCS::POST_ DEPLOY::VM_RECOVERY_ REDEPLOY_ERR ：1つのVMの再展開が失敗した直後に発生します。	• DISABLE_HOST ：展開またはVMが使用しているホストを無効にします。 • DROP_RECOVERIES ：展開内で保留中のすべてのリカバリをドロップします。	DISABLE_HOST DISABLE_HOSTが必要かどうかを選択します。 再展開が失敗した後にVMを元のホストに保持する必要がある場合は、DROP_RECOVERIESを選択します。 DROP_RECOVERIESを選択する場合は、SET_RECOVERY_ REDEPLOY:: SERIALIZEDアクションが完了していることを確認します。

再展開ポリシー

再展開ポリシーは、ポリシー駆動型フレームワークの一部です。このフレームワークを使用して、特定のライフサイクル条件用に事前定義されたアクションを指定できます。ESCポリシー駆動型フレームワークの詳細については、ポリシー駆動型データモデル（199ページ）を参照してください。

再展開ポリシーは、最大試行回数後にVMリカバリが失敗したときに呼び出されます。ESCはホストを無効にし、展開内のすべてのVMの再展開をトリガーします。すべてのVMが古いホストから展開解除され、新しいホストに再展開されます。ライフサイクルステージ（LCS）と事前定義されたアクションの組み合わせに基づいて、VMが再展開されます。再展開ポリシーは、展開全体に適用されます。

ポリシーデータモデルでは、次のライフサイクル条件とアクションの組み合わせを使用できます。

（注） ESC は、何も選択されていない場合、デフォルトのリカバリアクション REBOOT_THEN_REDEPLOY を使用します。

再展開ポリシーのデータモデルの例を次に示します。

```
<tenants>
        <tenant>
            <name>xyz-redeploy-ten-0502</name>
            <deployments>
                <deployment>
                    <name>dep</name>
                    <policies>
                        <policy>
                            <name>1</name>
                            <conditions>
                                <condition>
                                    <name>LCS::PRE_DEPLOY</name>
                                </condition>
                            </conditions>
                            <actions>
                                <action>
                                    <name>SET_RECOVERY::REBOOT_THEN_REDEPLOY</name>
                                    <type>pre-defined</type>
                                </action>
                                <action>
                                    <name>SET_RECOVERY_REDEPLOY::SERIALIZED</name>
                                    <type>pre-defined</type>
                                </action>
                            </actions>
                        </policy>
                        <policy>
                            <name>2</name>
                            <conditions>
                                <condition>
                                    <name>LCS::POST_DEPLOY_ALIVE</name>
                                </condition>
                            </conditions>
                            <actions>
                                <action>
                                    <name>SET_RECOVERY::REBOOT_ONLY</name>
                                    <type>pre-defined</type>
                                </action>
                            </actions>
                        </policy>
                        <policy>
                            <name>3</name>
                            <conditions>
                                <condition>
                                    <name>LCS::DEPLOY_ERR</name>
                                </condition>
                            </conditions>
                            <actions>
                                <action>
                                    <name>DISABLE_HOST</name>
                                    <type>pre-defined</type>
                                </action>
```

```
                    </actions>
                </policy>
                <policy>
                    <name>4</name>
                    <conditions>
                        <condition>
                            <name>LCS::POST_DEPLOY::VM_RECOVERY_ERR</name>
                        </condition>
                    </conditions>
                    <actions>
                        <action>
                            <name>REDEPLOY_ALL::DISABLE_HOST</name>
                            <type>pre-defined</type>
                        </action>
                    </actions>
                </policy>
                <policy>
                    <name>5</name>
                    <conditions>
                        <condition>
                        <name>LCS::POST_DEPLOY::VM_RECOVERY_REDEPLOY_ERR</name>

                        </condition>
                    </conditions>
                    <actions>
                        <action>
                            <name>DISABLE_HOST</name>
                            <type>pre-defined</type>
                        </action>
                        <action>
                            <name>DROP_RECOVERIES</name>
                            <type>pre-defined</type>
                        </action>
                    </actions>
                </policy>
            </policies>
            <vm_group>
                <name>Group1</name>
                <image>xyz-redeploy-img-0502</image>
                <flavor>xyz-redeploy-flv-0502</flavor>
                <recovery_policy>
                    <max_retries>1</max_retries>
                </recovery_policy>
        ......
        ......
      </deployment>
     </deployments>
  </tenant>
</tenants>
```

サポート対象のライフサイクルステージ（LCS）

条件名	範囲	説明
LCS::PRE_DEPLOY	展開	展開のVMを展開する直前に発生します。
LCS::POST_DEPLOY_ALIVE	展開	展開がアクティブになった直後に発生します。
LCS::DEPLOY_ERR	展開	展開が失敗した直後に発生します。

LCS::POST_DEPLOY:: VM_RECOVERY_ERR	展開	1 つの VM のリカバリが失敗した直後に発生します （これは展開レベルで指定され、すべての VM グループに適用されます）。
LCS::POST_DEPLOY:: VM_RECOVERY_REDEPLOY_ERR	展開	1 つの VM の再展開が失敗した直後に発生します （これは展開レベルで指定され、すべての VM グループに適用されます）。

サポートされている定義済みアクション

事前定義されたアクション名	範囲	説明
DISABLE_HOST	展開	展開または VM が使用しているホストを無効にします。
REDEPLOY_ALL::DISABLE_HOST	展開	VM が使用しているホストを無効にしてから、（展開内の）すべての VM またはそのホスト上のすべての VM の再展開をトリガーします。
DROP_RECOVERIES	展開	展開内で保留中のすべてのリカバリをドロップします。
SET_RECOVERY_REDEPLOY::SERIALIZED	展開	展開のリカバリをキューに入れます。つまり、現在進行中のリカバリが完了するまで、新しいリカバリは開始されません。

再展開回数の制限

Cisco Elastic Services Controller（ESC）は、次のパラメータを使用して再展開の回数を制限します。

- max_redep：再展開の最大数を制限します。デフォルトでは、max_redep の値は -1 です。これは再展開の最大数に制限がないことを示します。この値は、bootvm.py 引数または REST API を使用して変更できます。
- redep_count：現在の再展開の数で構成されます。redep_count は、再展開の成功または失敗に関係なく、再展開後に 1 ずつ自動的に増加します。

（注） 再展開の制限は次のとおりです。

- REDEPLOY_ALL::DISABLE_HOST ポリシーによってトリガーされる再展開。
- 単一の VIM 設定のみの展開。

次の場合、Cisco Elastic Services Controller（ESC）が再展開を実行します。

- 再展開の最大数がデフォルト値の -1 に設定されている場合（max_redep = -1）。
- 現在の再展開の数が再展開の最大数よりも少ない場合（redep_count < max_redep）、ESC が再展開を実行し、再展開の完了後に再展開数を 1 増やします。

再展開の回数が再展開の最大数以上の場合（redep_count >= max_redep）、ESC は再展開を実行しません。

各値は、bootvm.py パラメータと REST API を使用して設定できます。

bootvm.py パラメータの使用

次の行を含む esc_params.conf ファイルで max_redep 値を指定します。default.max_redep = 3

コマンド `bootvm.py ... --esc_params_file <path_to_file>/esc_params.conf ...` を実行します。

REST API の使用

次の API を使用して、redep_count パラメータを取得およびリセットできます。

- redep_count の現在の値を取得するには、次の手順を実行します。

  ```
  GET http://<ESC IP>:8080/ESCManager/v0/systemstate/redep_count
  ```
- redep_count をリセットするには、次の手順を実行します。

  ```
  POST http://<ESC IP>:8080/ESCManager/v0/systemstate/redep_count/reset
  ```

REST API を使用して max_redep 値を取得および変更することもできます。

- max_redep の現在の値を取得するには、次の手順を実行します。

  ```
  GET http://<ESC IP>:8080/ESCManager/v0/config/default/max_redep
  ```
- max_redep 値を変更するには、次の手順を実行します。

  ```
  PUT http://<ESC IP>:8080/ESCManager/v0/config/default/max_redep/<value>
  ```

  ここで、 <value> は次のいずれかです。

  -1：制限なしのデフォルト値。

  0：再展開を許可しない場合。

  1 以上（> 0）：許可される再展開の最大数を指定します。

これらの値は、ESCADM ツールを使用して設定することもできます。ESCADM ツールの詳細については、Elastics Services Controller インストールおよびアップグレードガイド [英語] を参照してください。

再展開ポリシーの詳細については、再展開ポリシー （353 ページ）を参照してください。

再展開の制限により再展開されない VM は、エラー状態に移行します。ESC では、各 VM でモニタリング操作を有効にすることで、エラー状態にある VM を手動で回復します。

エラー状態にある単一の VM でモニタリング操作を有効にするには、次の手順を実行します。

```
POST http://<ESC IP>:8080/ESCManager/v0/<internal-tenant-id>/deployments/vm/<vm-name> {
 "operation" : "enable_monitoring" }
```

esc_nc_cli コマンドを使用してモニタリングを有効にすることもできます。

```
esc_nc_cli --user <username> --password <password> vm-action ENABLE_MONITOR <generated
vm name>
```

手動リカバリプロセスの一環として、モニタリング操作の有効化により VM がエラー状態から稼働状態に移行します。VM の手動リカバリが失敗した場合、自動リカバリがトリガーされます。

展開内の VM（エラー状態）のモニタリング操作を有効にするには、次の手順を実行します。

```
POST http://<ESC
IP>:8080/ESCManager/v0/<internal-tenant-id>/deployments/service/<internal-deployment-id>
 { "operation" : "enable_monitoring" }
```

esc_nc_cli コマンドを使用してモニタリングを有効にすることもできます。

```
esc_nc_cli --user <username> --password <password> svc-action ENABLE_MONITOR <tenant>
<dep name>
```

手動リカバリプロセスの一環として、モニタリング操作の有効化により展開内のすべての VM がエラー状態から稼働状態に移行します。手動リカバリが失敗した場合、展開内のすべての VM に対して自動リカバリがトリガーされます。

詳細については、モニタリング操作（320 ページ）、および「リカバリポリシー」を参照してください。

ホストの有効化と無効化

NETCONF API および REST API を使用して、OpenStack でホストを有効または無効にできます。ホストは、VNF のリカバリまたは再展開のシナリオ中に無効にすることもできます。

（注） VMware vCenter でのホストの有効化と無効化はサポートされていません。

複数の OpenStack VIM がある ESC で NETCONF API および REST API を使用して、デフォルト以外の VIM でホストを有効または無効にすることはできません。

NETCONF の使用

```
/opt/cisco/esc/esc-confd/esc-cli/esc_nc_cli --user <username> --password <password>
host-action < ENABLE | DISABLE > <host-name>
```

ペイロードは次のとおりです。

```
<hostAction xmlns="http://www.cisco.com/esc/esc">
  <actionType>ENABLE/DISABLE</actionType>
  <hostName>my-server</hostName>
</hostAction>
```

値は次のとおりです。

- actionType は ENABLE または DISABLE です

- hostName はターゲットホストのホスト名または UUID です

REST の使用

```
POST /v0/hosts/{hostName}/disable
POST /v0/hosts/{hostName}/enable
GET /v0/hosts/{hostName}/status
```

ホストの有効化

ホストを有効にすることで、無効化されたホストを OpenStack に戻し、新しい VM インスタンスをそのホストに展開します。

NETCONF 通知の例は次のとおりです。

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-03-30T15:04:05.95+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Host action successful</status_message>
    <vm_source>
      <hostname>my-server</hostname>
    </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>HOST_ENABLE</type>
    </event>
  </escEvent>
</notification>
```

サンプル REST 通知は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <host_action_event xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <event_type>HOST_ENABLE</event_type>
    <host_name>my-server</host_name>
    <message>Host action successful</message>
</host_action_event>
```

ホストの無効化

VNF の再展開中にホストを無効にし、その展開内のすべての VM に対してホストベースの再展開をトリガーします。これにより、再展開された VM が別のホストにあることが保証されます。ホストが正常に動作していない場合は、ホストを無効にすることもできます。無効になったホストは OpenStack から削除されるため、新しいインスタンスは展開されません。

NETCONF 通知の例は次のとおりです。

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-03-30T15:03:48.121+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Host action successful</status_message>
    <vm_source>
      <hostname>my-server</hostname>
    </vm_source>
    <vm_target>
  </vm_target>
```

```
    <event>
      <type>HOST_DISABLE</type>
    </event>
  </escEvent>
</notification>
```

サンプル REST 通知は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<host_action_event xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <event_type>HOST_DISABLE</event_type>
  <host_name>my-server</host_name>
  <message>Host action successful</message>
</host_action_event>
```

通知とイベント

修復中に ESC によって次の通知が生成されます。

- VM_RECOVERY_INIT
- VM_RECOVERY_DEPLOYED
- VM_RECOVERY_UNDEPLOYED
- VM_RECOVERY_COMPLETE
- VM_RECOVERY_CANCELLED
- VM_RECOVERY_REBOOT

これらの通知は、ワークフローに基づいて生成されます。各通知には、通知がトリガーされる展開に関する詳細情報が含まれます。すべてのリカバリは VM_RECOVERY_INIT で始まり、VM_RECOVERY_COMPLETE で終わります。

VM のリカバリ中、リカバリ待機時間内に VM が正常に戻ると、実行するリカバリアクションがないため、VM_RECOVERY_CANCELLED 通知が送信されます。 リカバリ待機時間が経過すると、リカバリアクションがトリガーされます。リカバリが完了すると、ESC は成功または失敗の通知（VM_RECOVERY_REBOOT 通知など）を送信します。

次の表に、さまざまなシナリオと、イベントごとに生成される通知を示します。

シナリオ	通知
VM Alive 後の ESC-NORTHBOUND リカバリコールフロー：再起動	ノースバウンドから ESC に展開要求が送信されると、ESC は VM を展開し、受信したすべての VM Alive をモニタするように KPI を設定します。次の NETCONF 通知がトリガーされます。 <type>SERVICE_ALIVE</type> <status>SUCCESS</status> ESC が VM Down イベントを受信すると、次の NETCONF 通知がトリガーされます。 <type>VM_RECOVERY_INIT</type> <status>SUCCESS</status> ESC は VM でハード再起動を実行し、VM Alive イベントをブート時間内に受信します。 <type>VM_RECOVERY_COMPLETE</type> <status>SUCCESS</status> ESC は、再起動によるリカバリの試行中にエラーを受信します。次の NETCONF 通知がトリガーされます。 <type>VM_RECOVERY_COMPLETE</type> <status>FAILURE</status>

シナリオ	通知

VM Alive 後の ESC-NORTHBOUND リカバリコールフロー：展開解除/再展開	ノースバウンドから ESC に展開要求が送信されると、ESC は VM を展開し、受信したすべての VM Alive をモニタするように KPI を設定します。次の NETCONF 通知がトリガーされます。
	
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	
	ESC が VM Down イベントを受信すると、次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	
	ESC は再起動による VM の回復に失敗し、展開解除して再展開することで回復を進めます。
	
	モニタリングの設定を解除し、VM の展開を解除します。
	
	次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_UNDEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	ESC は VM を展開し、VM Alive イベントをモニタするように KPI を設定し 、次の NETCONF 通知をトリガーします。
	
	`<type>VM_RECOVERY_DEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	ESC は VM Alive イベントを受信し、次の NETCONF 通知をトリガーします。
	
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>SUCCESS</status>`

シナリオ	通知

ESC-NORTHBOUND リカバリコールフローによる複数回のリカバリの試行	ノースバウンドから ESC に展開要求が送信されると、ESC は VM を展開し、受信したすべての VM Alive をモニタするように KPI を設定します。次の NETCONF 通知がトリガーされます。
	
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	
	ESC が VM Down イベントを受信すると、次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	
	ESC は、VM Alive イベントを受信するまで、展開解除と再展開によってVMを回復できません。リカバリの最大試行回数に達するまで、指定されたブート時間リカバリを試行し続けます。
	
	モニタリングの設定を解除し、VM の展開を解除します。
	
	次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_UNDEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	ESC は VM を展開し、VM Alive イベントをモニタするように KPI を設定します。
	
	次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_DEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	ESC は VM Alive イベントを受信し、次の NETCONF 通知をトリガーします。
	
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>SUCCESS</status>`

シナリオ	通知
VM Alive 前の ESC-NORTHBOUND リカバリコールフロー：展開解除/再展開	ノースバウンドから ESC に展開要求が送信されると、ESC は VM を展開し、受信したすべての VM Alive をモニタするように KPI を設定します。 ESC は、展開後に VM Alive イベントを受信しません。リカバリは、VM の展開と再展開によって実行されます。 次の NETCONF 通知がトリガーされます。 <type>VM_RECOVERY_INIT</type> <status>SUCCESS</status> ESC はモニタリングの設定を解除し、VM の展開を解除します。 次の NETCONF 通知がトリガーされます。 <type>VM_RECOVERY_UNDEPLOYED</type> <status>SUCCESS</status> ESC は VM を展開し、VM Alive イベントをモニタするように KPI を設定し 、次の NETCONF 通知をトリガーします。 <type>VM_RECOVERY_DEPLOYED</type> <status>SUCCESS</status> ESC は VM Alive イベントを受信し、次の NETCONF 通知をトリガーします。 <type>VM_RECOVERY_COMPLETE</type> <status>SUCCESS</status>

シナリオ	通知

VM Alive 後の ESC-NORTHBOUND リカバリコールフローのエラーパス：展開解除/再展開	ノースバウンドから ESC に展開要求が送信されると、ESC は VM を展開し、受信したすべての VM Alive をモニタするように KPI を設定します。次の NETCONF 通知がトリガーされます。
	
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	
	ESC が VM Down イベントを受信すると、次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	
	ESC は再起動による VM の回復に失敗し、展開解除して再展開することで回復を進めます。
	
	モニタリングの設定を解除し、VM の展開を解除します。
	
	次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_UNDEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	ESC がエラーを受信した場合、またはリカバリの最大試行回数に達した場合。
	
	次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>FAILURE</status>`

シナリオ	通知

VM Alive 前の ESC-NORTHBOUND リカバリコールフローのエラーパス：展開解除/再展開	ノースバウンドから ESC に展開要求が送信されると、ESC は VM を展開し、受信したすべての VM Alive をモニタするように KPI を設定します。次の NETCONF 通知がトリガーされます。
	
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	
	ESC が VM Down イベントを受信すると、次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	
	ESC はモニタリングの設定を解除し、VM の展開を解除します。リカバリは、展開解除してから再展開することで実行されます。
	
	次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_UNDEPLOYED</type>`
	`<status>SUCCESS</status>`
	
	ESC がエラーを受信した場合、またはリカバリの最大試行回数に達した場合。
	
	次の NETCONF 通知がトリガーされます。
	
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>FAILURE</status>`
	
	`<type>SERVICE_ALIVE</type>`
	`<status>FAILURE</status>`

シナリオ	通知
VM Alive 後の ESC-NORTHBOUND リカバリコールフロー：VM_RECOVERY_CANCELLED	ノースバウンドから ESC に展開要求が送信されると、ESC は VM を展開し、受信したすべての VM Alive 通知をモニタするように KPI を設定します。次の NETCONF 通知がトリガーされます。 <type>SERVICE_ALIVE</type> <status>SUCCESS</status> ESC が VM Down イベントを受信すると、次の NETCONF 通知がトリガーされます。 <type>VM_RECOVERY_INIT</type> <status>SUCCESS</status> リカバリ待機時間中に VM が正常に戻ると、VM_RECOVERY_CANCELLED 通知が送信されます。リカバリアクションは実行されません。 <type>VM_RECOVERY_CANCELLED</type> <status>SUCCESS</status>
VM Alive 後の ESC-NORTHBOUND リカバリコールフロー：再起動	ノースバウンドから ESC に展開要求が送信されると、ESC は VM を展開し、受信したすべての VM Alive 通知をモニタするように KPI を設定します。次の NETCONF 通知がトリガーされます。 <type>SERVICE_ALIVE</type> <status>SUCCESS</status> ESC が VM Down イベントを受信すると、次の NETCONF 通知がトリガーされます。 <type>VM_RECOVERY_INIT</type> <status>SUCCESS</status> ESC は VM でハード再起動を実行し、再起動通知を送信します。 <type>VM_RECOVERY_REBOOT</type> <status>SUCCESS</status> VM Alive イベントは、ブート時間内に受信されます。 <type>VM_RECOVERY_COMPLETE</type> <status>SUCCESS</status>

シナリオ	通知

VM Alive 後の ESC-NORTHBOUND リカバリコールフローのエラーパス：再起動	ノースバウンドから ESC に展開要求が送信されると、ESC は VM を展開し、受信したすべての VM Alive 通知をモニタするように KPI を設定します。次の NETCONF 通知がトリガーされます。
	`<type>SERVICE_ALIVE</type>`
	`<status>SUCCESS</status>`
	ESC が VM Down イベントを受信すると、次の NETCONF 通知がトリガーされます。
	`<type>VM_RECOVERY_INIT</type>`
	`<status>SUCCESS</status>`
	次に、ESC が再起動通知を送信します。
	`<type>VM_RECOVERY_REBOOT</type>`
	`<status>FAILURE</status>`
	ESC は、再起動によるリカバリの試行中にエラーを受信します。
	次の NETCONF 通知がトリガーされます。
	`<type>VM_RECOVERY_COMPLETE</type>`
	`<status>FAILURE</status>`

第 VI 部

ESC ポータル

第 41 章

使用する前に

ESC ポータルへのログイン

（注）
- ESC ポータルはデフォルトで有効になっています。インストール時に ESC ポータルが無効になっていないことを確認する必要があります。ESC ポータルの有効化または無効化の詳細については、Cisco ESC インストールおよびアップグレードガイド [英語] の「Installing ESC」を参照してください。
- ESC ポータルへの初回ログイン時に、デフォルトパスワードの変更を求められます。

ESC ポータルにログインするには、次の手順を実行します。

始める前に

- ESC のインスタンスを登録します。ESC インスタンスの登録の詳細については、Cisco Elastic Services Controller インストールおよびアップグレードガイド [英語] を参照してください。
- ユーザ名とパスワードを取得していることを確認します。

手順

ステップ 1 Web ブラウザを使用して、ESC の IP アドレスを入力します。

例：

たとえば、ESC の IP アドレスが 192.0.2.254 の場合は、次のように入力します。

https://192.0.2.254 [https 経由でログイン]。ポータルは、デフォルトのセキュリティポート 443 で実行されます。

セキュリティ アラート メッセージが表示されます。

ステップ 2 [はい（Yes）] をクリックしてセキュリティ証明書を受け入れます。ログイン ページが表示されます。

ステップ 3 ユーザ名とパスワードを入力して、[ログイン（Login）] をクリックします。

初回ログイン時には、ログインページが再表示され、パスワードの変更を求められます。

ステップ 4 [古いパスワード（Old Password）] フィールドに古いパスワードを入力し、[新しいパスワード（New Password）] および [パスワードの確認（Confirm Password）] フィールドに新しいパスワードを入力します。

ステップ 5 [パスワードの更新（Update Password）] をクリックするか、Enter を押します。

（注）
- ポータルが応答しなくなった場合は、ESCADM ツールから escadm portal restart を実行してポータルを再起動します。
- ESC ポータルは 1 人のユーザのみをサポートします。
- 現在、事前インストールされた自己署名証明書は HTTPS をサポートしています。ESC ポータルの処理を進める前に、ユーザは自己署名証明書を確認する必要があります。
- HTTPS 通信モードでは、OpenStack によって返される URL プロトコルタイプが HTTPS ではない場合、VNF コンソールへのアクセスが無効になることがあります。セキュリティ上の理由から、HTTPS で実行している間は、安全性の低い通信は拒否されます。

ESC パスワードの変更

初回ログイン時にデフォルトのパスワードを変更する必要があります。ポータルでは、この手順をバイパスすることはできず、デフォルトのパスワードを変更するまでこのページに戻ります。パスワードを初めて変更した後、このセクションで説明されている手順を使用してパスワードを変更できます。また、ユーザが複数のブラウザまたはタブを持っている場合、または同じユーザが 2 台以上のコンピュータからログインしている場合、ユーザの 1 人がパスワードを変更すると、全員がログオフされ、新しいパスワードを再入力するように求められます。ポータルで20分以上アイドル状態になると、ユーザはログアウトされます。ポータル環境ファイルでユーザのアイドルタイムアウトを設定できます。パスワードを忘れた場合は、パスワードをリセットすることもできます。

ここでは、ポータルパスワードの変更方法について説明します。

ESC ポータルパスワードの変更

ポータルから既存の ESC ポータルパスワードを変更するには、次の手順を実行します。

手順

ステップ 1 ユーザ名とパスワードを使用して ESC ポータルにログインします。

ステップ 2 画面右上隅のユーザーアイコンをクリックします。

ステップ 3 [アカウント設定（Account Settings）] をクリックします。アカウント情報とパスワードを更新するページが表示されます。

ステップ 4 [パスワードの更新（Update Password）]をクリックします。

ステップ 5 [古いパスワード（Old Password）] フィールドに古いパスワードを入力し、[新しいパスワード（New Password）] および [パスワードの確認（Confirm Password）] フィールドに新しいパスワードを入力します。

ステップ 6 [作成（CREATE）] をクリックします。

次のタスク

CLI などを使用してパスワードを変更する方法については、『Cisco Elastic Services Controller Install and Upgrade Guide』を参照してください。

ESC ポータルダッシュボード

Cisco Elastic Services Controller ダッシュボードには、テナント、フレーバー、イメージ、展開、着信要求、通知、システムの正常性の視覚的なインジケータなど、管理対象のすべての ESC リソースが表形式で表示されます。次のダッシュボード要素は、データとシステムの正常性を経時的に追跡、監視、および診断するのに役立ちます。

ダッシュボードは、ダッシュボードを表示するシステムが専用のシステムであり、ポータルサーバを実行しているシステムとは異なる場合があるモニタリング デスク コンテキストで使用するのが最適です。ダッシュボードシステムは、ポータルサーバを実行しているシステムをブラウザでポイントする必要があります

異常なスパイクやアクティビティの低下に気付いた場合は、ネットワーク上で通信障害や停電が発生して調査する必要があります。

HA スイッチオーバーの場合、ユーザはログアウトしてからログインしてポータルリソースを表示する必要があります。

次の表に、ポータルで確認できる詳細を示します。

（注） これらのタスクは、NB API を使用して実行することもできます。詳細については、Elastic Services Controller NB API （9 ページ）を参照してください。

表 *29* : ポータルの詳細

タスク	移動方法	説明
ダッシュボードを表示する	[ダッシュボード（Dashboard）] を選択します。	すべての管理対象 ESC リソース、通知、システム設定、およびシステムの正常性の概要が表示されます。
通知を表示する	[通知（Notifications）] を選択するか、ポータルの右上隅にある通知アイコンをクリックします。	ポータルで ESC から受信した通知が表示されます。
VNF を展開する	[展開（Deployments）] を選択します。 **重要** フォームを使用して VMware vCenter に VNF を展開するには、「フォームを使用した展開」を参照してください。	VNF を展開します。 ドラッグアンドドロップ機能を使用すると、既存の展開データモデルを取得し、展開テーブルにファイルをドラッグして再利用できます。テーブルのツールバーにある [XML のアップロード（Upload XML）] を使用して、ファイルシステムから適切なファイルを参照することもできます。 （注） XML ファイルのみ受け入れられます。 ドラッグアンドドロップ機能は、現時点では REST コールを実行し、NETCONF コールは実行しません。
既存の展開を表示する（OpenStack と VMware vCenter の両方）	[展開（Deployments）] を選択し、テーブルから展開を選択します。 • [VMグループの表示（View VM Groups）] をクリックします。モニタリング、スケーリングなどの詳細、およびその他の情報は、対応するタブで確認できます。	現在展開されている展開の概要が表示されます。展開の名前とステータス、およびその展開に展開されている VM の数を表示できます。

タスク	移動方法	説明
VIM を表示する	[リソース（Resources）] > [VIM（VIMs）] の順に選択します。	VIM ID、VIM のタイプ、VIM のステータス、プロパティ、および VIM ユーザを含む VIM のリストが表示されます。
テナントを表示する（OpenStack のみ）	[リソース（Resources）] > [テナント（Tenants）] の順に選択します。	テナントの名前、説明、および ID を含む、テナントのリストが表示されます。 **重要** ESC は、VMware vCenter でのマルチテナント機能をサポートしていません。 VIM でのリソースの作成に失敗した場合、ポータルはリソースの自動ロールバックを実行します。（競合する依存関係が原因で）場合によっては、ロールバック障害通知が表示された後で、テナントを手動で削除する必要があります。
VNF イメージを表示する	[リソース（Resources）] > [イメージ（Images）] の順に選択します。	選択したリソースのイメージのリストが表示されます。
VNF 展開フレーバーを表示する（OpenStack のみ）	[リソース（Resources）]>[フレーバー（Flavors）] の順に選択します。	選択したリソースのフレーバーのリストが表示されます。

タスク	移動方法	説明
ネットワークを表示する	[リソース（Resources）]>[ネットワーク（Networks）] の順に選択します。	サブネットワークとインターフェイスのネットワークごとに、ネットワークの詳細、テナント名、ネットワーク ID、ネットワークタイプなどが表示されます。それぞれの名前、ネットワーク ID、テナント ID などの詳細を確認できます。
サブネットワークを表示する（OpenStack のみ）	[リソース（Resources）] > [サブネットワーク（Subnetworks）] の順に選択します。	サブネットワークごとにサブネットワークの詳細、ネットワーク ID、サブネット ID などが表示されます。 （注） サブネットワークとインターフェイスのタブは、OpenStack でのみ使用できます。 ESC VM の初回起動時に、ネットワークおよびサブネットワークの作成フォームに空のテナントコンボボックスが表示されることがあります。テナントを正しくロードするには、ページを更新します。
インターフェイスを表示する（OpenStack のみ）	[リソース（Resources）] > [インターフェイス（Interfaces）] の順に選択します。	インターフェイスごとにインターフェイスの詳細、ネットワーク ID、サブネット ID、VM 名などが表示されます。
スイッチの詳細を表示する（VMware vCenter のみ）	[リソース（Resources）] > [スイッチ（Switches）] の順に選択します。	スイッチ、スイッチの名前、説明、UUID、およびホストのリストが表示されます。

タスク	移動方法	説明
展開テンプレートを使用してVNFを展開する	[システム（System）] > [展開テンプレート（Deployment Template）] の順に選択します。	事前設定済み展開テンプレートを作成します。
ESC への着信要求を表示する	[システム（System）] > [着信要求（Incoming Requests）] の順に選択します。	トランザクションIDや要求の詳細など、ESC へのすべての着信要求が一覧表示されます。
設定を表示する	[システム（System）] > [設定（Configuration）]の順に選択します。	VMの設定、モニタリングルール、VM 展開中のポリシーの適用などに使用されるすべての設定パラメータが一覧表示されます。
起動パラメータを表示する（OpenStack のみ）	[システム（System）]>[起動パラメータ（Boot Parameters）] の順に選択します。	ESC の起動に使用されるすべての起動パラメータが一覧表示されます。
ホストの詳細を表示する（OpenStack のみ）	[システム（System）] > [ホストの詳細（Host Details）] の順に選択します。	オペレーティングシステム（OS）、OSのバージョン、システム稼働時間、RAM、ストレージなどのホストの詳細が一覧表示されます。
ESC の正常性を表示する（OpenStack のみ）	[システム（System）] > [正常性（Health）] の順に選択します。	ESCの正常性、CONFDステータス、動作ステータス、およびその他の詳細が表示されます。
ログをダウンロードする	[システム（System）] > [ログ（Logs）] の順に選択します。	ログメッセージをダウンロードできます。
インフラストラクチャの詳細を表示する（OpenStack のみ）	[インフラストラクチャ（Infrasctructure）] > [インスタンス（Instances）] の順に選択します。	仮想化インフラストラクチャで実行されているすべてのVM。
ハイパーバイザを表示する（OpenStackのみ）	[インフラストラクチャ（Infrasctructure）]>[ハイパーバイザ（Hypervisors）] の順に選択します。	仮想化インフラストラクチャで実行されているすべてのハイパーバイザ。

タスク	移動方法	説明
VNF を展開解除する	• [展開（Deployments）] を選択します。 • テーブルから展開を選択し、テーブルのツールバーの [X] をクリックして展開を解除します。	VNF を展開解除します。
VDC を表示する（VMware vCenter のみ）	[リソース（Resources）] > [データセンター（Datacenters）] の順に選択します。	すべての仮想データセンターのリストが表示されます。

（注） ESC ポータルのページを小さな画面で表示すると、テーブルのフォーマットに問題が生じることがあります。テーブルを正しく表示するには、ブラウザ画面が 15 インチ以上である必要があります。

[システムパネル（System Panel ）] は、次のタブで構成されています。

- [パフォーマンス（Performance）] : パフォーマンスデータが表形式およびグラフ表示で表示されます。
- [ストレージ（Storage）] : ディスクの使用状況が表示されます。
- [vCPU使用状況（vCPU Utilization）] : ESC VM の vCPU の使用状況が表示されます。
- [正常性（Health）] : ネットワーク、データベース、tomcat など、さまざまな ESC プロセスの正常性が表示されます。
- [ホストの詳細（Host Details）] : オペレーティングシステム（OS）、OS のバージョン、システム稼働時間、RAM、ストレージの詳細などのホストの詳細が表示されます。

通知

[通知（Notification）] ページには、ESC 展開に関するすべての通知が一覧表示されます。

- エラーイベント：

  [通知（Notification）] ページからエラーイベントを選択し、[詳細情報の表示（View More Info）] をクリックして、エラーイベントの完全なレポートを表示します。

（注） 明示的なエラーメッセージを含むエラーイベントには、詳細レポートはありません。

完全なレポートは、REST API を使用して生成することもできます。troubleshooting-Id は、レポートを生成するために ESC-Status-Message に含まれています。

- 通知のクリア：

  通知を日付でソートして、削除できます。すべての通知を削除するには、[通知のクリア（Clear Notifications）] をクリックします。

第 42 章

ESC ポータルを使用したリソースの管理

ESC ポータルを使用した VIM コネクタの管理

ESC は、ESC ポータルを使用した VIM コネクタおよび VIM ユーザの追加と更新をサポートしています。複数の VIM を追加または更新して、マルチ VIM 展開を管理できます。マルチ VIM 展開の詳細については、「複数の OpenStack VIM での VNF の展開」を参照してください。

VIM コネクタテーブルには、VIM ID、VIM のタイプ、VIM のステータス、プロパティ、VIM ユーザなどの詳細が表示されます。

VIM コネクタの追加と削除

VIM コネクタを追加または削除するには、次の手順を実行します。

手順

ステップ 1 [リソース（Resources）] > [VIM（VIMs）] の順に選択します。

ステップ 2 [XMLのアップロード（Upload XML）] をクリックして、ファイルを選択します。[VIMの確認（Confirm VIMs）] ダイアログボックスが表示されます。

ステップ 3 [確認（CONFIRM）] をクリックして、XML ファイルをアップロードします。

ステップ 4 VIM のリストから VIM を削除するには、VIM を選択して [X] をクリックします。ダイアログボックスが表示されます。

ステップ 5 [OK] をクリックして VIM を削除します。

デフォルトの VIM コネクタ、およびリソースの依存関係がある VIM コネクタは削除できません。

VIM ユーザの管理

VIM ユーザの詳細は、[詳細を表示（View Details）] タブで確認できます。ESC ポータルでは、VIM ユーザを作成、更新、削除できます。

手順

ステップ 1 [リソース（Resources）] > [VIM] テーブルから VIM コネクタを選択し、[詳細を表示（View Details）] をクリックします。

[プロパティ（Properties）] および [VIMユーザ（VIM user）] ページが表示されます。

ステップ 2 [OK] をクリックして確定します。

VIM ユーザを更新するには、ユーザを選択し、[XMLのアップロード（Upload XML）]をクリックして更新された XML をアップロードします。

VIM ユーザを削除するには、テーブルで VIM ユーザを選択し、[X] をクリックします。VIM ユーザが削除されます。

VIM コネクタおよび VIM ユーザの詳細については、VIM コネクタの設定 （52 ページ）を参照してください。

ESC ポータルを使用した OpenStack リソースの管理

次のセクションでは、ESC ポータルを使用して OpenStack リソースを管理する方法について説明します。

- テナントの追加と削除
- イメージの追加と削除
- フレーバの追加と削除
- ネットワークの追加と削除
- サブネットワークの追加と削除

ESC ポータルでのテナントの追加と削除

ESC ポータルでテナントを追加および削除するには、次の手順を実行します。

手順

ステップ 1 [リソース（Resources）] > [テナント（Tenants）] の順に選択します。

ステップ 2 [+] をクリックして、テナントを追加します。[テナントの追加（Add Tenant）] ダイアログボックスが表示されます。

ステップ 3 名前と説明を追加し、[作成（Create）] をクリックします。

ステップ 4 テナントを削除するには、テナントのリストからテナントを選択し、[X] をクリックします。

ステップ 5 削除するには [OK] をクリックします。

ESC ポータル（OpenStack）でのイメージの追加と削除

ESC ポータルでイメージを追加および削除するには、次の手順を実行します。

手順

ステップ 1 [リソース（Resources）] > [イメージ（Images）] の順に選択します。

ステップ 2 イメージファイルをイメージテーブルにドラッグアンドドロップします。[イメージの確認（Confirm Image）] ダイアログボックスが表示されます。

ステップ 3 [確認（CONFIRM）] をクリックして、ドラッグしたテンプレートからイメージを作成します。

ステップ 4 イメージのリストからイメージを削除するには、そのイメージを選択して [X] をクリックします。ダイアログボックスが表示されます。

ステップ 5 [OK] をクリックして、イメージを削除します。

ESC ポータルでのフレーバーの追加と削除

ESC ポータルでフレーバーを追加および削除するには、次の手順を実行します。

手順

ステップ 1 [リソース（Resources）] > [フレーバー（Flavors）] の順に選択します。

ステップ 2 [フレーバー（Flavor）] テーブルにファイルをドラッグアンドドロップします。[フレーバーの確認（Confirm Flavor）] ダイアログボックスが表示されます。

ステップ 3 [確認（CONFIRM）] をクリックして、ドラッグしたテンプレートからフレーバーを作成します。

ステップ 4 フレーバーのリストからフレーバーを削除するには、そのフレーバーを選択して [X] をクリックします。ダイアログボックスが表示されます。

ステップ 5 [OK] をクリックして、フレーバーを削除します。

ESC ポータルでのネットワークの追加と削除

ESC ポータルからネットワークを追加および削除するには、次の手順を実行します。

手順

ステップ 1 [リソース（Resources）] > [ネットワーク（Networks）] の順に選択します。

ステップ 2 [ネットワーク（Networks）] テーブルにファイルをドラッグアンドドロップします。[ネットワークの確認（Confirm Network）] ダイアログボックスが表示されます。

ステップ 3 ネットワークのリストからネットワークを削除するには、ネットワークを選択して [X] をクリックします。ダイアログボックスが表示されます。

ステップ 4 [OK] をクリックして、ネットワークを削除します。

ESC ポータルでのサブネットワークの追加と削除

ESC ポータルでサブネットワークを追加および削除するには、次の手順を実行します。

手順

ステップ 1 [リソース（Resources）] > [サブネットワーク（Subnetworks）] の順に選択します。

ステップ 2 [サブネットワーク（Subnetworks）] テーブルにファイルをドラッグアンドドロップします。

（注） ドラッグアンドドロップ機能は、現時点では REST コールを実行し、NETCONF コールは実行しません。

ステップ 3 サブネットのリストからサブネットを削除するには、サブネットを選択して [X] をクリックします。ダイアログボックスが表示されます。

ステップ 4 [OK] をクリックして、サブネットワークを削除します。

ESC ポータルを使用した VMware vCenter リソースの管理

次の各項では、ESC ポータルを使用して VMware vCenter リソースを管理する方法について説明します。

- イメージの追加と削除
- ネットワークの追加と削除

ESC ポータルでのイメージの追加と削除（VMware）

ESC ポータルでは、フォームの適切なフィールドに入力してイメージを作成できます。

フォームからのイメージの作成

フォームからイメージを作成するには、次の手順を実行します。

手順

ステップ 1 [リソース（Resources）] > [イメージ（Images）] の順に選択します。

ステップ 2 [+] をクリックして、VNF イメージを追加します。[データセンターへのイメージの追加（Add Image to Datacenter）] ウィンドウが表示されます。

ステップ 3 [仮想データセンター（Virtual Datacenter）] ドロップダウンリストから、イメージを作成するデータセンターを選択します。

ステップ 4 [イメージ名（Image Name）] フィールドに、イメージ名を入力します。

ステップ 5 [イメージパス（Image Path）] フィールドに、イメージパスを入力します。

ステップ 6 [作成（Create）] をクリックして、イメージを作成します。

ステップ 7 イメージを削除するには、リストからそのイメージを選択して [X] をクリックします。ダイアログボックスが表示されます。

ステップ 8 [OK] をクリックして、イメージを削除します。

ESC ポータルでのネットワークの追加と削除（VMware）

ESC ポータルからネットワークを追加および削除するには、次の手順を実行します。

手順

ステップ 1 [リソース（Resources）] > [ネットワーク（Networks）]を選択して、フォームからネットワークを作成します。

ステップ 2 [+] をクリックしてネットワークを追加します。[データセンターへのネットワークの追加（Add Network to Datacenter）] ウィンドウが表示されます。

ステップ 3 [仮想データセンター（Virtual Datacenter）]ドロップダウンリストから、ネットワークを追加するデータセンターを選択します。

ステップ 4 [スイッチ（Switch）] ドロップダウンリストで、スイッチを選択します。

ステップ 5 [ネットワーク名（Network Name）] フィールドに、ネットワーク名を入力します。

ステップ 6 [VLAN] フィールドに、VLAN の番号を入力します。

ステップ 7 [ポート番号（Number of Ports）] フィールドにポート番号を入力します。

ステップ 8 [作成（Create）] をクリックします。

ステップ 9 ネットワークを削除するには、リストからネットワークを選択し、[X] をクリックします。ダイアログボックスが表示されます。

ステップ 10 [OK] をクリックして、ネットワークを削除します。

第 43 章

ESC ポータルを使用した VNF の展開

ESC ポータルを使用した仮想ネットワーク機能の展開（OpenStack のみ）

ESC ポータルを使用して、データモデル XML ファイルを展開することで、単一の VNF または複数の VNF をまとめて展開できます。ESC ポータルを使用して、次のいずれかの方法で単一の VNF または複数の VNF をまとめて展開できます。

手順

ファイルを使用した展開：既存のデータモデルファイルをアップロードできます。

次の項では、ESC ポータルを使用して VNF を展開する方法について説明します。

ファイルを使用した展開（展開データモデル）

既存の展開データモデルを使用して VNF を展開します。展開データモデルは、VNF の数およびその他の仕様が事前設定されています。展開データモデルを検索してアップロードするか、既存の展開データモデルをドラッグアンドドロップできます。ドラッグアンドドロップ機能を使用すると、既存の展開データモデルを取得し、ファイルをドラッグして展開テーブルにドロップすることで再利用できます。

（注） XML ファイルのみ受け入れられます。

手順

ステップ 1 [展開（Deployments）] を選択します。

ステップ 2 ファイルを [展開（Deployments）] テーブルにドラッグアンドドロップするか、テーブルツールバーの [XMLのアップロード（Upload XML）] をクリックして、ファイルを参照して選択します。

（注） ドラッグアンドドロップ機能は、現時点では REST コールを実行し、NETCONF コールは実行しません。

ESC ポータルを使用した VMware vCenter での VNF の展開

ESC ポータルでは、単一の VNF または複数の VNF を一緒に展開できます。既存の展開データモデルがポータルを介してアップロードされるか、新しい展開データモデルが作成されます。新しい展開データモデルは、ESC ポータルの該当するすべてのフィールドに入力することによって作成されます。ESC では、ポータルから展開データモデルをエクスポートすることもできます。次のセクションでは、ESC ポータルを使用して VNF を展開する複数の方法について説明します。

次の項では、ESC ポータルを使用して VNF を展開する方法について説明します。

手順

ステップ 1 ファイルを使用して展開します。

ステップ 2 フォームを使用して展開します。

ファイルを使用した展開（展開データモデル）

既存の展開データモデルを使用して VNF を展開します。展開データモデルは、VNF の数およびその他の仕様が事前設定されています。展開データモデルを検索してアップロードするか、既存の展開データモデルをドラッグアンドドロップできます。ドラッグアンドドロップ機能を使用すると、既存の展開データモデルを取得し、ファイルをドラッグして展開テーブルにドロップすることで再利用できます。

（注） XML ファイルのみ受け入れられます。

手順

ステップ 1 [展開（Deployments）] を選択します。

ステップ 2 ファイルを [展開（Deployments）] テーブルにドラッグアンドドロップするか、テーブルツールバーの [XMLのアップロード（Upload XML）] をクリックして、ファイルを参照して選択します。

（注） ドラッグアンドドロップ機能は、現時点では REST コールを実行し、NETCONF コールは実行しません。

フォームを使用した展開

新しい展開テンプレートを作成するには、次の手順を実行します。

（注） [テンプレートのエクスポート（Export Template）] をクリックして、展開データモデルをエクスポートします。

手順

ステップ 1 [展開（Deployments）] を選択します。

ステップ 2 [+]をクリックして、フォームを使用して展開します。

ステップ 3 展開名を入力します。

ステップ 4 [データセンター（Datacenter）] ドロップダウンリストから、VNF を展開するデータセンターを選択します。

仮想データセンターの詳細については、「VMware vCenter での仮想ネットワーク機能の展開」を参照してください。

ステップ 5 [全般（General）] タブで、フィールドに適切な値を入力します。

a) [配置（Placement）] フィールドで、[クラスタ（Cluster）] または [ホスト（Host）] オプションボタンを選択します。

- [クラスタ（Cluster）]：同じクラスタで VNF を展開するクラスタの名前を選択します。
- [ホスト（Host）]：同じホストで VNF を展開するホストを選択します。
- [データストア（Datastore）]：選択したクラスタのデータストアを選択します。
- [イメージ（Image）]：イメージを選択します。

ステップ 6 スマートライセンスを有効にするには、[スマートライセンスの有効化（Enable Smart Licensing）] をクリックします。

ステップ 7 [グループ内ルールの有効化（Enable Intragroup Rules）] をクリックして、グループ内ルールを有効にします。

a) [タイプ（Type）] ドロップダウンリストから、[アフィニティ（Affinity）] または [アンチアフィニティ（Anti-Affinity）] を選択して、アフィニティルールまたはアンチアフィニティルールを有効にします。

グループ内アフィニティルールの詳細については、アフィニティルールとアンチアフィニティルール （205 ページ）を参照してください。

ステップ 8 （オプション）[VNFグループ内ルールの追加（Add VNF Intergroup Rule）] タブをクリックして、アフィニティルールまたはアンチアフィニティルールを適用する VNF を選択します。

グループ内アフィニティルールの詳細については、アフィニティルールとアンチアフィニティルール （205 ページ）を参照してください。

ステップ 9 障害発生時に ESC が VNF を修復するために使用するパラメータを指定するには、[リカバリ（Recovery）] タブをクリックします。

リカバリまたは修復の詳細については、仮想ネットワーク機能の修復 （341 ページ）を参照してください。

ステップ 10 インターフェイスの数と各インターフェイスのプロパティを指定するには、[インターフェイス（Interfaces）] タブをクリックします。ここで指定されたインターフェイスの順序は、VM のインターフェイスの順序とは一致しません。

a) [インターフェイスの追加（Add Interfaces）] をクリックしてインターフェイスを追加します。

ステップ 11 インスタンス化する必要がある特定のタイプの VM のインスタンス数を指定し、柔軟にスケールインおよびスケールアウトするには、[スケーリング（Scaling）] タブをクリックします。

a) [静的IPプールの追加（Add Static IP Pool）] をクリックして、静的 IP プールを追加します。

ステップ 12 ESC 内のモニタモジュールの設定に使用するモニタリングルールを指定するには、[モニタリング（Monitoring）] タブをクリックします。

モニタリングの詳細については、仮想ネットワーク機能のモニタリング （309ページ）を参照してください

ステップ 13 [設定データ（Config Data）] タブで、フィールドに適切な値を入力します。

ステップ 14 （オプション）[OVF設定（OVF Settings）] タブで、フィールドに適切な値を入力します。

a) [OVFプロパティの追加（Add OVF Property）] をクリックして、OVF プロパティのリストを追加します。

展開テンプレートを使用した仮想ネットワーク機能の展開

ESC ポータルから事前設定済み展開テンプレートをアップロードすることで、VNF を展開できるようになりました。

1. [システム（System）] > [展開テンプレート（Deployment Templates）] に移動します。
2. [XMLのアップロード（Upload XML）] をクリックします。

   ドラッグアンドドロップするか、事前設定済み展開テンプレート（dep.xml）を選択して [確認（Confirm）] をクリックします。展開テンプレートがテーブルに表示されます。
3. アップロードした展開テンプレートを選択し、[テンプレートから展開（Deploy from Template）] をクリックします。
4. 展開名とテナント名は、アップロードしたテンプレートから追加されます。必要に応じてフィールドを変更するか、[作成（Create）] をクリックしてテンプレートを作成します。
5. 正常終了のメッセージが画面に表示されます。[OK] をクリックします。

新しい展開テンプレートが [展開（Deployments）] ビューに表示されます。

事前設定済みテンプレート

既存の dep.xml に変更を加えて、事前設定済みテンプレートとして使用できます。ユーザは、データモデルに次の変更を加える必要があります。

- esc_datamodel の代わりに esc_datamodel_template タグを使用します。
- esc_datamodel_template name プロパティは一意であり、テンプレートを識別するために指定する必要があります。
- param_key は、カスタマイズ可能な値を識別するためにポータルで使用されます。必須フィールドです。このキーは一意ですが、テンプレートに複数回表示されることがあります。
- prompt に、ユーザが追加する必要がある入力値が表示されます。必須フィールドです。ドキュメント内の別の場所で指定された同じ param_key に対する prompt が異なる場合は、最初の prompt が使用されます。
- core はデフォルト値で、空白のままにできます。
- required は、ユーザがこの値を入力する必要があるかどうかを指定します。これはオプションのフィールドです。デフォルト値は true です。
- range は、数値フィールドを検証します。これはオプションのフィールドです。

事前設定済みテンプレートの例：

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel_template xmlns="http://www.cisco.com/esc/esc" name="VPC Template 1">
   <tenants>
      <tenant>
         <name param_key="tenant_name" prompt="Tenant Name">core</name>
         <managed_resource>false</managed_resource>
         <deployments>
            <deployment>
               <name param_key="dep_name" prompt="Deployment
Name">vnfd3-deployment-1.0.0-1</name>
               <policies>
                  <placement>
                     <target_vm_group_ref>c2</target_vm_group_ref>
                     <type>anti_affinity</type>
                     <enforcement>strict</enforcement>
                     <vm_group_ref>c1</vm_group_ref>
                  </placement>
               </policies>
            </deployment>
         </deployments>
      </tenant>
   </tenants>
</esc_datamodel_template>
```

第 44 章

ESC ポータルを使用した VNF および VM の操作

ポータルを使用して、起動、停止、再起動などの VNF 操作を実行できます。VNF 操作は、展開の状態に応じて、展開された VNF で実行できます。

VNF 操作の実行

VNF 操作を実行するには、次の手順を実行します。

手順

ステップ 1 [展開（Deployments）] を選択します。

ステップ 2 展開ページで VNF を選択します。

（注） 操作は、展開の状態に応じて有効になります。

ステップ 3 テーブルのツールバーから必要な操作をクリックします。実行できる操作のリストについては、次の表を参照してください。

操作を実行するには、VNF が次の展開状態になっている必要があります。

VNF 操作	展開状態
モニタの有効化	非アクティブまたはエラー
モニタの無効化	アクティブ
VNF の開始	停止
VNF の停止	アクティブまたは非アクティブ

VNF 操作	展開状態
VNF の再起動	アクティブまたは非アクティブ
VNF の回復	エラー
VNF のモニタと回復（手動リカバリ）	エラー

VM 操作の実行

VM 操作を実行するには、次の手順を実行します。

手順

ステップ 1 [展開（Deployments）] を選択します。

ステップ 2 展開ページで VNF を選択します。

（注） 操作は、展開の状態に応じて有効になります。

ステップ 3 [VMグループの表示（View VM Groups）] をクリックします。

ステップ 4 [VMグループインスタンス（VM Group Instances）] で、操作を選択します。実行できる操作のリストについては、次の表を参照してください。

ステップ 5 [確認（Confirm）] をクリックします。

操作を実行するには、VM が次の展開状態である必要があります。

VM の操作	展開状態
モニタの有効化	非アクティブまたはエラー
モニタの無効化	アクティブ
VM の起動	シャットオフ
VM の停止	アクティブまたは非アクティブ
VM のリブート	アクティブまたは非アクティブ
VM の回復	エラー

第 45 章

ポータルを使用したVNFおよびVMのリカバリ

ポータルを使用した VNF および VM のリカバリ

ESC ポータルを使用して VNF と VM の手動回復を実行できるようになりました。

手順

ステップ 1 [展開（Deployments）] を選択します。

ステップ 2 エラー状態の展開を選択します。

VM レベルのリカバリでは、[VNFの表示（View VNFs）] ページからエラー状態の VM を選択します。

ステップ 3 [VNFの回復（Recover VNF）] または [VNFのモニタと回復（Monitor＋Recover VNF）] をクリックします。

ステップ 4 [OK] をクリックして確定します。

ステップ 5 [リカバリアクション（Recovery Action）] ドロップダウンからリカバリアクションを選択し、[OK] をクリックします。

次のリカバリアクションを使用できます。

- [デフォルト（Default）]：データモデルで定義されたリカバリアクションをトリガーします。
- REBOOT_ONLY
- REDEPLOY_ONLY
- REBOOT_THEN_REDEPLOY

リカバリオプションの詳細については、リカバリポリシー （343 ページ）を参照してください。

重要なポイント

1. 設定可能な手動回復は、実行中のトランザクション動作をサポートしていません。したがって、設定可能な手動回復中にフェールオーバーが発生すると、手動回復は事前定義されたリカバリアクションで再開されます。
2. 展開の移行では、デフォルトのリカバリポリシーを使用します。LCS ベースのリカバリでは、VM/VNF 手動回復のリカバリアクションは提供されません。

第 46 章

ESC システムレベルの設定

ESC ポータルからのログのダウンロード

ESC ポータルからすべてのログファイルをダウンロードできるようになりました。ログには次の 2 種類があります。

- トレースログ：vimmanager ログ、esc_rest ログ、および esc_netconf ログが含まれます。
- システムログ：escmanager ログ、vimmanager ログ、およびトレースログを除く他のすべての ESC 関連ログが含まれます。

手順

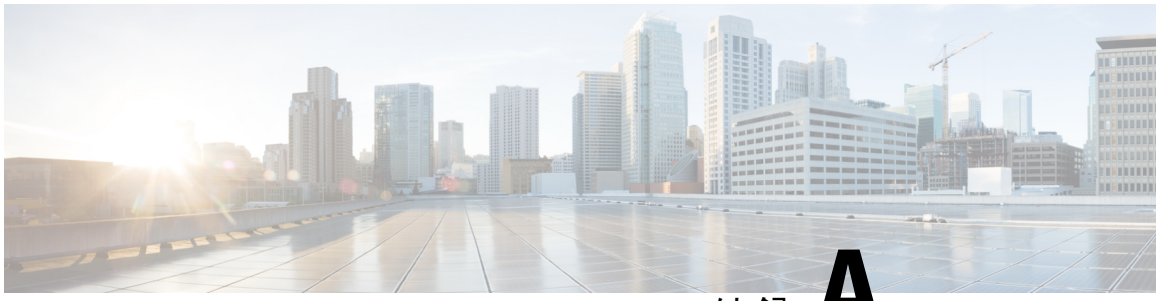
ステップ 1 [システム（System）] > [ログ（Logs）] の順に選択します。

ステップ 2 トレースログの [メッセージのトレースログを要求（Request message trace logs）] をクリックするか、すべての ESC 関連ログの [システムログを要求（Request system logs）] をクリックします。

ダウンロード可能なファイルが（作成後に）テーブルに表示されます。

ログが大きい場合は、コンパイルに時間がかかることがあります。ファイルをダウンロードする前に、しばらく待つ必要があります。

ステップ 3 ダウンロード可能なファイルをクリックして、マシンに保存します。



付録 A

Cisco Cloud Services Platform（CSP）拡張機能

クラウド サービス プロバイダーの拡張機能

次の表に、VIM として CSP をサポートするために ESC に追加されたすべての拡張機能を示します。VIM コネクタの詳細については、「VIM コネクタの設定」を参照してください。

表 *30 : CSP* の拡張機能

リソース	内線番号 導入例
展開/VM グループ	拡張機能：なし。 展開例：「VIM コネクタの管理」を参照

フレーバ	拡張機能：なし。
	展開例：
	
	`<flavor>`
	`  <name>FLAVOR_2_4096_10000</name>`
	`  <vcpus>2</vcpus>`
	`  <memory_mb>4096</memory_mb>`
	`  <root_disk_mb>10000</root_disk_mb>`
	`</flavor>`

Cisco Cloud Services Platform（CSP）拡張機能

リソース	内線番号 導入例
ストレージディスク 展開/ボリューム	拡張機能： <extension> <name>volumes</name> <containers> <container> <name>1</name> <properties> <property> <name>storage_disk_format</name> <value>raw | qcow2</value> </property> <property> <name>storage_disk_device</name> <value>disk | cdrom</value> </property> <property> <name>storage_disk_location</name> <value>local | NFS mount </value> </property> </properties> </container> </containers> </extension> 展開例： <volumes> <volume> <volid>1</volid> <sizeunit>GiB</sizeunit> <size>20</size> <bus>virtio</bus> </volume> </volumes>
Deployment/ vm group / extentions/ image	拡張機能： <extension> <name>image</name> <properties> <property> <name>disk-resize</name> <value>true</value> </property> <property> <name>disk_type</name> <value>virtio</value> </property> <property> <name>disk_storage_name</name> <value>esc_nas_old</value> </property> <property> <name>image_storage_name</name> <value>esc_nas_old</value> </property> </properties> </extension>

リソース	内線番号 導入例
Deployment/ vm group / extentions/ vnc	拡張機能： <extension> <name>vnc</name> <properties> <property> <name>vnc_password</name> <value>**********</value> </property> </properties> </extension>
Deployment/ vm group / extentions/ vnf_mgmt_ip	拡張機能： <extension> <name>vnf_mgmt_ip</name> <properties> <property> <name>nicid</name> <value>0</value> </property> </properties> </extension>
Deployment/ vm group / serial_ports	拡張機能： <extension> <name>serial_ports</name> <containers> <container> <name>0</name> <properties> <property> <name>serial_type</name> <value>console</value> </property> </properties> </container> </containers> </extension>

Cisco Cloud Services Platform（CSP）拡張機能

リソース	内線番号 導入例
Deployment/ vm group / interfaces /	拡張機能： <extensions> <extension> <name>interfaces</name> <containers> <container> <name>1</name> <properties> <property> <name>passthroughMode</name> <value>none</value> </property> <property> <name>tagged</name> <value>false</value> </property> <property> <name>type</name> <value>access</value> </property> <property> <name>bandwidth</name> <value>7500</value> </property> <property> <name>vlan</name> <value>1</value> </property> </properties> </container> </containers> </extension> </extensions> 展開例： <interface> <nicid>1</nicid> <type>virtual</type> <model>virtio</model> <network>Eth0-2</network> <admin_state_up>false</admin_state_up> </interface>
Deployment/ vm group / <vim_vm_name>	拡張機能：なし。 展開例：<vim_vm_name>my-custom-csr</vim_vm_name>

リソース	内線番号 導入例
Deployment/ vm group /day0-volume-id	拡張機能：なし。 展開例： <config_type>CONFIG_DATA_OPTIONS</config_type> <config_options> <options> <option> <name>day0-volume-id</name> <value>cidata</value> </option> </options> </config_options>

Cisco Cloud Services Platform（CSP）拡張機能

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。