



Cisco Elastic Services Controller 5.2 ユーザガイド

初版：2020年5月29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/>

【注意】 シスコ製品をご使用になる前に、安全上の注意（www.cisco.com/jp/go/safety_warning/）をご確認ください。本書は、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。また、契約等の記述については、弊社販売パートナー、または、弊社担当者にご確認ください。

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/c/en/us/about/legal/trademarks.html>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2020 Cisco Systems, Inc. All rights reserved.



目次

Full Cisco Trademarks with Software License ?

はじめに :

[このマニュアルについて](#) **xiii**

[対象読者](#) **xiii**

[用語および定義](#) **xiii**

[関連資料](#) **xv**

第 I 部 :

[はじめに](#) **17**

第 1 章

[Elastic Services Controller の概要](#) **1**

[Elastic Services Controller の主な機能](#) **1**

[ESC アーキテクチャ](#) **2**

[ESC ライフサイクルについて](#) **3**

第 2 章

[Elastic Services Controller インターフェイス](#) **9**

[Elastic Services Controller インターフェイス](#) **9**

[Elastic Services Controller NB API](#) **9**

[NETCONF/YANG ノースバウンド API](#) **10**

[複数リソースを設定するための NETCONF 要求](#) **13**

[REST ノースバウンド API](#) **14**

[ETSI NFV MANO Northbound API](#) **15**

[Elastic Services Controller ポータル](#) **16**

第 II 部 :

[リソースの管理](#) **17**

リソース管理の概要 17

第 3 章**OpenStack のリソースの管理 21**

OpenStack のリソースの管理 21

テナントの管理 21

ネットワークの管理 30

サブネットの管理 33

フレーバの管理 34

イメージの管理 35

ボリュームの管理 36

第 4 章**VMware vCenter のリソースの管理 43**

VMware vCenter でのイメージの追加 43

VMware vCenter での分散ポートの作成 44

第 5 章**vCloud Director のリソースの管理 47**

vCloud Director (vCD) のリソースの管理 47

第 6 章**ESC リソースの管理 49**

VIM コネクタの管理 49

VIM コネクタの設定 50

デフォルトの VIM コネクタ 51

VIM コネクタの削除 52

VIM コネクタ API を使用した VIM コネクタの管理 52

VIM コネクタのステータス API 57

VIM コネクタ操作のステータス 59

第 7 章**VIM コネクタの設定 61**

OpenStack の VIM コネクタの設定 61

OpenStack エンドポイントの上書き 68

	AWS の VIM コネクタ設定	69
	VMware vCloud Director (vCD) の VIM コネクタの設定	70
第 8 章	異なる VIM の VIM コネクタのプロパティ	73
	VIM コネクタのプロパティ	73
第 9 章	外部設定ファイルの認証	77
	外部設定ファイルの認証	77
	設定データの暗号化	83
第 III 部 :	仮想ネットワーク機能のオンボーディング	85
第 10 章	仮想ネットワーク機能のオンボーディング	87
	OpenStack での仮想ネットワーク機能のオンボーディング	87
	OpenStack 展開のためのデータモデルの準備	88
	VMware vCenter での仮想ネットワーク機能のオンボーディング	90
	VMware vCenter 展開のためのデータモデルの準備	90
第 IV 部 :	仮想ネットワーク機能の展開と設定	95
第 11 章	仮想ネットワーク機能の展開	97
	仮想ネットワーク機能の展開	97
第 12 章	OpenStack での仮想ネットワーク機能の展開	99
	OpenStack での仮想ネットワーク機能の展開	99
	単一の OpenStack VIM での VNF の展開	100
	再起動時間パラメータ	101
	複数の OpenStack VIM への VNF の展開	103
第 13 章	複数の VIM への仮想ネットワーク機能の展開	109
	複数の VIM への仮想ネットワーク機能の展開	109

マルチ VIM 展開でサポートされる機能 110

第 14 章

ブラウнフィールド VM の展開 113

VNF のインポート 113

VM の展開 115

VM の作成 117

VM の展開解除 117

ブラウнフィールドモードでの VM の操作 117

MONA トリガーイベントのブロッキング 118

第 15 章

VMware vCenter での仮想ネットワーク機能の展開 121

VMware vCenter での仮想ネットワーク機能の展開 121

単一の VMware vCenter VIM での VNF の展開 122

VMware vCloud Director (vCD) での仮想ネットワーク機能の展開 126

第 16 章

Amazon Web Services での仮想ネットワーク機能の展開 131

Amazon Web Services での仮想ネットワーク機能の展開 131

単一または複数の AWS リージョンでの VNF の展開 132

第 17 章

統合型の展開 137

統合型の展開 137

第 18 章

仮想ネットワーク機能の展開解除 139

仮想ネットワーク機能の展開解除 139

第 19 章

展開パラメータの設定 141

導入パラメータ 141

第 20 章

デイゼロ設定 145

デイゼロ設定 145

データモデルの設定のデイゼロ 145

ファイルロケータ	148
vCD 展開のデイレゾ設定	150

第 21 章	KPI、ルール、およびメトリック	153
	KPI、ルール、およびメトリック	153
	ルール (Rules)	153
	メトリックおよびアクション	154
	メトリックおよびアクション API	155
	スクリプトアクション	160
	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	164
	カスタムスクリプト通知	167

第 22 章	ポリシー駆動型データモデル	171
	ポリシー駆動型データモデル	171

第 23 章	サポート対象のライフサイクルステージ (LCS)	173
	サポート対象のライフサイクルステージ (LCS)	173
	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	175

第 24 章	アフィニティおよびアンチアフィニティルール	177
	アフィニティルールとアンチアフィニティルール	177

第 25 章	OpenStack のアフィニティおよびアンチアフィニティルール	179
	OpenStack のアフィニティおよびアンチアフィニティルール	179
	グループ内アンチアフィニティポリシー	180
	グループ間アフィニティポリシー	180
	グループ間アンチアフィニティポリシー	181
	制限事項	183
	展開間アンチアフィニティポリシー	183

第 26 章	VMware vCenter のアフィニティールールとアンチアフィニティールール	185
	VMware vCenter のアフィニティールールとアンチアフィニティールール	185
	グループ内アフィニティポリシー	185
	グループ内アンチアフィニティ	186
	クラスタの配置	186
	ホストの配置	187
	グループ間アフィニティポリシー	187
	グループ間アンチアフィニティポリシー	187
	制限事項	189

第 27 章	VMware vCloud Director のアフィニティールールとアンチアフィニティールール	191
	VMware vCloud Director のアフィニティールールとアンチアフィニティールール	191

第 28 章	カスタム VM 名の設定	193
	カスタム VM 名の設定	193

第 29 章	既存の展開の管理	197
	既存の展開の更新	197

第 30 章	展開状態とイベント	229
	展開またはサービスの状態	229
	イベント通知またはコールバックイベント	231

第 31 章	LCS を使用した VNF ソフトウェアのアップグレード	237
	VNF ソフトウェアのアップグレード	237
	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	238
	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	238
	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)	240
	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	243

	展開内の VNF のアップグレード	247
第 32 章	仮想ネットワーク機能の操作	249
	VNF 操作	249
	個々の VNF と複合 VNF の管理	250
第 V 部 :	モニタリング、スケーリング、および修復	251
第 33 章	仮想ネットワーク機能のモニタリング	253
	VNF のモニタリング	253
	モニタリング方式	260
	VM のモニタリング	261
	VM モニタリングステータスの通知	263
	モニタリング操作	264
第 34 章	D-MONA を使用した VNF のモニタリング	265
	D-MONA のオンボーディング	265
	D-MONA の展開	265
	D-MONA の設定	266
	モニタリングのための D-MONA を使用した VNF の展開	267
	D-MONA の回復	269
	D-MONA ログの取得	269
第 35 章	ローカル MONA と D-MONA 間での VNF モニタリングの切り替え	271
	前提条件	271
	ローカル MONA VNF を D-MONA に移行する手順	271
	D-MONA をローカル MONA VNF に移行する手順	272
第 36 章	仮想ネットワーク機能のスケーリング	273
	スケーリングの概要	273
	VM のスケールインとスケールアウト	273

スケーリングのためのリソースの一貫した順序付け 275

スケーリング通知とイベント 276

第 37 章

仮想ネットワーク機能の修復 279

修復の概要 279

VM の修復 279

リカバリポリシー 281

リカバリポリシーと再展開ポリシー 286

リカバリポリシー（ポリシーフレームワークを使用） 286

再展開ポリシー 289

ホストの有効化と無効化 294

通知とイベント 296

第 VI 部 :

ESC ポータル 305

第 38 章

使用する前に 307

ESC ポータルへのログイン 307

ESC パスワードの変更 308

ESC ポータルパスワードの変更 309

ESC ポータルダッシュボード 309

第 39 章

ESC ポータルを使用したリソースの管理 315

ESC ポータルを使用した VIM コネクタの管理 315

VIM ユーザの管理 316

ESC ポータルを使用した OpenStack リソースの管理 316

ESC ポータルでのテナントの追加と削除 316

ESC ポータル（OpenStack）でのイメージの追加と削除 317

ESC ポータルでのフレーバーの追加と削除 317

ESC ポータルでのネットワークの追加と削除 318

ESC ポータルでのサブネットワークの追加と削除 318

ESC ポータルを使用した VMware vCenter リソースの管理 318

	ESC ポータルでのイメージの追加と削除 (VMware)	319
	ESC ポータルでのネットワークの追加と削除 (VMware)	319
<hr/>		
第 40 章	ESC ポータルを使用した VNF の展開	321
	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)	321
	ファイルを使用した展開 (展開データモデル)	321
	ESC ポータルを使用した VMware vCenter での VNF の展開	322
	ファイルを使用した展開 (展開データモデル)	322
	フォームを使用した展開	323
	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	325
<hr/>		
第 41 章	ESC ポータルを使用した VNF および VM の操作	327
	VNF 操作の実行	327
	VM 操作の実行	328
<hr/>		
第 42 章	ポータルを使用した VNF および VM のリカバリ	329
	ポータルを使用した VNF および VM のリカバリ	329
	重要なポイント	330
<hr/>		
第 43 章	ESC システムレベルの設定	331
	ESC ポータルからのログのダウンロード	331
<hr/>		
付録 A :	Cisco Cloud Services Platform (CSP) 拡張機能	333
	クラウド サービス プロバイダーの拡張機能	333



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、および Amazon Web Services (AWS) は、サポートされている VIMs です。管理者は、VIM レイヤ、vCenter、OpenStack および AWS のリソース、ならびに使用するコマンドに精通している必要があります。

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

用語および定義

次の表で、このガイドで使用されている用語を定義します。

表 1:用語および定義

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

用語	定義
ETSI	欧州電気通信標準化機構（ETSI）は、欧州内の情報通信技術（ICT）の標準開発において貢献してきた独立標準化機関です。
ETSI 展開 フレーバ	展開フレーバの定義には、VNF インスタンスに適用するアフィニティ関係、スケーリング、最小/最大 VDU インスタンス、その他のポリシーと制限に関する情報が含まれています。VNF 記述子（VNFD）で定義された展開のフレーバは、インスタンス化 VNF LCM 操作時に <code>InstantiateVNFRequest</code> ペイロードで <code>flavour_id</code> 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クサービスを迅速に導入できるようにするサービスの作成と配信のプラットフォームです。
NFV	ネットワーク機能仮想化（NFV）は、仮想ハードウェアの抽象化を使用して実行するネットワーク機能をハードウェアから分離する原則です。
NFVO	NFV オーケストレータ（NFVO）は、ネットワークサービス（NS）のライフサイクルを管理し、NS ライフサイクル、VNF ライフサイクル（VNFM でサポート）、NFVI リソース（VIM でサポート）の管理を調整して、必要なリソースと接続の割り当てを最適化します。
NSO	Cisco Network Services Orchestrator（NSO）は、サービス アクティベーションのためのオーケストレータであり、純粋な物理ネットワーク、ハイブリッドネットワーク（物理および仮想）、および NFV の使用をサポートします。
OpenStack コンピューティングの フレーバ	フレーバで、Nova コンピューティングインスタンスのコンピューティング、メモリ、およびストレージ容量を定義します。フレーバは、サーバに使用可能なハードウェア設定です。起動可能な仮想サーバのサイズを定義します。
サービス	サービスは、1 つまたは複数の VNF で構成されます。
VDU	仮想化展開ユニット（VDU）は、情報モデルで使用できる構成要素であり、VNF のサブセットの展開と運用動作の説明、またはサブセットにコンポーネントとして含まれていない場合は VNF 全体の説明をサポートします。

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

関連資料

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

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

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

ドキュメントの入手方法

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

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



第 1 部

はじめに

- [Elastic Services Controller の概要 \(1 ページ\)](#)
- [Elastic Services Controller インターフェイス \(9 ページ\)](#)



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

ESC は VNF マネージャとして、仮想マネージドサービスと、仮想パケットコア、仮想ロードバランサ、仮想セキュリティサービスなどのすべてのサービスプロバイダーの NFV 展開を対象とします。複雑なサービスには複数の VM が含まれており、それらの間に依存関係がある単一のサービスとして調整されています。

- [Elastic Services Controller の主な機能 \(1 ページ\)](#)
- [ESC アーキテクチャ \(2 ページ\)](#)
- [ESC ライフサイクルについて \(3 ページ\)](#)

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ー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、サービスアダプタイズメント、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 への VM の展開をサポートします (OpenStack のみ)。
- 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ー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

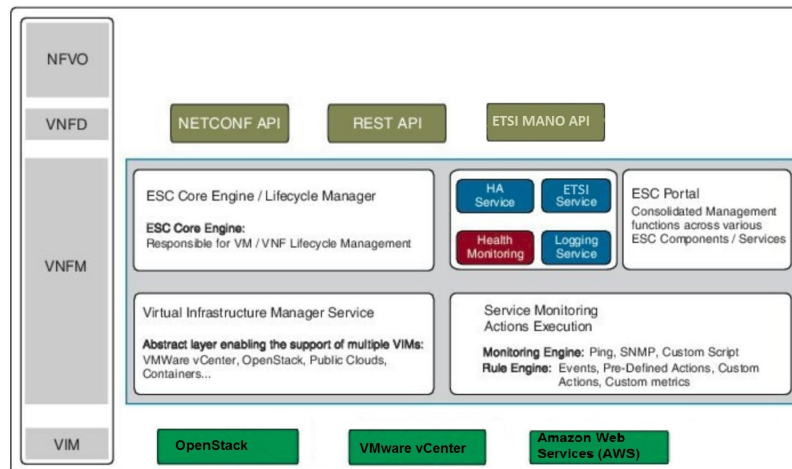
されます。つまり、VNF のオンボーディング、展開、モニタリング、および KPI 要件に基づく修復やスケーリングなどの VNF レベルのライフサイクルの決定が行われます。ESC とその管理対象 VNF は、仮想インフラストラクチャ マネージャ (VIM) 内で実行される VM として展開されます。現在サポートされている VIM は、OpenStack、VMware vCenter、および AWS です。ESC コアエンジンは、トランザクション、検証、ポリシー、ワークフロー、および VM ステートマシンを管理します。ESC のモニタリングおよびアクションサービスエンジンは、複数のモニタリング方式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。

このセクションを更新します。

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。

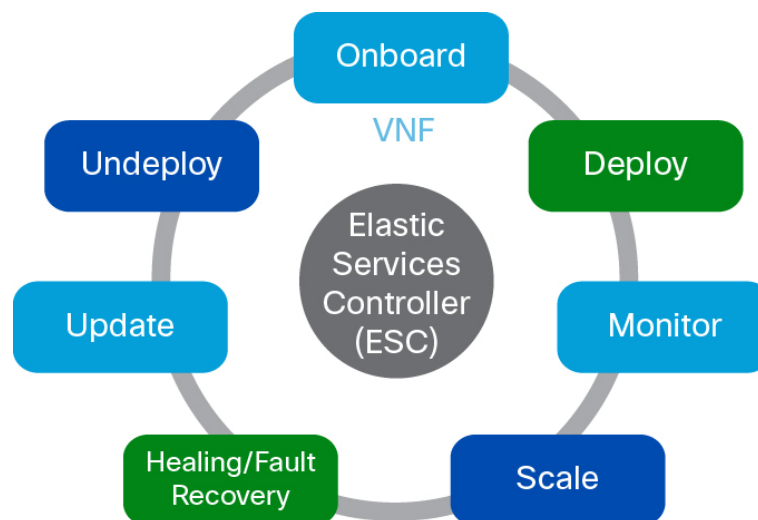


(注) VNF は、OpenStack または VMware vCenter に展開できます。ハイブリッド VNF 展開はサポートされていません。

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 ライフサイクル



407058

- 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フォルト設定を適用します。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west 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。	<ul style="list-style-type: none"> 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。	<ul style="list-style-type: none"> 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（99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せん。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単一 VIM への 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。	<ul style="list-style-type: none"> 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。	<ul style="list-style-type: none"> 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。	<ul style="list-style-type: none"> 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。	<ul style="list-style-type: none"> 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せん。

VMware vCenter での VNF の展開の詳細については、[VMware vCenter での仮想ネットワーク機能の展開 \(121 ページ\)](#) 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](#) (9 ページ)
- [Elastic Services Controller NB API](#) (9 ページ)
- [Elastic Services Controller ポータル](#) (16 ページ)

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。

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 command <file name> を入力するだけです。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 サブスクリバに送信されます。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>
  </rpc-error>
```

```

    <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のユーザー名が表示されます。

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ユーザー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ユーザーが表示されます。
- `member_role` : OpenStackユーザー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 \(17 ページ\)](#)
- [OpenStack のリソースの管理 \(21 ページ\)](#)
- [VMware vCenter のリソースの管理 \(43 ページ\)](#)
- [vCloud Director のリソースの管理 \(47 ページ\)](#)
- [ESC リソースの管理 \(49 ページ\)](#)
- [VIM コネクタの設定 \(61 ページ\)](#)
- [異なる VIM の VIM コネクタのプロパティ \(73 ページ\)](#)
- [外部設定ファイルの認証 \(77 ページ\)](#)

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。 <ul style="list-style-type: none">• 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。 <ul style="list-style-type: none">• 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。 <ul style="list-style-type: none">• NETCONF API• REST API• ESC ポータル

リソースの定義	OpenStack	VMware vCenter
サブネット (Subnets)	<p>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。</p> <ul style="list-style-type: none"> • NETCONF API • REST API • ESC ポータル 	適用なし
フレーバ	<p>OpenStack です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。</p> <ul style="list-style-type: none"> • NETCONF API • REST API • ESC ポータル 	適用なし
画像	<p>OpenStack です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。</p> <ul style="list-style-type: none"> • NETCONF API • REST API • ESC ポータル 	<p>VMware vCenter です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。</p> <ul style="list-style-type: none"> • NETCONF API • REST API • ESC ポータル
音量 (Volumes)	<p>OpenStack です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。</p>	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 (Version)
OpenStack	<ul style="list-style-type: none">• 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 \(21 ページ\)](#)
- [テナントの管理 \(21 ページ\)](#)
- [ネットワークの管理 \(30 ページ\)](#)
- [サブネットの管理 \(33 ページ\)](#)
- [フレーバの管理 \(34 ページ\)](#)
- [イメージの管理 \(35 ページ\)](#)
- [ボリュームの管理 \(36 ページ\)](#)

OpenStack のリソースの管理

テナントの管理

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



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

ESC では、次の 3 種類のテナントを作成できます。

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

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

テナント名は一意である必要があります。



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

データモデルのテナントは、次の属性で管理します。

- managed_resource 属性
- vim_mapping 属性

次の表に、データモデルのテナントと属性のマッピングの詳細が示されています。

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

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

同じタイプの複数の VIM (OpenStack VIM) に VM を展開するには、vim_mapping 属性を false に設定してテナントを作成する必要があります。このテナントは、個別に作成することも、展開の一部として作成することもできます。これにより、ESC 内にテナントが作成され、マルチ VIM 展開のルートテナントとして機能します。マルチ VIM 展開の場合は、各 VM グループ内で VIM ロケータ属性を指定する必要があります。詳細については、[単一の VMware vCenter VIM での VNF の展開 \(122 ページ\)](#) 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
- ルーター

次の展開データモデルは、テナントのクォータ設定を示しています。

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

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

```

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

```



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

ノースバウンド API を使用したテナントの追加

次に、NETCONF を使用してテナント定義を作成する例を示します。

```

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

```



- (注) NETCONF API を使用したテナント定義の作成と削除の詳細については、[Cisco Elastic Services Controller API ガイド \[英語\]](#) を参照してください。ESC VM から REST API ドキュメントに直接アクセスする場合は、[REST ノースバウンド API \(14 ページ\)](#) 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 \(315 ページ\)](#) 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ワークを作成して設定し、OpenStack または VMware vCenter のいずれかのサービスにそれらのネットワークのポートに仮想マシンを接続するように指示することで、豊富なネットワークトポロジを設定できます。

OpenStack ネットワーク

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

ESC は次のネットワーク機能をサポートしています。

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

次の属性で、プロバイダーネットワークを定義します。

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

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

ノースバウンド API を使用したネットワークの追加

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

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



```

    <tenant>
      <name>quicktest4</name>
    </tenant>
  </tenants>
</esc_datamodel>

```

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

```

<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>quicktest4</name>
    </tenant>
  </tenants>
</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 \(315 ページ\)](#) 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 \(315 ページ\)](#) 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 \(315 ページ\)](#) 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 \(315 ページ\)](#) 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>
    <valid>1</valid>
    <bus>ide</bus>
    <size>1</size>
    <sizeunit>GiB</sizeunit>
  </volume>
</volumes>
```

ESC によって作成されたブート可能ボリューム

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

次の例を参考にしてください。

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

アウトオブバンドボリューム

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

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



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

```
<volumes>
  <volume>
    <name>pre-existing</name>
    <valid>1</valid>
    <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>
    <valid>1</valid>
    <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>
    <valid>0</valid>
    <bus>ide</bus>
    <type>lvm</type>
    <boot_index>0</boot_index>
  </volume>
</volumes>
```

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

パラメータの説明

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

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

テナントボリューム API

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

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

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

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



- (注)
- テナントボリューム API は、NETCONF API と REST API の両方でサポートされています。
 - テナントボリューム API を使用して、エフ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>
                <valid>1</valid>
                <bus>ide</bus>
              </volume>
            </volumes>
            <interfaces>
              <interface>
                <nicid>0</nicid>
                <network>mynetwork</network>
              </interface>
            </interfaces>
            <scaling>
              <min_active>1</min_active>
              <max_active>1</max_active>
              <elastic>true</elastic>
            </scaling>
            <kpi_data>
              <kpi>
                <event_name>VM_ALIVE</event_name>
                <metric_value>1</metric_value>
                <metric_cond>GT</metric_cond>
                <metric_type>UINT32</metric_type>
                <metric_collector>
                  <type>ICMPPing</type>
                  <nicid>0</nicid>
                  <poll_frequency>3</poll_frequency>
                  <polling_unit>seconds</polling_unit>
                  <continuous_alarm>>false</continuous_alarm>
                </metric_collector>
              </kpi>
            </kpi_data>
          </deployment>
        </deployments>
      </tenant>
    </tenants>
  </esc_datamodel>
</esc_datamodel>
```

```
</kpi_data>
<rules>
  <admin_rules>
    <rule>
      <event_name>VM_ALIVE</event_name>
      <action>"ALWAYS log"</action>
      <action>"TRUE
        servicebooted.sh"</action>
      <action>"FALSE recover
        autohealing"</action>
    </rule>
  </admin_rules>
</rules>
<config_data />
</vm_group>
</deployment>
</deployments>
</tenant>
</tenants>
</esc_datamodel>
```

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



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



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 \(43 ページ\)](#)
- [VMware vCenter での分散ポートの作成 \(44 ページ\)](#)

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>
```

```

    </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 \(318 ページ\)](#) 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>vdsSwitch-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 \(47 ページ\)](#)

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 \(126 ページ\)](#) 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 vSphere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>
    <valid>1</valid>
  </volume>
</volumes>
```



第 6 章

ESC リソースの管理

- [VIM コネクタの管理 \(49 ページ\)](#)

VIM コネクタの管理

VIM コネクタには、URL や認証クレデンシヤ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ヤルは、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 \(103 ページ\)](#) 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 \(52 ページ\)](#) 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 \(103 ページ\)](#) 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 \(50 ページ\)](#) 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 \(103 ページ\)](#) 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 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 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>
  </properties>
</vim_connector>

```

```

    <!-- 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_conf.d.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 (User Authentication)	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 \(49 ページ\)](#) 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 \(61 ページ\)](#)
- [AWS の VIM コネクタ設定 \(69 ページ\)](#)
- [VMware vCloud Director \(vCD\) の VIM コネクタの設定 \(70 ページ\)](#)

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 \(50 ページ\)](#) 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	Permission	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	<code>/etc/keystone/policy.json</code> "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 <code>identity: create_grant</code> 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	<code>/etc/keystone/policy.json</code> "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	<code>/etc/glance/policy.json</code> "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。 <pre><image> <name>mk-test-image</name> ... <disk_bus>virtio</disk_bus> <visibility>private</visibility> </image></pre>
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	Permission	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	<code>/etc/nova/policy.json</code> "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	<code>/etc/nova/policy.json</code> "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	<code>/etc/neutron/policy.json</code> "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	<code>/etc/neutron/policy.json</code> "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。 <pre>< n e t w o r k > <name>provider-network</name> <!-- <shared>>false</shared> //default is t r u e - - > <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></pre>

ESC VIM 操作	説明	Permission	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	<code>/etc/neutron/policy.json</code> "get_subnet"	<p>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。</p> <p>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。</p> <pre>< n e t w o r k > <name>esc-created-network</name> <!--network must be created by E S C - - > <admin_state>>false</admin_state> < s u b n e t > <name>makulandyescextnet1-subnet1</name> <ipversion>ipv4</ipversion> < d h c p > t r u e < / d h c p > <address>10.6.0.0</address> <netmask>255.255.0.0</netmask> </subnet> </network></pre>
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	Permission	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> < n i c i d > 0 < / n i c i d > <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> < d h c p > f a l s e < / d h c p > <address>172.16.0.0</address> <netmask>255.255.255.0</netmask> <gateway>172.16.0.1</gateway> < / s u b n e t > < s h a r e d _ i p > < n i c i d > 0 < / n i c i d > <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	Permission	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	<code>/etc/neutron/policy.json</code> "update_port:allowed_address_pairs"	<pre><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></pre>
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	Permission	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	<code>/etc/nova/policy.json</code> <code>"os_compute_api:servers:create:forced_host"</code>	<code><placement> <type>zone_host</type></code> <code><enforcement>strict</enforcement></code> <code><host>anyHOST</host> </placement></code>
	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（50 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>
      </properties>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

```

        <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 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>
    </vim_connector>
  </vim_connectors>
</esc_system_config>

```

```

    </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 delete-vimuser
AWS_EAST_2 AWS_ACCESS_ID
```

```
[admin@localhost ~]# esc_nc_cli 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 \(132 ページ\)](#) 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>
```




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 \(73 ページ\)](#)

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 \(49 ページ\)](#) 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	<pre><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></pre>	詳細については、 OpenStack の VIM コネクタの設定 (61 ページ) を参照してください。

VIM	プロパティ	参考資料
AWS	<pre> <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> </pre>	<p>詳細については、AWSのVIMコネクタ設定（69ページ）を参照してください。</p>
VMware vCD	<pre> <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> </pre>	<p>詳細については、VMware vCloud Director (vCD) のVIMコネクタの設定（70ページ）を参照してください。</p>

VIM	プロパティ	参考資料
Cisco Cloud Services Provider (CSP) 2100	<pre><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> </credentials> </user> </users></pre>	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 (333 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 \(77 ページ\)](#)
- [設定データの暗号化 \(83 ページ\)](#)

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>
              </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>
              </remote_file>
            </file_locator>
          </file_locators>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```

```

        <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>
    </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>
        </kpi>
    </kpi_data>
</vm_group>

```



```

        <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、ESCVM 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な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-riden 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>
```

デイレゾ設定の例

```
<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>
      </vm_group>
      <config_data>
        <!-- exisiting configuration example - remains valid -->
      </config_data>
    </deployment>
  </tenant>
</tenants>
```

```

<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>

```

デイズロ設定および LCS アクションの詳細については、「[デイズロ設定](#)」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:esbigdata;
    }
  }
  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。



第 III 部

仮想ネットワーク機能のオンボーディング

- [仮想ネットワーク機能のオンボーディング \(87 ページ\)](#)



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 \(87 ページ\)](#)
- [VMware vCenter での仮想ネットワーク機能のオンボーディング \(90 ページ\)](#)

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ー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、ネットワークング、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、デイズロ設定、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. デイズロ設定の準備
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 \(145 ページ\)](#) 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 \(141 ページ\)](#) 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となります。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ズ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 \(99 ページ\)](#) 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 \(90 ページ\)](#) 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 に渡されるデイズロ設定ファイルは、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、デイズ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、ネットワークキング、デイズロ設定、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. デイズロ設定の準備
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>
                <valid>1</valid>
              </volume>
            </volumes>
          </vm_group>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```

```

    </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>
              <valid>1</valid>
            </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>vds1</switch_name>
      <vlan_id>0</vlan_id>
    <subnet>
      <name>vnf_management-subnet</name>
      <ipversion>ipv4</ipversion>
    </subnet>
  </networks>
</deployment>

```

```

        <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 \(145 ページ\)](#) 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 \(141 ページ\)](#) 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となります。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フォルト設定を適用します。詳細については、[VMware vCenter での仮想ネットワーク機能の展開 \(121 ページ\)](#) 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 \(88 ページ\)](#) 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

- [仮想ネットワーク機能の展開 \(97 ページ\)](#)
- [OpenStack での仮想ネットワーク機能の展開 \(99 ページ\)](#)
- [複数の VIM への仮想ネットワーク機能の展開 \(109 ページ\)](#)
- [ブラウンフィールド VM の展開 \(113 ページ\)](#)
- [VMware vCenter での仮想ネットワーク機能の展開 \(121 ページ\)](#)
- [Amazon Web Services での仮想ネットワーク機能の展開 \(131 ページ\)](#)
- [統合型の展開 \(137 ページ\)](#)
- [仮想ネットワーク機能の展開解除 \(139 ページ\)](#)
- [展開パラメータの設定 \(141 ページ\)](#)
- [デイズロ設定 \(145 ページ\)](#)
- [KPI、ルール、およびメトリック \(153 ページ\)](#)
- [ポリシー駆動型データモデル \(171 ページ\)](#)
- [サポート対象のライフサイクルステージ \(LCS\) \(173 ページ\)](#)
- [アフィニティおよびアンチアフィニティルール \(177 ページ\)](#)
- [OpenStack のアフィニティおよびアンチアフィニティルール \(179 ページ\)](#)
- [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 \(185 ページ\)](#)
- [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 \(191 ページ\)](#)
- [カスタム VM 名の設定 \(193 ページ\)](#)
- [既存の展開の管理 \(197 ページ\)](#)
- [展開状態とイベント \(229 ページ\)](#)
- [LCS を使用した VNF ソフトウェアのアップグレード \(237 ページ\)](#)

- [仮想ネットワーク機能の操作 \(249 ページ\)](#)



第 11 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 \(97 ページ\)](#)

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 12 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 \(99 ページ\)](#)
- [複数の OpenStack VIM への VNF の展開 \(103 ページ\)](#)

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。	<ul style="list-style-type: none">• 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す。	<ul style="list-style-type: none"> 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_NwDepModVmGrpl</depname>
    <tenant>admin</tenant>
    <tenant_id>62cd11f560b44bf5815eaad41fc94c80</tenant_id>
  </event>
</notification>
```

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_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>
          <!-- Admin Rules -->
          <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 -->
          <config_data />
          <!-- Scaling -->
          <scaling>
            <min_active>1</min_active>
            <max_active>2</max_active>
            <elastic>true</elastic>
          </scaling>
          <!-- Recovery Policy -->
          <recovery_policy>
            <recovery_type>AUTO</recovery_type>
            <action_on_recovery>REBOOT_ONLY</action_on_recovery>
            <max_retries>1</max_retries>
          </recovery_policy>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```



```

        </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 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 \(99 ページ\)](#)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 \(49 ページ\)](#) 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する必要があります。
- `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（100 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 attribute. -->
      <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 のアフィニティおよびアンチアフィニティルール \(179 ページ\)](#) 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 \(ポリシーフレームワークを使用\) \(286 ページ\)](#) 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 \(197 ページ\)](#) を参照してください。



第 13 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 \(109 ページ\)](#)
- [マルチ VIM 展開でサポートされる機能 \(110 ページ\)](#)

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 organization -->
              <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 <small>start/stop/start/resize/monitor/resize/monitor</small>	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 14 章

ブラウフィールド VM の展開

ここでは、ESCでの単純なブラウフィールド展開の展開方法を示します。VMが作成されてインポートされると、ブラウフィールド展開としてESCに展開されます。ブラウフィールド展開が成功すると、ESCはESCで作成された他のVMと同様にそのVMを監視および管理します。VMの展開の詳細については、[VMの展開（115ページ）](#)を参照してください。

- [VNFのインポート（113ページ）](#)
- [VMの展開（115ページ）](#)
- [VMの作成（117ページ）](#)
- [VMの展開解除（117ページ）](#)
- [ブラウフィールドモードでのVMの操作（117ページ）](#)

VNFのインポート

アクティブ VNF をインポートするには、次の手順を実行します。

1. RPC ロードを通じて、ランタイムデータを含む XML を ESC に送信します。

次に例を示します。

```
esc_nc_cli import-deployment-data CREATE admin name-dep /opt/cisco/existing_vms.xml
```

2. 次のコマンドを使用して、関連する dep.xml ファイルを ESC に展開します。

```
esc_nc_cli edit-config dep.xml
```

3. 必要に応じて、展開の問題を修正します。問題を修正するには、次の手順を実行します。

1. インポートしたデータを再発行します（ステップ 1）。

2. dep.xml ファイルを展開解除します。

次に例を示します。

```
esc_cli delete-dep aTenantName aDeploymentName
```

3. CLI を使用して dep.xml を再展開します。

次に例を示します。

```
esc_cli edit-config dep.xml
```

4. 必要に応じてステップ a ~ c を繰り返します。

4. RPC を呼び出して、ESC が VNF を完全に管理することを宣言します。

次に例を示します。

```
esc_nc_cli import-deployment-data FINALIZE admin name-dep
```

サンプル import.xml の例

```
<import>
  <vms>
    <vm_details> <!--First VM details-->
      <flavor/>
      <host/>
      <host_id/>
      <image/>
      <port/>
      <uuid>1bb008e1-d4dd-4107-bc45-7e298a5ac510</uuid>
      <name>parvmoha-bf-vm</name>
      <attached_volume/>
    </vm_details>
    <vm_details>
      ..... <!--add more vms using vm_details tag-->
    </vm_details>
  </vms>
  <deployment_name>parvmoha-dep</deployment_name>
  <project_name>admin</project_name>
  <project_uuid>563fba7044c847a6a370cc10d5ef7d57</project_uuid>
</import>
```

dep.xml の例 :

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>aTenantName</name>
      <managed_resource>>false</managed_resource>
      <deployments>
        <deployment>
          <name>brownfield-deployment</name>
          <vm_group>
            <name>g2</name>
            <vim_vm_name>vm-cirros</vim_vm_name>
            <image>Automation-Cirros-Image</image>
            <flavor>Automation-Cirros-Flavor</flavor>
            <bootup_time>100</bootup_time>
            <recovery_wait_time>0</recovery_wait_time>
            <interfaces>
              <interface>
                <nicid>0</nicid>
                <network>esc-net</network>
              </interface>
            </interfaces>
          </vm_group>
          <vim_interface_name>vm-cirros-interface</vim_interface_name>
        </deployment>
      </deployments>
      <kpi_data>
        <kpi>
          <event_name>VM_ALIVE</event_name>
          <metric_value>1</metric_value>
          <metric_cond>GT</metric_cond>
          <metric_type>UINT32</metric_type>
        </kpi>
      </kpi_data>
    </tenant>
  </tenants>
</esc_datamodel>
```

```

        <metric_occurrences_true>2</metric_occurrences_true>
        <metric_occurrences_true>3</metric_occurrences_true>
        <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>1</max_active>
    <elastic>>true</elastic>
</scaling>
<recovery_policy>
    <recovery_type>AUTO</recovery_type>
    <action_on_recovery>REBOOT_THEN_REDEPLOY</action_on_recovery>

    <max_retries>1</max_retries>
</recovery_policy>
</vm_group>
</deployment>
</deployments>
</tenant>
</tenants>
</esc_datamodel>

```

インポートの制限

- VM グループごとに 1 つの VM のみがサポートされます。
- スケーリングはサポートされていません。

VM の展開

展開中にインポートされたコンポーネントが更新されていないことを確認することをお勧めします。たとえば、VM、サブネット、または OpenStack で実行されているポートです。

次の例は、OpenStack にブラウフィールドを展開する方法を示しています。

```

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

```

```

<deployment>
  <name>brownfield-deployment</name>
  <vm_group>
    <name>g2</name>
    <vim_vm_name>vm-cirros</vim_vm_name>
    <image>Automation-Cirros-Image</image>
    <flavor>Automation-Cirros-Flavor</flavor>
    <bootup_time>100</bootup_time>
    <recovery_wait_time>0</recovery_wait_time>
    <interfaces>
      <interface>
        <nicid>0</nicid>
        <network>esc-net</network>
        <vim_interface_name>vm-cirros-interface</vim_interface_name>
      </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>2</metric_occurrences_true>
        <metric_occurrences_true>3</metric_occurrences_true>
        <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>1</max_active>
      <elastic>>true</elastic>
    </scaling>
    <recovery_policy>
      <recovery_type>AUTO</recovery_type>
      <action_on_recovery>REBOOT_THEN_REDEPLOY</action_on_recovery>

      <max_retries>1</max_retries>
    </recovery_policy>
  </vm_group>
</deployment>
</deployments>
</tenant>
</tenants>
</esc_datamodel>

```



- (注) 展開モードがブラウフィールドの [ファイナライズされたインポート (Finalized Import)] としてマークされている場合、VNF が ESC によって完全に管理されていることを意味します。ただし、完全にインポート済みとしてマークされている場合、この展開をインポートモードに戻すことはできません。

VM の作成

展開後、実際のコールをバイパスして、ブラウフィールドモードの OpenStack でポートと VM を作成します。

ESC では、dep.xml 内の展開およびテナント名だけが import.xml を使用してロードされた内容と一致する場合、ブラウフィールド展開が検出されます。ブラウフィールド展開が検出されると、ESC は VIM が稼働しているかどうか、dep.xml で指定されたイメージとフレーバーが使用可能かどうかをチェックします。

ESC は、dep.xml で指定されたカスタムポートがアクティブかどうかもチェックします。VIM ドライバは、VM の作成要求を受信すると、実際の OpenStack の作成操作をバイパスし、DB から読み取った VNF の UUID を返します。

VM の展開解除

ブラウフィールド展開が正常に展開されると、ブラウフィールドモードを [ファイナライズされたインポート (Finalized Import)] に変更した後にのみ、ESC によって完全に管理されます。

展開がブラウフィールドモードの場合、VIM で VM を削除せずにブラウフィールド展開を ESC から展開解除できます。

展開が成功したら、ブラウフィールドモードを [ファイナライズされたインポート (Finalized Import)] に変更できます。ブラウフィールドモードが [ファイナライズされたインポート (Finalized Import)] の場合、VIM から VM が削除されます。

ブラウフィールドモードでの VM の操作

ブラウフィールドモードで VM のすべての操作をブロックします。ブラウフィールド展開が発生すると、VM が展開され、VM_ALIVE 状態がトリガーされます (該当する場合)。その後、展開は SERVICE_ACTIVE 状態に移行します。

ブラウフィールド展開が展開されていて、ブラウフィールドモードが [オン (ON)] の場合、手動でトリガーされたすべての VM アクション (START、STOP、RECOVER、REBOOT、ENABLE/DISABLE MONITOR) がブロックされます。

ブラウフィールド展開の一部として展開されたVMは、最初はブラウフィールドモードが [オン (ON)] の状態です。ブラウフィールドモードを手動でオフにした後のみ、VMはESCによって完全に管理されます。したがって、ブラウフィールドモードでのVM操作（手動または自動トリガー）はブロックされます。

ユーザが、まだ完了していないブラウフィールド展開に対してサービスアクションまたはVMアクションを手動でトリガーしようとする、検証エラーが表示されます。次に、検証エラーメッセージの例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<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:recoveryVmAction
    </error-path>
    <error-message xml:lang="en">Exception from action callback: Recovery VM action
action not allowed in brownfield mode.</error-message>
    <error-info>
      <bad-element>recoveryVmAction</bad-element>
    </error-info>
  </rpc-error>
</rpc-reply>
```

MONA トリガーイベントのブロッキング

ブラウフィールドの展開が成功すると、VM_DEPLOYEDおよびVM_ALIVEイベントが送信されます。展開がSERVICE_ACTIVE状態に移行します。この展開では、ブラウフィールドモードが [オン (ON)] になっています。

展開を手動で終了する前にMONAからRECOVERYイベントがトリガーされると、展開はSERVICE_ERROR状態に移行し、インポートが失敗したことを知らせる通知がESCからノースバウンドに送信されます。イベントに影響するその他のVMはブロックされ、MONAによってトリガーされたVMアクションがブロックされたことを知らせるVM_ACTION_BLOCKED通知がESCからノースバウンドに送信されます。

例：

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2019-10-03T14:06:53.042+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>VM Action [VM_RECOVERY_AUTO] Blocked for VM
[parvmiha-cirros-_Grp1_0_89192a9f-d624-46c7-a264-bb8326ec5d29].</status_message>
    <event>
      <type>VM_ACTION_BLOCKED</type>
    </event>
  </escEvent>
</notification>
```

ブラウフィールドモードを [オフ (OFF)] に切り替えて、VMを手動で回復できます。この場合、ESCの変更がブロックされたため、サービスはSERVICE_ACTIVE状態のままです。ブ

ラウンフィールドモードが [オフ (OFF)] になると、サービスが `SERVICE_ACTIVE` 状態になるため、`CONF` の `svc-action` は手動リカバリを拒否します。ブラウフィールドモードを [オフ (OFF)] にした後、VM を手動で回復するには、`recovery-vm-action` コマンドを使用します。



(注) `VM_ALIVE` イベント通知は、`SERVICE_ACTIVE` 状態への最初の移行に必要なためブロックされません。



第 15 章

VMware vCenter での仮想ネットワーク機能の展開

- [VMware vCenter での仮想ネットワーク機能の展開 \(121 ページ\)](#)
- [単一の VMware vCenter VIM での VNF の展開 \(122 ページ\)](#)
- [VMware vCloud Director \(vCD\) での仮想ネットワーク機能の展開 \(126 ページ\)](#)

VMware vCenter 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、VMware に VNF を展開する手順について説明します。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単一 VIM への 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。	<ul style="list-style-type: none">• 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。	<ul style="list-style-type: none">• 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。	<ul style="list-style-type: none"> 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。	<ul style="list-style-type: none"> 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、VIM タイプの VMware vSphere での IPv6 展開をサポートしていません。

単一の VMware vCenter VIM での VNF の展開

VNF の展開は、ESC ポータルまたはノースパ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) 複数の VIM での VNF の展開は、VMware vCenter ではサポートされていません。



- (注) 単一の ESC インスタンスは、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。「ネットワーク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 を選択できません。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。VDC での VNF の展開の詳細については、を参照してください。

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。



(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>
      </opdata>
    </esc_datamodel>
  </data>
</rpc-reply>

```

```

</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、Elastic Services Controller (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 \(47 ページ\)](#) を参照してください。



(注) ESC は VMware vCloud Director 8.2 をサポートしています。

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>
  <!-- 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>

```




第 16 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 \(131 ページ\)](#)

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。	<ul style="list-style-type: none"> 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 \(132 ページ\)](#) 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 \(69 ページ\)](#) 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>
          </vm_group>
          <interfaces>
            <interface>
              <nicid>0</nicid>
            </interface>
          </interfaces>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```



```
<network>vpc-d7eelbac</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 17 章

統合型の展開

- [統合型の展開 \(137 ページ\)](#)

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 18 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 \(139 ページ\)](#)

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。VMAlive 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 19 章

展開パラメータの設定

- [導入パラメータ \(141 ページ\)](#)

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、またはデイレゾ設定です。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
デイレゾ設定	デイレゾ設定は、次のいずれかの方法で行います。 <ul style="list-style-type: none"> • NETCONF API • REST API • ESC ポータル 	デイレゾ設定は、次のいずれかの方法で行います。 <ul style="list-style-type: none"> • NETCONF API • REST API • ESC ポータル 	<ul style="list-style-type: none"> • 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。 <ul style="list-style-type: none"> • 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。 <ul style="list-style-type: none"> • 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。 <ul style="list-style-type: none"> • 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	<p>展開解除は、次のいずれかの方法で行います。</p> <ul style="list-style-type: none"> • NETCONF API • REST API • ESC ポータル 	<p>VNF の展開解除は、次のいずれかの方法で行います。</p> <ul style="list-style-type: none"> • NETCONF API • REST API • ESC ポータル 	<p>VNF の展開解除は、次のいずれかの方法で行います。</p> <ul style="list-style-type: none"> • NETCONF API • REST API • ETSI API
アフィニティールールとアンチアフィニティールール	<p>アフィニティールールとアンチアフィニティー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。</p> <ul style="list-style-type: none"> • NETCONF API • REST API 	<p>アフィニティー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。</p> <ul style="list-style-type: none"> • 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）。 	<p>アフィニティールールとアンチアフィニティー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。</p> <ul style="list-style-type: none"> • NETCONF API • REST API • ETSI API
VNF 操作	<p>VNF 操作は、次のいずれかの方法で行います。</p> <ul style="list-style-type: none"> • REST API • NETCONF API • ESC ポータル 	<p>VNF 操作は、次のいずれかの方法で行います。</p> <ul style="list-style-type: none"> • REST API • NETCONF API • ESC ポータル <p>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。</p>	<p>VNF 操作は、次のいずれかの方法で行います。</p> <ul style="list-style-type: none"> • 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	N/A	<p>マルチクラスタ設定は、次のいずれかの方法で行います。</p> <ul style="list-style-type: none"> • REST API • ESC ポータル <p>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。</p>	N/A
複数の仮想データセンター (マルチ VDC)	N/A	<p>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。</p> <ul style="list-style-type: none"> • REST API • ESC ポータル 	N/A
ハードウェアアクセラレーション	<p>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。</p> <ul style="list-style-type: none"> • NETCONF API • REST API <p>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。</p>	N/A	N/A
単一のルート I/O 仮想化	<p>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。</p> <ul style="list-style-type: none"> • NETCONF API • REST API 	<p>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。</p> <ul style="list-style-type: none"> • NETCONF API • REST API 	N/A

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 \(99 ページ\)](#) を参照してください。



第 20 章

デイゼロ設定

- [デイゼロ設定 \(145 ページ\)](#)
- [データモデルの設定のデイゼロ \(145 ページ\)](#)
- [vCD 展開のデイゼロ設定 \(150 ページ\)](#)

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、およびスケールアップ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、インターフェースの起動、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ー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。各ユーザー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。ユーザーデータは、`<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	string
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は展開中にデイズロ設定ファイルの\$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}uf2qLUwGTjHgZp1kP78xA
```

- 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$uf2qLUwGTjHgZp1kP78xA ]]#
```

ファイルロケータ

外部設定ファイルを取得するため、デイズ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ダ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ダイゼロ設定の例：

```
<?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>
              <!-- existing 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"
                              indicates 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、ESCVM 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 展開のデイレゼロ設定

vCD 展開のデイレゼロ設定は、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レゼロ設定 :

```

</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レゼロ設定 :

```
</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レゼロ設定：

```
<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 \(126 ページ\)](#) を参照してください。


```

    </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せん。

KPI とルールの更新については、[KPI とルールの更新 \(209 ページ\)](#) 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>ICMPING</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 ICMPING 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

グは、dynamic_mappings.xml ファイルを使用して有効化されていました。このファイルは ESC VM に保存されていて、テキストエディタを使用して変更していました。ESC 2.2 以降には、esc-dynamic-mapping ディレクトリおよび dynamic_mappings.xml ファイルはありません。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 ESC 以外の場所（ホームディレクトリなど）にバックアップします。
2. ESC VM に esc-dynamic-mapping ディレクトリを作成します。読み取り権限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 (Response)	説明
Read	internal/dynamic_mapping/actions/ <action_name>	GET	該当なし	アクション XML	名前アクションを取得する
Read All	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 (Response)	説明
削除 (Delete)	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 (Response)	説明
Read	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
Read All	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 (Delete)	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 \(155 ページ\)](#) 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せん。

- [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、マスター HA インスタンスとバックアップ 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>
 </properties>
 </metaData>
 </action>
</actions>

```

```

 </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](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
type	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。	<p>progress-steps タイプの場合 :</p> <pre>{current_step} {total_steps}</pre> <p>progress-percentage タイプの場合 :</p> <pre>{percentage}</pre>
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の UUID です。



第 22 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 \(171 ページ\)](#)

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。

ポリシー駆動型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。[ポリシーデータモデル \(171 ページ\)](#) 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。`#unique_116 unique_116_Connect_42_section_7FC6CE2FEC9244B489674DBA94904393`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 \(289 ページ\)](#) 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>
  </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 \(286 ページ\)](#) 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 \(286 ページ\)](#) 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 \(238 ページ\)](#) を参照してください。



第 23 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 \(LCS\) \(173 ページ\)](#)

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 (Condition Name)	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ズ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>
      <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 -->
          </deployment>
          <vm_group>
            <name>ASA-group</name>
          </vm_group>
        </deployments>
      </tenant>
    </tenant>
  </tenants>
</esc_datamodel>
```



```

<!-- 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 (Condition Name)	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 24 章

アフィニティおよび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 \(177 ページ\)](#)

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 VNF 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
affinity	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 25 章

OpenStack のアフィニティおよびアンチアフィニティルール

- [OpenStack のアフィニティおよびアンチアフィニティルール \(179 ページ\)](#)

OpenStack のアフィニティおよび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せん。

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せん。

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ルリング更新 (VM を追加するための最小スケールリングの増加)
 - 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 \(103 ページ\)](#) を参照してください。



第 26 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 \(185 ページ\)](#)

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>tlg1</vm_group>
<vm_group>tlg2</vm_group>
<vm_group>tlg7</vm_group>
</placement_group>
<placement_group>
<name>placement-affinity-2</name>
<type>affinity</type>
<enforcement>strict</enforcement>
<vm_group>tlg3</vm_group>
<vm_group>tlg4</vm_group>
</placement_group>
<placement_group>
<name>placement-affinity-3</name>
<type>affinity</type>
<enforcement>strict</enforcement>
<vm_group>tlg5</vm_group>
<vm_group>tlg6</vm_group>
</placement_group>
<placement_group>
<name>placement-anti-affinity-1</name>
<type>anti_affinity</type>
<enforcement>strict</enforcement>
<vm_group>tlg1</vm_group>
<vm_group>tlg3</vm_group>
<vm_group>tlg5</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 27 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 \(191 ページ\)](#)

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 \(126 ページ\)](#) を参照してください。



第 28 章

カスタム VM 名の設定

・ [カスタム VM 名の設定 \(193 ページ\)](#)

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>xyzy</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_instance>
              </vm_group>
            </deployments>
          </tenant>
        </tenants>
      </opdata>
    </esc_datamodel>
  </data>

```

- 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 29 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 \(197 ページ\)](#)

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グループのデイズロ設定、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、ノースパ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 グループのデフォルト設定の更新	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ズロ設定をサポートしません。デイズロ設定の一部として、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、デイズロ設定を更新する必要があります。将来、ESC は新しいデイズロ設定をリカバリに使用しません。

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。

既存のデイズ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>
</escEvent>
<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>ul-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>ul-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>

```

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
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。	Rejected
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

要求	サポート対象または拒否
対応するネットワークなしで IP プールを追加します。	Rejected
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。	Rejected
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。	Rejected

VM グループのデイレゾ設定の更新

既存の展開で VM グループのデイレゾ設定を更新（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。新しいデイレゾ設定ファイルは、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) 既存のデイレゾ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-lee6-4147-8c23-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（145 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>UIN32</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ルアップセクションで `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 (New Value)	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。

シナリオ	古い値	新しい値 (New Value)	アクティブ値
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い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。
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とも展開は有効です。	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。

シナリオ	古い値	新しい値 (New Value)	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 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とも展開は有効です。	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 -->
    </static_ip_address_pool>
  </scaling>
</vm_group>
</esc_datamodel>
```

```

    <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>
  </vm_group>
</esc_datamodel>

```

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>
</vm_group>

```

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>
</vm_group>

```

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>
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	

ロールバック条件	予想される動作	通知
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。	<pre> ?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> </pre>

ロールバック条件	予想される動作	通知
<p>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。</p>	<p>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。</p>	

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 organization -->
        <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>
        </kpi>
    </kpi_data>
</vm_group>

```

```

        <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>
  </vm_group>

```

```

<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>
  </scaling>
</vm_group>

```



```

    </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 organization -->
    <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 organization -->
    <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>
  </scaling>
</vm_group>

```

```
<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 30 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 \(229 ページ\)](#)
- [イベント通知またはコールバックイベント \(231 ページ\)](#)

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パ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 VNF 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 (Deployment Action)	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 (Deployment Action)	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 (Deployment Action)	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	Subnet	サブネットが作成されました
DELETE_SUBNET	Subnet	サブネットが削除されました
CREATE_IMAGE	イメージ (Image)	イメージが作成されました
DELETE_IMAGE	イメージ (Image)	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 31 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](#) (237 ページ)
- [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](#) (238 ページ)
- [展開内の VNF のアップグレード](#) (247 ページ)

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。esc_nc_cli edit-config csr_dep2.xml コマンドを実行します。
-

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>
              <valid>0</valid>
            </volume>
            <volume>
              <name>v2.0</name>
              <valid>1</valid>
              <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>
            </actions>
          </policy>
        </policies>
      </deployment>
    </tenant>
  </tenants>
</esc_datamodel>
```

```

        <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、**valid** が 0 の既存のボリューム v1.0 が削除されます。**valid** 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](#)

ル (171 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 (Condition Name)	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>
    </action>
  </actions>
</policy>
```

```

        <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
18fbcf5d5-8b63-44e0-97ec-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,valid=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 8e96b8830d7bfbb337ce665586210fccca9644cbe238240e207350735
    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-calf-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,valid=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-calf-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 8e96b8830d7bfb337ce665586210fcc9644cbe238240e207350735
    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-calf-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 8e96b8830d7bfb337ce665586210fcc9644cbe238240e207350735
  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-calf-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_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 \(160ページ\)](#) 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 \(175ページ\)](#) 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 \(197ページ\)](#) 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 (Error)]

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 (Error)]

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 32 章

仮想ネットワーク機能の操作

- [VNF 操作 \(249 ページ\)](#)
- [個々の VNF と複合 VNF の管理 \(250 ページ\)](#)

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。

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

- [仮想ネットワーク機能のモニタリング \(253 ページ\)](#)
- [D-MONA を使用した VNF のモニタリング \(265 ページ\)](#)
- [ローカル MONA と D-MONA 間での VNF モニタリングの切り替え \(271 ページ\)](#)
- [仮想ネットワーク機能のスケーリング \(273 ページ\)](#)
- [仮想ネットワーク機能の修復 \(279 ページ\)](#)



第 33 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 (253 ページ)
- モニタリング方式 (260 ページ)
- VM のモニタリング (261 ページ)
- VM モニタリングステータスの通知 (263 ページ)
- モニタリング操作 (264 ページ)

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 22: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。	
type	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 23:メトリック メタデータ セクション

[タグ名 (Tag Name)]	説明	値
type	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 24:アクション

[タグ名 (Tag Name)]	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 (Tag Name)]	説明	値
type	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 25: アクションメタデータ セクション

[タグ名 (Tag Name)]	説明	値
type	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 26: 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<p>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)。</p>

タイプ	プロパティと各プロパティの説明
custom_script_threshold	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • esc_url • vm_external_id • vm_name • esc_event • event_name
snmp_get	<ul style="list-style-type: none"> • 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 SNMPv2c コミュニティストリング。
snmp_get_threshold	<ul style="list-style-type: none"> • 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 SNMPv2c 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	<ul style="list-style-type: none"> • 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

ESC2.1 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、プライマリ VM とセカンダリ 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 \(153 ページ\)](#) 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>UIN32</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>
```

```

    </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ップします。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 34 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 \(265 ページ\)](#)
- [D-MONA の展開 \(265 ページ\)](#)
- [D-MONA の設定 \(266 ページ\)](#)
- [モニタリングのための D-MONA を使用した VNF の展開 \(267 ページ\)](#)
- [D-MONA の回復 \(269 ページ\)](#)
- [D-MONA ログの取得 \(269 ページ\)](#)

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。

D-MONA を展開する必要があります。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。

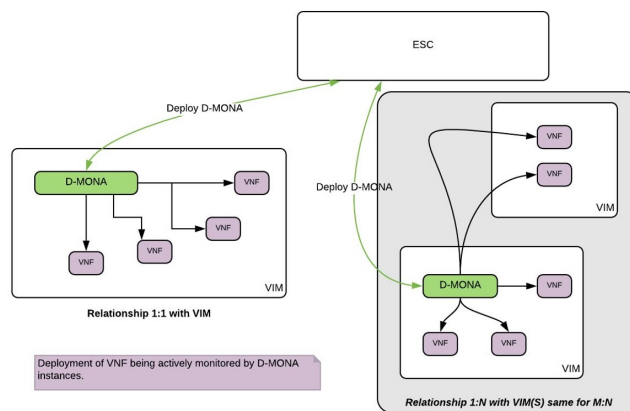
D-MONA の展開

ESC は、VIM の 1 対 1 D-MONA 展開をサポートします。単一の D-MONA インスタンスが単一の VIM で VNF をモニタします。

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、各 VNF の稼働状態を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。

D-MONA の設定

ESC イメージを再利用する D-MONA。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 に指定されるデイズロ設定によって制御されま。デイズロ設定の詳細については、「[デイズロ設定](#)」を参照してください。

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/iser-data.template</file>
  <variable>
    <name>vm_credentials</name>
    <val>REPLACED_WITH_GENERATED_PWD</val>
  </variable>
</configuration>
```

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>

```

モニタリングのための D-MONA を使用した VNF の展開

モニタリングのために D-MONA を使用して VNF を展開するには、同じ `vim_connector` 内に、`monitoring.agent.vim.mapping` day-0 変数を `true` に設定した D-MONA が必要です。ESC が D-MONA を検出した場合のみ、VNF のモニタリングがその D-MONA に割り当てられます。それ以外の場合、ローカル MONA は以前のすべての ESC リリースに従ってモニタリングを処理します。

次の例は、D-MONA ESC 展開記述子を示しています。

```

<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>sample</name>
      <deployments>
        <deployment>
          <name>sample-dmona-dep</name>
          <vm_group>
            <name>g1</name>
            <!-- Image version you want to use for dmona deployment. Image
must already exist in VIM -->
            <image>ESC-5_0_0_<latest></image>
            <flavor>m1.large</flavor>
            <bootup_time>600</bootup_time>
            <recovery_wait_time>0</recovery_wait_time>
            <interfaces>
              <interface>
                <nicid>0</nicid>
                <network>esc-net</network>
              </interface>
            </interfaces>
          </vm_group>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>

```

```

<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>HTTPGET</type>
      <nicid>0</nicid>
      <poll_frequency>3</poll_frequency>
      <polling_unit>seconds</polling_unit>
      <continuous_alarm>>false</continuous_alarm>
      <properties>
        <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>
<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>
  <!-- day 0 configuration -->
  <configuration>
    <dst>--user-data</dst>

<file>file:///opt/cisco/esc/esc-config/dmona/user-data.template</file>
    <variable>
      <name>vm_credentials</name>
      <val><REPLACE_WITH_GENERATED_PWD></val>
      <!--password field will look something like the
following-->
      <!--
<val>${RANDOM}56000${RANDOM}71z9Nf7457E1sEG8nDieDh6241W13Q8Psp9Nts.OZBe9rG/IE56W0kDzB.DsJAtj9pBrE.rSQw112r0N/</val>-->
    </variable>
  </configuration>
</configuration>
    <dst>/opt/cisco/esc/mona/dmona.crt</dst>
    <data>${DMONA_CERT}</data>
  </configuration>
</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-->
        <variable>
          <name>monitoring.agent.vim.mapping</name>
          <val>true</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>dmona</val>
        </variable>
        <variable>
          <name>security_user_password</name>
          <val>defaultUser</val>
        </variable>
      </configuration>
    </config_data>
  </vm_group>
</deployment>
</deployments>
</tenant>
</tenants>
</esc_datamodel>

```

D-MONA の回復

D-MONA を完全に回復できます。リカバリプロセス中は、D-MONA による VNF のモニタリングはできません。D-MONA リカバリが正常に完了した場合にのみ、各 VNF モニタリンググループを再プログラミングすることで、VNF モニタリング状態が自動的に更新されます。

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。

```

MethodType:
GET
MONA EndPoint:
https://ip-address:8443/mona/v1/files/getLogs
HTTPSRequestHeaders:
--remote-name --remote-header-name --write-out "Downloaded %{filename_effective} file"
--silent -k -u <username>:<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 ユーザガイド [英語] の「Monitoring VNF Using D-MONA」の章を参照してください。



第 35 章

ローカル MONA と D-MONA 間での VNF モニタリングの切り替え

ここでは、ローカル MONA から同じ VIM コネクタ内に導入された D-MONA モニタリングへの VNF モニタリングの切り替えについて説明します。

- [前提条件 \(271 ページ\)](#)
- [ローカル MONA VNF を D-MONA に移行する手順 \(271 ページ\)](#)
- [D-MONA をローカル MONA VNF に移行する手順 \(272 ページ\)](#)

前提条件

VNF モニタリングをローカル MONA から D-MONA モニタリングに移行する前に完了する必要がある前提条件を次に示します。

手順

- ステップ 1** ESC 5.0 以降が必要です。
 - ステップ 2** D-MONA は、1 対 1 VIM マッピングを true に設定して展開する必要があります。
1 対 1 の展開の詳細については、「[D-MONA の展開](#)」を参照してください。
 - ステップ 3** VNF は、D-MONA と同様に同じ VIM コネクタに展開されていて存在している必要があります。
-

ローカル MONA VNF を D-MONA に移行する手順

次に、ローカル MONA VNF を D-MONA に移行する手順を示します。

手順

- ステップ1 ローカル MONA が 2 つの VNF で ESC をモニタしていることを確認します。
 - ステップ2 同じ VIM (VIM コネクタ) 内で 1:1 のマッピングを使用した D-MONA 展開が True に設定され、ESC MONA が D-MONA VM をアクティブに監視していることを確認します。
 - ステップ3 ノースバウンド API を使用して、VNF-2 のモニタリングを設定解除する要求を送信します。これによりローカル MONA による VNF-2 のモニタリングを停止します。
 - ステップ4 ノースバウンド API を使用して、ESC バックエンドサーバのモニタリングを開始する VNF-2 のモニタリングを設定する要求を送信します。同じ VIM (VIM コネクタ) 内に 1:1 のマッピングを使用した D-MONA が存在する場合、VNF-2 のモニタリングは自動的に D-MONA に割り当てられます。
-

D-MONA をローカル MONA VNF に移行する手順

D-MONA をローカル MONA VNF に移行する手順は次のとおりです。

手順

- ステップ1 ノースバウンド API を使用して、VNF-2 のモニタリングを設定解除する要求を送信します。VNF-2 の D-MONA モニタリングが停止します。
 - ステップ2 ノースバウンド API を使用して、D-MONA 仮想マシンを展開解除する要求を送信します。ESC から D-MONA が削除されます。
 - ステップ3 ノースバウンド API を使用して、ESC バックエンドサーバのモニタリングを開始する VNF-2 のモニタリングを設定する要求を送信します。同じ VIM (VIM コネクタ) 内に D-MONA がいない場合、VNF-2 のモニタリング機能は自動的にローカル MONA に再割り当てされます。
-



第 36 章

仮想ネットワーク機能のスケールリング

- [スケールリングの概要 \(273 ページ\)](#)
- [VM のスケールインとスケールアウト \(273 ページ\)](#)
- [スケールリングのためのリソースの一貫した順序付け \(275 ページ\)](#)
- [スケールリング通知とイベント \(276 ページ\)](#)

スケールリングの概要

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。スケールリングは、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 \(153 ページ\)](#) 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

スケール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、VMAlive 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プライマリ 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 NFW 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、デイレゾ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。

デイレゾ設定、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 \(141 ページ\)](#) 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	<p>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。</p> <pre><type>SERVICE_ALIVE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre><type> VM_SCALE_OUT_INIT</type> <status>SUCCESS</status></pre> <p>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。</p> <pre><type> VM_SCALE_OUT_DEPLOYED</type> <status>SUCCESS</status></pre> <p>展開が完了すると、次のNETCONF通知が送信されます。</p> <pre><type>VM_SCALE_OUT_COMPLETE</type> <status>SUCCESS</status></pre>

シナリオ	通知
スケールイン	<p>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。</p> <p>NETCONF通知が送信されます。</p> <pre data-bbox="980 436 1312 485"><type>SERVICE_ALIVE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="980 627 1365 676"><type> VM_SCALE_IN_INIT</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="980 856 1403 905"><type>VM_SCALE_IN_COMPLETE</type> <status>SUCCESS</status></pre>

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 37 章

仮想ネットワーク機能の修復

- [修復の概要 \(279 ページ\)](#)
- [VM の修復 \(279 ページ\)](#)
- [リカバリポリシーと再展開ポリシー \(286 ページ\)](#)
- [ホストの有効化と無効化 \(294 ページ\)](#)
- [通知とイベント \(296 ページ\)](#)

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。ESC は KPI を使用して VM を監視し、KPI 条件に基づいてイベントがトリガーされます。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。

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、修復ワークフローがリカバリポリシーを使用して VM の回復を試みます。

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によって異なります。たとえば、[再起動のみ (Reboot Only)]、[再展開のみ (Redeploy Only)]、または[再起動と再展開 (Reboot and Redeploy)]などのリカバリポリシーをユーザが設定した場合、ESC は同じ設定済みのポリシーに従います。

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 \(153 ページ\)](#) 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、同じデイズロ設定の新しい 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ています。

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

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、修復ワークフローがリカバリポリシーを使用して VM の回復を試みます。

ESC が展開後に 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によって異なります。

ESC には、VNF の展開時に指定できる次の VM リカバリポリシーがあります。

- 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。

VM リカバリのアクションには、次の 3 つのタイプがあります。

- 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、同じホストに VM を再展開するよう試みます。
- REBOOT_ONLY : VM ダウンイベントを受信されるか、タイマーが期限切れになると、修復ワークフローは VM の再起動のみを試みます。
- REDEPLOY_ONLY : VM ダウンイベントを受信されるか、タイマーが期限切れになると、修復ワークフローは VM の再展開のみを試みます。



(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 \(126 ページ\)](#) を参照してください。

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。さらに、リカバリアクションは、リカバリポリシーの action-on-recovery パラメータに基づいています。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 \(284 ページ\)](#)

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 Deployment Attributes](#)」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-gl_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-gl
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 を使用して手動回復を実行できます。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 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"
      }
    ]
  }
}
    
```

```
}
}
```

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。

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 \(282 ページ\)](#) を参照してください。

展開の手動回復用 API

NETCONF API と REST API を使用して手動回復を実行できます。

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 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。

- **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せん。

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。

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 \(171 ページ\)](#) 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 \(281 ページ\)](#) 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）

Cisco ESC リリース 2.2 以降では、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 \(281 ページ\)](#) 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。

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（286 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
<p>LCS::PRE_DEPLOY : 展開で VM を展開する直前に発生します。</p> <p>LCS:: POST_DEPLOY_ALIVE : 展開がアクティブになった直後に発生します。</p>	<ul style="list-style-type: none"> • 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。 	<p>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。</p> <p>SET_RECOVERY_REDEPLOY::SERIALIZED は、</p> <p>DROP_RECOVERIES アクションが使用される場合を選択します。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。</p>
<p>LCS::DEPLOY_ERR : 展開が失敗した直後に発生します。</p>	<p>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。</p>	-

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。	<ul style="list-style-type: none"> • 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 \(171 ページ\)](#) 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せん。



- (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>
        </policies>
      </deployment>
    </deployments>
  </tenant>
</tenants>
```

```

        </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 (Condition Name)	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

定義済みアクション名	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、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。

再展開ポリシーの詳細については、[再展開ポリシー \(289 ページ\)](#) 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 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せん。

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 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 \(264 ページ\)](#)、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 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せん。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>
```

```
</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、リカバリワークフローによって VM が再起動されて、VM_RECOVERY_REBOOT 通知が送信されます。

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
<p>VM Alive 後の ESC-NORTHBOUND リカバリ リコールフロー：再起動</p>	<p>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。</p> <pre data-bbox="781 485 1114 531"><type>SERVICE_ALIVE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="781 663 1149 709"><type>VM_RECOVERY_INIT</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="781 842 1203 888"><type>VM_RECOVERY_COMPLETE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="781 999 1203 1066"><type>VM_RECOVERY_COMPLETE</type> <status>FAILURE</status></pre>

シナリオ	通知
<p>VM Alive 後の ESC-NORTHBOUND リカバリ コールフロー：展開解除/ 再展開</p>	<p>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せん。</p> <pre data-bbox="743 485 1073 531"><type>SERVICE_ALIVE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="743 663 1110 709"><type>VM_RECOVERY_INIT</type> <status>SUCCESS</status></pre> <p>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。</p> <p>モニタリングの設定を解除し、VM の展開を解除します。</p> <p>次の NETCONF 通知がトリガーされます。</p> <pre data-bbox="743 951 1187 997"><type>VM_RECOVERY_UNDEPLOYED</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="743 1129 1162 1176"><type>VM_RECOVERY_DEPLOYED</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="743 1308 1162 1354"><type>VM_RECOVERY_COMPLETE</type> <status>SUCCESS</status></pre>

シナリオ	通知
<p>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</p>	<p>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。</p> <pre data-bbox="781 478 1114 527"><type>SERVICE_ALIVE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="781 659 1151 707"><type>VM_RECOVERY_INIT</type> <status>SUCCESS</status></pre> <p>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。</p> <p>モニタリングの設定を解除し、VM の展開を解除します。</p> <p>次の NETCONF 通知がトリガーされます。</p> <pre data-bbox="781 1018 1227 1066"><type>VM_RECOVERY_UNDEPLOYED</type> <status>SUCCESS</status></pre> <p>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。</p> <p>次の NETCONF 通知がトリガーされます。</p> <pre data-bbox="781 1251 1203 1299"><type>VM_RECOVERY_DEPLOYED</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="781 1432 1203 1480"><type>VM_RECOVERY_COMPLETE</type> <status>SUCCESS</status></pre>

シナリオ	通知
<p>VM Alive 前の ESC-NORTHBOUND リカバリ リコールフロー：展開解除/ 再展開</p>	<p>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。</p> <p>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。</p> <p>次の NETCONF 通知がトリガーされます。</p> <pre data-bbox="743 583 1112 636"><type>VM_RECOVERY_INIT</type> <status>SUCCESS</status></pre> <p>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。</p> <p>次の NETCONF 通知がトリガーされます。</p> <pre data-bbox="743 821 1188 873"><type>VM_RECOVERY_UNDEPLOYED</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="743 999 1162 1052"><type>VM_RECOVERY_DEPLOYED</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="743 1178 1162 1230"><type>VM_RECOVERY_COMPLETE</type> <status>SUCCESS</status></pre>

シナリオ	通知
<p>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</p>	<p>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。</p> <pre data-bbox="781 485 1114 531"><type>SERVICE_ALIVE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="781 663 1151 709"><type>VM_RECOVERY_INIT</type> <status>SUCCESS</status></pre> <p>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。</p> <p>モニタリングの設定を解除し、VM の展開を解除します。</p> <p>次の NETCONF 通知がトリガーされます。</p> <pre data-bbox="781 951 1227 997"><type>VM_RECOVERY_UNDEPLOYED</type> <status>SUCCESS</status></pre> <p>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。</p> <p>次の NETCONF 通知がトリガーされます。</p> <pre data-bbox="781 1182 1203 1228"><type>VM_RECOVERY_COMPLETE</type> <status>FAILURE</status></pre>

シナリオ	通知
<p>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</p>	<p>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。</p> <pre data-bbox="743 485 1073 531"><type>SERVICE_ALIVE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="743 663 1110 709"><type>VM_RECOVERY_INIT</type> <status>SUCCESS</status></pre> <p>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。</p> <p>次の NETCONF 通知がトリガーされます。</p> <pre data-bbox="743 932 1187 978"><type>VM_RECOVERY_UNDEPLOYED</type> <status>SUCCESS</status></pre> <p>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。</p> <p>次の NETCONF 通知がトリガーされます。</p> <pre data-bbox="743 1167 1162 1213"><type>VM_RECOVERY_COMPLETE</type> <status>FAILURE</status></pre> <pre data-bbox="743 1262 1073 1308"><type>SERVICE_ALIVE</type> <status>FAILURE</status></pre>

シナリオ	通知
<p>VM Alive 後の ESC-NORTHBOUND リカバリ リコールフロー： VM_RECOVERY_CANCELLED</p>	<p>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。</p> <pre data-bbox="781 457 1114 506"><type>SERVICE_ALIVE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="781 615 1151 663"><type>VM_RECOVERY_INIT</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="781 806 1214 854"><type>VM_RECOVERY_CANCELLED</type> <status>SUCCESS</status></pre>
<p>VM Alive 後の ESC-NORTHBOUND リカバリ リコールフロー：再起動</p>	<p>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。</p> <pre data-bbox="781 1041 1114 1089"><type>SERVICE_ALIVE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="781 1199 1151 1247"><type>VM_RECOVERY_INIT</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="781 1356 1175 1404"><type>VM_RECOVERY_REBOOT</type> <status>SUCCESS</status></pre> <p>VM Alive イベントは、ブート時間内に受信されます。</p> <pre data-bbox="781 1472 1203 1520"><type>VM_RECOVERY_COMPLETE</type> <status>SUCCESS</status></pre>

シナリオ	通知
<p>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</p>	<p>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。</p> <pre data-bbox="743 457 1073 506"><type>SERVICE_ALIVE</type> <status>SUCCESS</status></pre> <p>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。</p> <pre data-bbox="743 615 1110 663"><type>VM_RECOVERY_INIT</type> <status>SUCCESS</status></pre> <p>次に、ESC が再起動通知を送信します。</p> <pre data-bbox="743 730 1138 779"><type>VM_RECOVERY_REBOOT</type> <status>FAILURE</status></pre> <p>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せん。</p> <p>次の NETCONF 通知がトリガーされます。</p> <pre data-bbox="743 940 1162 989"><type>VM_RECOVERY_COMPLETE</type> <status>FAILURE</status></pre>



第 VI 部

ESC ポータル

- [使用する前に \(307 ページ\)](#)
- [ESC ポータルを使用したリソースの管理 \(315 ページ\)](#)
- [ESC ポータルを使用した VNF の展開 \(321 ページ\)](#)
- [ESC ポータルを使用した VNF および VM の操作 \(327 ページ\)](#)
- [ポータルを使用した VNF および VM のリカバリ \(329 ページ\)](#)
- [ESC システムレベルの設定 \(331 ページ\)](#)



第 38 章

使用する前に

- [ESC ポータルへのログイン](#) (307 ページ)
- [ESC パスワードの変更](#) (308 ページ)
- [ESC ポータルダッシュボード](#) (309 ページ)

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 8443 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 [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 で実行中は、非セキュアな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。この手順を飛ばすことはできません。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。初めてパスワードを変更した後は、このセクションで説明す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使用している場合、または同じユーザが 2 台以上のコンピュータでログオンしている場合、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、新しいパスワードの再入力が必要になります。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 [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 27: ポータルの詳細

タスク	Navigate	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	Navigate	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。
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。

タスク	Navigate	説明
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 でのみ使用できません。 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。
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。

タスク	Navigate	説明
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 (Infrastructure)] > [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 (Infrastructure)] > [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。
VNF を展開解除する	<ul style="list-style-type: none"> • [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。

[Notifications]

[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 39 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 \(315 ページ\)](#)
- [ESC ポータルを使用した OpenStack リソースの管理 \(316 ページ\)](#)
- [ESC ポータルを使用した VMware vCenter リソースの管理 \(318 ページ\)](#)

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 \(50 ページ\)](#) 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 40 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) (321 ページ)
- ESC ポータルを使用した VMware vCenter での VNF の展開 (322 ページ)
- 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 (325 ページ)

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 \(177 ページ\)](#) 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 \(177 ページ\)](#) 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 \(279 ページ\)](#) 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 \(253 ページ\)](#) 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 41 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 \(327 ページ\)](#)
- [VM 操作の実行 \(328 ページ\)](#)

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 (Active)
VNF の開始	停止 (Stopped)
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（Error）

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（Active）
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（Error）



第 42 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 \(329 ページ\)](#)

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 \(281ページ\)](#) 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 43 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 \(331 ページ\)](#)

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 (333 ページ)

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 28: 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： <pre><flavor> <name>FLAVOR_2_4096_10000</name> <vcpus>2</vcpus> <memory_mb>4096</memory_mb> <root_disk_mb>10000</root_disk_mb> </flavor></pre>

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	<p>Extension:</p> <pre><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></pre> <p>展開例 :</p> <pre><volumes> <volume> <valid>1</valid> <sizeunit>GiB</sizeunit> <size>20</size> <bus>virtio</bus> </volume> </volumes></pre>
Deployment/ vm group / extentions/ image	<p>Extension:</p> <pre><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></pre>

リソース	内線番号 導入例
Deployment/ vm group / extentions/ vnc	Extension: <pre><extension> <name>vnc</name> <properties> <property> <name>vnc_password</name> <value>*****</value> </property> </properties> </extension></pre>
Deployment/ vm group / extentions/ vnf_mgmt_ip	Extension: <pre><extension> <name>vnf_mgmt_ip</name> <properties> <property> <name>nicid</name> <value>0</value> </property> </properties> </extension></pre>
Deployment/ vm group / serial_ports	Extension: <pre><extension> <name>serial_ports</name> <containers> <container> <name>0</name> <properties> <property> <name>serial_type</name> <value>console</value> </property> </properties> </container> </containers> </extension></pre>

リソース	内線番号 導入例
Deployment/ vm group / interfaces /	<p>Extension:</p> <pre> <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> <property> <name>netmask</name> <value>xx.xx.xx.xx</value> </property> <property> <name>gateway</name> <value>xx.xx.xx.xx</value> </property> <property> <name>ip_prefix</name> <value>xx</value> </property> </properties> </container> </containers> </extension> </pre> <p>展開例 :</p> <pre> <interfaces> <interface> <nicid>0</nicid> <type>virtual</type> <model>virtio</model> <network>xyz</network> <ip_address>xx.xx.xx.xx</ip_address> </interface> </interfaces> </pre>
Deployment/ vm group / <vim_vm_name>	<p>拡張機能 : なし。</p> <p>展開例 : <vim_vm_name>my-custom-csr</vim_vm_name></p>

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： <pre><config_type>CONFIG_DATA_OPTIONS</config_type> <config_options> <options> <option> <name>day0-volume-id</name> <value>cidata</value> </option> </options> </config_options></pre>

