Cisco Elastic Services Controller 5.2 ETSI NFV MANO 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 NONINFRINGEMENT 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

このマニュアルについて

このガイドは、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、スケーリングなどのタスク実行を支援するためのものです。

- 対象読者 （vii ページ）

対象読者

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

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

用語および定義

次の表で、このガイドで使用されている用語を定義します。

表 *1* : 用語および定義

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

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

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

関連資料

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

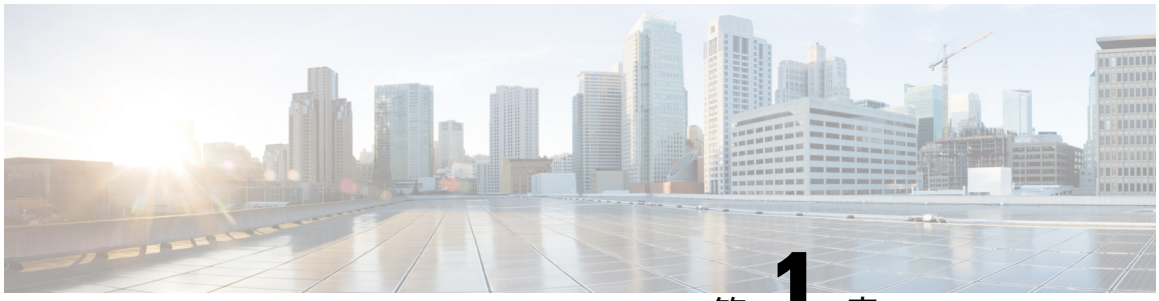
ガイド	このガイドに記載されている情報
Cisco Elastic Services Controller Release Notes	新機能とバグ、既知の問題が記載されています。
Cisco Elastic Services Controller Install and Upgrade Guide	新規インストールとアップグレードのシナリオ、インストール前後のタスク、ESC 高可用性（HA）展開の手順が記載されています。
Cisco Elastic Services Controller User Guide	VNF のライフサイクル管理操作、モニタリング、修復、スケーリングが記載されていま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

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

ドキュメントの入手方法

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

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



第 1 章

ETSI NFV MANO Northbound API の概要

ETSI NFV MANO Northbound API の概要

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。API は JavaScript オブジェクト表記（JSON）ペイロードを含む HTTP メッセージを受け取り、返します。API には、ESC コアデータモデルから取り除かれた ETSI MANO 仕様に基づいて設計された独自のデータモデルが含まれています。

REST/NETCONF API を使用した VNF ライフサイクル管理操作の詳細については、『Cisco Elastic Services Controller User Guide』を参照してください。

表 *2 : ETSI MANO* の仕様

仕様	バージョン サポート	説明
SOL001	v2.5.1	VNF 記述子のフォーマットと構造
SOL002	v2.5.1	Ve-Vnfm 参照ポイント上のすべてのインタラクションを定義します
SOL003	v2.4.1	Or-Vnfm 参照ポイント上のすべてのインタラクションを定義します

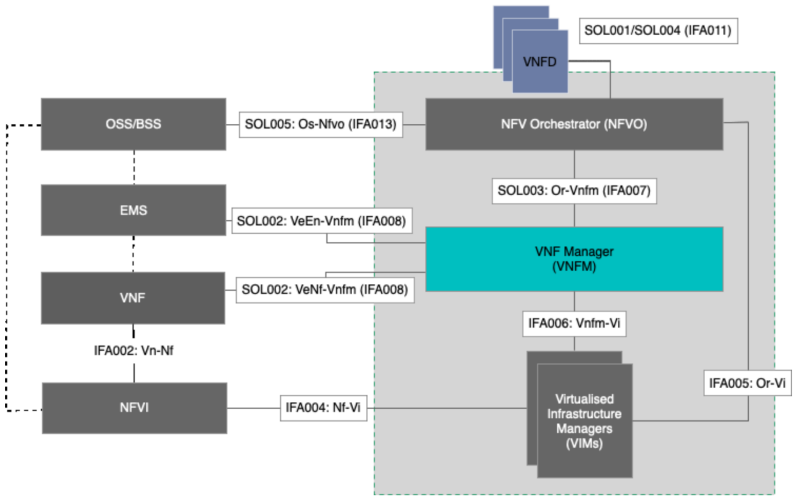
（注） ユーザガイドの ETSI 固有のセクションで使用される用語は、ETSI ドキュメントで定義される ETSI MANO 標準に準拠しています。詳細については、ETSI Webサイトを参照してください。

ETSI NFV MANO Northbound API の概要

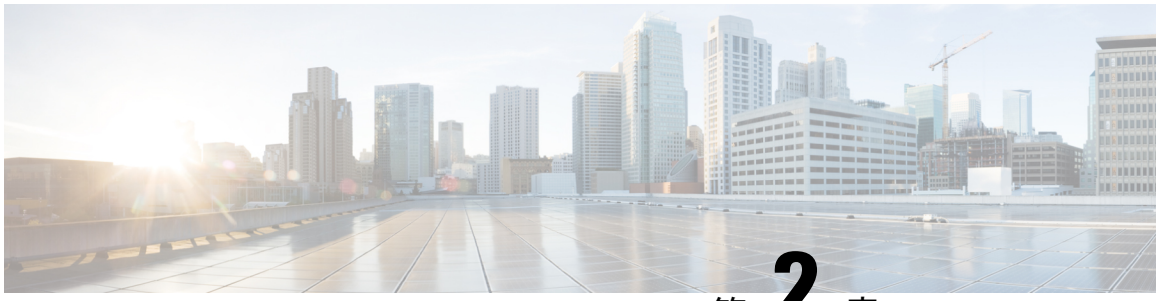
ETSI NFV MANO 標準の現在の実装は、それぞれ NFVO と VNFM、EM と VNFM 間のインターフェイスである Or-Vnfm と Ve-Vnfm の参照ポイントで構成されています。どちらも、ETSI 準拠の CSAR パッケージのオンボーディング、仮想化リソースの管理、および VNF ライフサイクル管理（LCM）操作が可能です。

Or-Vnfm および Ve-Vnfm の参照ポイントの詳細については、ETSI Web サイトの *ETSI* グループ仕様ドキュメントを参照してください。次の図は、すべての参照ポイントの NFV MANO アーキテクチャを示しています。

図 *1* : 参照ポイントと *NFV MANO* アーキテクチャ



リソースの管理については、ETSI API のリソース定義 （3 ページ）を参照してください。



第 2 章

リソースの管理

リソースの管理

ETSI API のリソース定義

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。

ETSI MANO の場合、これらのリソース定義は、VNF パッケージのオンボーディング時またはテナントのオンボーディング時に NFVO によって作成され、ESC への要求の VIM ID によって表されます。

NETCONF または REST API を使用したリソースの管理については、『Cisco Elastic Services Controller User Guide』の「Managing Resources 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。

次の表に、VNF のインスタンス化の前に使用可能にする必要がある VIM のリソース定義を示します。

表 *3 : VIM* でのリソース定義

リソースの定義	OpenStack
テナント	アウトオブバンドテナント NETCONF API、REST API、または ESC ポータルを使用してテナントを作成できます。テナントを VIM で直接作成することもできます。その後、テナントは vimConnectionInfo データ構造内で参照されます。詳細については、VIM コネクタの概要 （13 ページ）を参照してください。
画像	アウトオブバンドイメージ NFVO は VNF パッケージをオンボードし、VNF パッケージに含まれるイメージを抽出して VIM にオンボードします。VNFD はイメージファイルを参照しますが、イメージファイルのサイズを理由に、展開時にイメージをオンボーディングする代わりに、Grant の vimAssets が使用するイメージを指定します。
フレーバ	アウトオブバンドフレーバ VNF パッケージのオンボーディング中、NFVO は VNFD 内の各 cisco.nodes.nfv.Vdu.Compute node の機能を調べて、作成するフレーバを決定します。これはインスタンス化された後で使用できます。またはオプションで、インスタンス化したときに追加パラメータとして指定された VIM フレーバによってオーバーライドされます。 （注） ETSI 展開フレーバは、OpenStack コンピューティングフレーバとは異なる概念です。詳細については、このガイドの「用語と定義」を参照してください。
音量（Volumes）	ESC は、シスコの拡張としてアウトオブバンドボリュームをサポートします。
外部ネットワーク（仮想リンク）	外部接続ポイントが接続するインスタンス化ペイロードで指定された外部ネットワーク。

リソースの定義	OpenStack
外部管理の内部仮想リンク	エフェメラルネットワークを作成する代わりに、内部仮想リンクがバインドされるインスタンス化ペイロードで指定された外部ネットワーク。
サブネットワーク	アウトオブバンドサブネット

ETSI API を使用した VNF パッケージのオンボーディングとライフサイクル操作の詳細については、VNF ライフサイクルの管理 （25 ページ）を参照してください。

リソース定義の更新

このセクションでは、ETSI API リソース定義の更新について詳しく説明します。

VNF フレーバの更新

次の TOSCA パラメータを使用して、1 つの VNFD の代替 VNF ノードと展開フレーバを定義できます。

- **Import statements**：import statement により、実行時に動的に指定できる入力値に基づいて、1 つの親の VNFD yaml ファイルに他のファイルを条件付きで含めることができます。
- **Substitution mappings**：substitution mapping は *tosca.nodes.nfv.VNF* から派生したノードタイプにのみ適用されます。他のノードタイプの値（接続ポイント、仮想リンクなど）を置き換えることはできません。

例 1：

この例では、yaml ファイルに 3 つのインポートファイルが含まれています。

3 つのファイルはすべて、インポートする親ファイルと同じ場所にある VNFD ZIP アーカイブファイルに存在する必要があります。

要件と機能は、派生した *tosca.nodes.nfv.VNF* ノードで定義されません。これらは、この VNFD を使用してインスタンス化された VNF の特性を定義するために必須です。これらはインポートされたファイル内で定義されます。

```
tosca_definitions_version: tosca_simple_yaml_1_2
description: Substitution Mapping Example

imports:
- df_default.yaml
- df_silver.yaml
- df_gold.yaml

. . .

node_types:
my-vnf:
derived_from: tosca.nodes.nfv.VNF
```

```
. . .

topology_template:

. . .

########################
# Substitution Mapping #
########################
substitution_mappings:
node_type: my-vnf
requirements:
# None

node_templates:

vnf:
type: my-vnf
properties:
descriptor_id: 8717E6CC-3D62-486D-8613-F933DE1FB3A0

. . .

flavour_id: default
flavour_description: Default VNF Deployment Flavour
```

例 2：

VNF がインスタンス化されると、必要なフレーバがインスタンス化要求で VNFM に送信されます。TOSCA パーサーが、フレーバおよび VNF ノード名と、定義された substitution mappings との照合を試みます。これらは、VNFD 自体内でインポートまたは定義できます。たとえば、*df_silver.yaml* には次の内容が含まれています。

tosca_definitions_version: tosca_simple_yaml_1_2

説明：Silver 展開フレーバ

インポート：

```
topology_template:
substitution_mappings:
node_type: my-vnf
properties:
flavour_id: silver
flavour_description: Silver VNF Deployment Flavour
requirements:
- virtual_link: [ vm1_nic1, virtual_link ]
```

silver は、インスタンス化要求ペイロードで渡される flavourId です。上記の親の *yaml* には、silver プロファイルからの要件で更新された空の要件セクションがあり、既存の *flavour_id* プロパティと *flavour_description* プロパティも更新されます。

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: Deployment Flavour SILVER
topology_template:
  substitution_mappings:
    node_type: tosca.nodes.nfv.VNF.CiscoESC
    requirements:
      virtual_link: [ anECP, external_virtual_link ]
    capabilities:
        deployment_flavour:
```

```
          properties:
            flavour_id: silver
            description: 'SILVER Deployment Flavour'
            vdu_profile:
              vdu_node_1:
                min_number_of_instances: 2
                max_number_of_instances: 2
            instantiation_levels:
              default:
                description: 'Default Instantiation Level'
                vdu_levels:
                  vdu_node_1:
                    number_of_instances: 1
                scale_info:
                  default_scaling_aspect:
                    scale_level: 2
              silver_level:
                description: 'SILVER Instantiation Level'
                vdu_levels:
                  vdu_node_1:
                    number_of_instances: 2
                scale_info:
                  default_scaling_aspect:
                    scale_level: 2
            default_instantiation_level_id: default
            vnf_lcm_operations_configuration: {}
            scaling_aspect:
              - default_scaling_aspect
            cisco_esc_properties:

    description: "SILVER: This is substituted if not already defined"
```

ESC は POST 要求を送信して VNF フレーバを更新します。

メソッドタイプ：

POST

VNFM エンドポイント：

```
/vnflcm/v1/vnfinstances/{vnfInstanceId}/change_flavour
```

外部 VNF 接続の更新

既存の展開で外部 VNF 接続を更新できます。API は次の変更をサポートします。

- 既存の外部仮想リンクへの既存の接続ポイント（CP）を切断し、別の仮想リンクに接続します。
- アドレスの変更を含め、既存の外部 CP の接続パラメータを変更します。

ESC は VNF 外部接続を更新するための POST 要求を送信します。

メソッドタイプ

POST

VNFM エンドポイント

```
/vnflcm/v1/vnfinstances/{vnfInstanceId}/change_ext_conn
```

要求ペイロード（データ構造 = ChangeExtVnfConnectivityRequest）

```
{
  "extVirtualLinks": [
    {
      "id": "extVL-98345443-7797-4c6d-a0ed-e18771dacf1c",
      "resourceId": "node_1_ecp",
      "extCps": [
        {
          "cpdId": "node_1_ecp",
          "cpConfig": [
            {
              "cpProtocolData": [
                {
                  "layerProtocol": "IP_OVER_ETHERNET",
                  "ipOverEthernet": {
                    "ipAddresses": [
                      {
                        "type": "IPV4",
                        "numDynamicAddresses": 2,
                        "subnetId": "esc-subnet"
                      }
                    ]
                  }
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

（注） extVirtualLinks の ID（上記の例では *extVL-98345443-7797-4c6d-a0ed-e18771dacf1c*）は、vnfInstance の InstantiatedVnfInof にも存在する必要があります。

マージポリシー

置換により、新しい値が VNFD にマージされます。

1. name=joe などの通常のスカラプロパティの場合、値は VNFD で置き換えられます。
2. [list, of, strings] などの配列はマージされます。新しい値が存在しない場合は、配列に追加されます。
3. キーが別のキーの下にインデントされているなどのオブジェクトは置き換えられます。一致した置換の configurable_properties オブジェクトは、VNFD で定義されたものを上書きします。

パーサーの動作

- substitution mappings が作成された後、パーサーは提供された *additionalParams* を事前入力しようとします。入力パラメータがテンプレートのパラメータと一致しない場合、コマンドは失敗します。

VNF ライフサイクル操作の詳細については、VNF ライフサイクルの管理 （25 ページ）を参照してください。

OAuth（Open Authorization）2.0 認証

ETSI NFV MANO は、SOL003 Or-Vnfm リファレンスポイントの OAuth 2.0 認証をサポートします。NFVO は、認証用のクライアント ID やクライアントシークレットなどのクライアントログイン情報を提供する ESC にトークン要求を行います。次に、ESC は要求を確認し、アクセストークンを返します。

NFVO は、プライマリ認証として clientId と secret を提供する POST 要求を行います。

メソッドタイプ

POST

URL

```
{apiRoot}/oauth2/token
```

ヘッダー

```
Authorization: Basic {base 64 encoded CLIENT_ID:CLIENT_SECRET}
Accept: application/json
Content-Type: application/x-www-form-urlencoded
```

本文

```
grant_type=client_credentials
```

ESC は応答でアクセストークンを返します。

例：

```
{
    "access_token":
"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJjaHJpcyIsImlzcyI6IkVUU0ktVk5GTSIsImlhdCI6MTU1ODYwMzk2NiwiZXhwIjoxNTU4NjA0NTY2f


Q.lAtre7vdCKJjgzNs7p9P3NS2qMcXegC-oWXmy5Kakn0AL95gLWF6liOqPViMZNnWZLOsG5r1kPnGoBWnN0tgIw",

    "token_type": "bearer",
    "expires_in": 600
}
```

次に、アクセストークンは or_vnfm エンドポイントにアクセスするために使用されます。

例：

方法

GET

URL

```
{apiRoot}/vnflcm/v1/subscriptions
```

ヘッダー

```
Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJjaHJpcyIsImlzcyI6IkVUU0k
tVk5GTSIsImlhdCI6MTU1ODYwMzk2NiwiZXhwIjoxNTU4NjA0NTY2fQ.lAtre7vdCKJjgzNs
7p9P3NS2qMcXegC-oWXmy5Kakn0AL95gLWF6liOqPViMZNnWZLOsG5r1kPnGoBWnN0tgIw
```

（注） ETSI サービスが再起動されると、既存のトークンは無効になります。

OAuth プロパティファイルへのアクセスと更新

ESC は、*etsi-production.properties* ファイルと同じ場所にある新しい *etsi-production.yaml* プロパティファイルにクライアント ID とシークレットを保存します。クライアント ID とシークレット値を管理するために、新しい escadm etsi コマンドを使用できます。クライアントシークレットは、既存の REST ユーザ名と同じ方法で暗号化されます。

クライアント ID を追加または更新するには

```
sudo escadm etsi oauth2_clients --set <CLIENT_ID>:<CLIENT_SECRET>
```

クライアント ID を削除するには

```
sudo escadm etsi oauth2_clients --remove <CLIENT_ID>
```

（注） OAuth 2.0 値を更新した後、ETSI サービスを再起動します。

その他のプロパティの情報については、ETSI 製品のプロパティ （99 ページ）を参照してください。

ETSI から NFVO への OAuth コール

ESCは、ETSI から NFVO への OAUTH 2.0 コールをサポートします。

次のプロパティが etsi-product.properties ファイルに追加されます。

```
nfvo.clientID=<YourClientID>
nfvo.clientSecret=<YourClientSecret>
nfvo.tokenEndpoint=<Your NFVO Token Endpoint>
nfvo.authenticationType=OAUTH2
```

クライアント ID、ClientSecret、および TokenEndpoint は、OAUTH 2.0 サーバのものと一致する必要があります。認証タイプは、ESC から NFVO への発信コールの認証を決定します。認証タイプは、BASIC または OAUTH2 のいずれかである必要があります。

NFVO からのトークンは、プロパティファイルのトークンエンドポイントに保存されます。

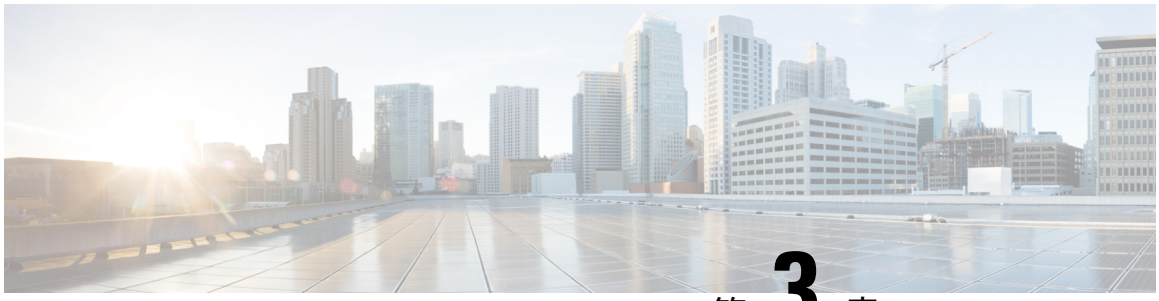
NFVO がコール要求を送信すると、ETSI はトークンエンドポイントに保存されているトークンをチェックします。トークンの有効期限が切れていない場合、ETSI は古いトークンを要求のヘッダーに追加し、コールを実行します。トークンの実行に失敗した場合は、新しいトークンが必要です。

トークンエンドポイントに対してトークンがない場合は、コールを実行するための新しいトークンが必要です。

OAuth 2.0 通知およびサブスクリプション

通知で OAuth 2.0 認証を有効にするには、サブスクリプション ペイロードに以下を追加する必要があります。

```
{
  "authentication": {
    "authType": [
      "OAUTH2_CLIENT_CREDENTIALS"
    ],
    "paramsOauth2ClientCredentials": {
      "clientId": <client_id>,
      "clientPassword": <client_secret>,
      "tokenEndpoint": <token_endpoint>
    }
  }
}
```



第 3 章

VIM コネクタの管理

VIM コネクタの概要

ETSI API は、LCM 操作の処理中に VIM コネクタを作成するか、既存のコネクタを使用します。

NFVO からの 付与応答または LCM 操作要求は、新しい VimConnectionInfo を VnfInstance に提供します。LCM 操作の処理中に、ETSI は新しい *VimConnectionInfo* を ESC の VIM コネクタと同期します。

VnfInstance に同じ ID を持つ既存の VimConnectionInfo がない場合、*VimConnectionInfo* は新規です。 LCM 要求の一部として任意の VnfInstance に保存されている既存の *VimConnectionInfo* ID と一致する、指定された *VimConnectionInfo* は、既存のコネクタを使用し、新しい要求で送信された変更を無視します。

ESC は、一致する VIM コネクタが使用できない場合にのみ、新しい VIM コネクタを作成します。

ETSI API では、VNF 情報の変更操作により、既存の *VimConnectionInfo*、および関連付けられた VIM コネクタのみを更新できます。

NFVO からの付与は、各リソースの vimConnectionId を指定します。この値は、各リソースのロケータを作成するための VimConnectionInfo、および関連付けられた VIM コネクタを識別します。VIM 固有の *VimConnectionInfo.accessInfo* プロパティは、ロケータの追加プロパティとして設定されます。

OpenStack の VimConnectionInfo：

```
{
  "id": "435456",
  "vimType": "OPENSTACK_V3",
  "interfaceInfo": {
```

```
    "endpoint": "https://10.18.54.42:13001/v3/"
  },
  "accessInfo": {
    "username": "admin",
    "password": "bmkQJtyDrbPFnJT8ENdZw2Maw",
    "project": "cbamnso",
    "projectDomain": "Default",
    "userDomain": "Default",
    "vim_project": "cbamnso"
  }
}
```

vCloud Director の VimConnectionInfo：

```
{
  "id": "435456",
  "vimType": "VMWARE_VCD",
  "interfaceInfo": {
    "endpoint": "https://10.85.103.150"
  },
  "accessInfo": {
    "username": "admin@cisco",
    "password": "bmkQJtyDrbPFnJT8ENdZw2Maw",
    "vim_project": "cbamnso",
    "vim_vdc": "vdc1"
  }
}
```

新しい VIM コネクタの作成

ETSI LCM の操作中、ESC は各 *VimConnectionInfo* を既存の VIM コネクタレコードと照合します。既存の VIM コネクタが使用できない場合、ESC は新しい VIM コネクタを作成します。

VimConnectionInfo.vimId が指定されている場合、この値は新しい VIM コネクタの ID として使用されます。*VimConnectionInfo.vimId* が指定されていない場合、新しい VIM コネクタの ID が生成され、この値も *VimConnectionInfo.vimId* として設定されます。

既存の VIM コネクタを使用するには、既存の VIM コネクタの使用 （14 ページ）を参照してください。

既存の VIM コネクタの使用

ETSI LCM 操作中、ESC は、VnfInstance に保存された、一致する ID を持つ既存の vimConnectionInfo を確認します。

既存の VIM コネクタは次によって検出されます。

- *VimConnectionInfo.vimId*（指定されている場合）を VIM コネクタの ID と照合します。
- *VimConnectionInfo* の VIM 固有のプロパティを VIM コネクタと照合します。
  - OpenStack
    - vimType

- interfaceInfo.endpoint
- accessInfo.project

- vCloud Director
  - vimType
  - interfaceInfo.endpoint

一致する VIM コネクタが見つかり、*VimConnectionInfo.vimId* が設定されていない場合、*VimConnectionInfo.vimId* が VIM コネクタの ID に設定されます。

NFVO が VimConnectionInfo に accessInfo を提供して、接続プロパティの一部を指定する場合、次のキーを使用して VIM コネクタを設定します。

Openstack

- username
- パスワード
- プロジェクト
- projectDomain
- userDomain
- vim_project

vCloud Director

- username
- パスワード
- vim_project
- vim_vdc

ETSI の仕様では、accessInfo 属性の一部として使用するキーを指定していません。統合しやすくするため、NFVO が異なるキーを使用する場合、プロパティファイルを使用して、サードパーティキーから ESC が理解できるキーへのマッピングを指定できます。

```
mapping.vimConnectionInfo.accessInfo.username
mapping.vimConnectionInfo.accessInfo.password
mapping.vimConnectionInfo.accessInfo.project
mapping.vimConnectionInfo.accessInfo.projectDomain
mapping.vimConnectionInfo.accessInfo.userDomain
mapping.vimConnectionInfo.accessInfo.vim_project
mapping.vimConnectionInfo.accessInfo.vim_vdc
```

新しい VIM コネクタを作成するには、新しい VIM コネクタの作成 （14 ページ）を参照してください。

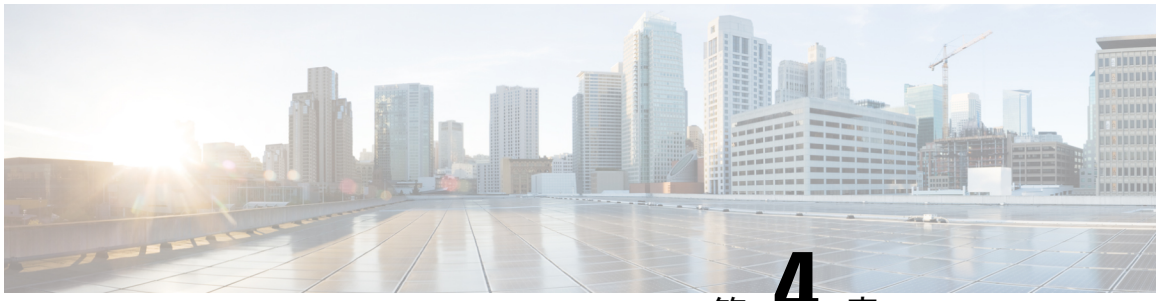
VIM コネクタの更新

ETSI API は、仮想ネットワーク機能の変更 （42 ページ）操作を介して、既存の VimConnectionInfo と関連付けられた VIM コネクタを更新します。変更要求ペイロードの *VimConnectionInfo* は、VnfInstance に保存されている既存の *VimConnectionInfo* と比較されます。

一致する ID を持つ VnfInstance に保存されている既存の VimConnectionInfo が見つからない場合は、この VimConnectionInfo が VnfInstance に追加されます。

一致する ID を持つ VnfInstance に保存されている既存の *VimConnectionInfo* が見つかった場合は、その VimConnectionInfo が更新されます。VimConnectionInfo が変更され、それに関連付けられた VIM コネクタがある場合、その VIM コネクタも更新されます。

新しい VIM コネクタを作成するには、新しい VIM コネクタの作成 （14 ページ）を参照してください。



第 4 章

仮想ネットワーク機能記述子について

仮想ネットワーク機能記述子の概要

ESC は、VNF プロパティを記述する TOSCA ベースの仮想ネットワーク機能記述子（VNFD）をサポートします。VNFD は、ETSI で指定された *GS NFV-SOL 001* 仕様および標準に準拠しています。

VNFD ファイルには、VNF のインスタンス化パラメータと操作の動作が記述されています。これには、KPI、およびオンボーディングと VNF のライフサイクル管理のプロセスで使用できるその他の主要な要件が含まれています。

VNF ライフサイクル操作については、VNF ライフサイクル操作 （26 ページ）を参照してください。

仮想ネットワーク機能記述子への拡張定義

ESC はシスコが定義した VNFD の拡張機能を実装しており、ESC ではサポートされているものの ETSI 標準にはない、より高度な概念を公開しています。これらの拡張は、オーバーライドされるデータ、ノード、およびインターフェイスタイプを記述するために、シスコのタイプ定義に厳密に入力されます。

VNF の設定可能なプロパティ

VNF ノードタイプは、VNF ごとに常にスタマイズされます。シスコの拡張機能は、ESC が緩和アクションを考慮する前に、VNF が復旧するまでの復旧ポリシーと待機時間を指定する機能を提供します。

次に例を示します。

```
vnf:
      type: cisco.VPC.1_0.1_0
      properties:
        descriptor_id: b98450dd-f532-4a42-8419-e3dc04327318
```

```
    descriptor_version: '3.8'
    provider: cisco
    product_name: VPC
    software_version: 1.0
   product_info_name: 'Virtual Packet Core (VPC); 32 vCPUs, 64Gb RAM, 66Gb vStorage'

    vnfm_info:
      - '9:Cisco Elastic Services Controller:v04.04.01'
    configurable_properties:
      is_autoscale_enabled: false
      is_autoheal_enabled: false
    lcm_operations_configuration:
      heal:
        recovery_action: REBOOT_THEN_REDEPLOY
        recovery_wait_time: 0
    flavour_id: default
    flavour_description: 'Default VNF Deployment Flavour'
```

コンピューティング

Cisco コンピューティングノードでは、拡張 ETSI データモデルを介して多くの ESC 機能を公開できます。これには、次の事項が含まれます。

- VIM 上の VNFC の自動生成された名前のオーバーライド
- VIM フレーバ（VNFC に指定された ETS I 機能をオーバーライド）。
- このタイマーが期限切れになるまでアクションが実行されるのを防ぐため、ESCに予想されるブートアップ時間を指定。
- VNFC を展開後、実行/保存する Day-0 設定ブロックを提供。
- モニタリングエージェントを設定するための KPI パラメータと関連ルールの指定。
- VM グループ内の配置ルール。

次に例を示します。

```
  vdu1:
    type: cisco.nodes.nfv.Vdu.Compute
    properties:
      name: Example VDU1
      description: Example VDU
      boot_order:
        - boot1-volume
      configurable_properties:
        additional_vnfc_configurable_properties:
          vim_flavor: Automation-Cirros-Flavor
          bootup_time: 1800
      name_override: my-vdu-1
      vdu_profile:
        min_number_of_instances: 1
        max_number_of_instances: 1
        static_ip_address_pool:
          network: esc-net
          ip_address_range:
            start: { get_input: VDU1_NETWORK_START }
            end: { get_input: VDU1_NETWORK_END }
          ip_addresses: { get_input: VDU1_SCALE_IP_LIST }
      kpi_data:
        VM_ALIVE-1:
         event_name: 'VM_ALIVE-1'
```

```
        metric_value: 1
        metric_cond: 'GT'
        metric_type: 'UINT32'
        metric_occurrences_true: 1
        metric_occurrences_false: 30
        metric_collector:
          type: 'ICMPPing'
          nicid: 1
          poll_frequency: 10
          polling_unit: 'seconds'
          continuous_alarm: false
    admin_rules:
      VM_ALIVE-1:
        event_name: 'VM_ALIVE-1'
        action:
          - 'ALWAYS log'
          - 'FALSE recover autohealing'
          - 'TRUE esc_vm_alive_notification'
    placement_type: zone
    placement_target: nova
    placement_enforcement: strict
    vendor_section:
      cisco_esc:
        config_data:
          example.txt:
            file: ../Files/Scripts/example.txt
            variables:
              DOMAIN_NAME: { get_input: DOMAIN_NAME }
              NAME_SERVER: { get_input: NAME_SERVER }
              VIP_ADDR: { get_input: VIP_ADDR }
              VIP_PREFIX: { get_input: VIP_PREFIX }
  capabilities:
    virtual_compute:
      properties:
        virtual_cpu:
          num_virtual_cpu: 8
        virtual_memory:
          virtual_mem_size: 16
  requirements:
    - virtual_storage:  cdr1-volume
    - virtual_storage:  boot1-volume
```

（注） 多数の入力パラメータを指定できるため、複数の展開で1つのテンプレートを使用できます。

接続ポイント

VduCp ノードタイプへのシスコの拡張機能により、主に IP アドレッシング機能とインターフェイスへのアクセシビリティが向上します。接続ポイントに追加された機能は次のとおりです。

- VIM 上のポートの自動生成された名前のオーバーライド
- 静的 IP アドレス（およびスケーリング用のプール）
- ポートが管理ポート（モニタリングに使用する）かどうかの識別
- 許可されるアドレスペア
- 特定のネットワークカードタイプとインターフェイスタイプ（SR-IOV など）のサポート

- ポート バインディング プロファイルのサポート
- ポートセキュリティが有効かどうか

次に例を示します。

```
vdu1_nic0:
      type: cisco.nodes.nfv.VduCp
      properties:
        layer_protocols: [ ipv6 ]
        protocol:
          - associated_layer_protocol: ipv6
        trunk_mode: false
        order: 0
        nw_card_model: virtio
        iface_type: direct
        management: true
        name_override: my-vdu1-nic0
        ip_subnet:
          - ip_address: { get_input: VDU1_NIC0_IP }
        allowed_address_pairs:
          - ip_address: { get_input: VDU1_NIC0_AADR_PAIRS }
        port_security_enabled: false
        binding_profile:
        trusted: true
        requirements:
        - virtual_binding: vdu1
```

展開ごとにこれらのプロパティを制御する必要がある場合は、着信要求で additionalParams として指定できる VNFD の入力で、ハードコードされた値を置き換えます。

（注） ポートバインディングプロファイルは、OpenStack の Pike 以上のバージョンで使用できます。

音量

ESC は、シスコの拡張としてアウトオブバンドボリュームをサポートします。これにより、永続的なボリューム UUID の仕様を、cisco.nodes.nfv.Vdu.VirtualBlockStorage ノードに対する resourceId プロパティとして、VNFD で定義されたエフェメラルボリュームの代わりに使用できます。余分なプロパティを追加する代わりに、ESC は VIM からの UUID で識別することによって、VNFD で指定されたボリュームをオーバーライドし、独自の永続的な（展開されたアウトオブバンド）ストレージを指定することを許可します。

次に例を示します。

```
    boot1-volume:
      type: cisco.nodes.nfv.Vdu.VirtualBlockStorage
      properties:
        resource_id: { get_input: VDU1_BOOT_VOL_UUID }
        virtual_block_storage_data:
          size_of_storage: 4GB
          vdu_storage_requirements:
            vol_id: 1
            bus: ide
            type: LUKS
        sw_image_data:
          name: 'Automation_Cirros'
          version: '1.0'
```

```
        checksum: 9af30fce37a4c5c831e095745744d6d2
        container_format: bare
        disk_format: qcow2
        min_disk: 2 GB
        size: 2 GB
    artifacts:
      sw_image:
        type: tosca.artifacts.nfv.SwImage
        file: ../Files/Images/Automation-Cirros.qcow2
```

エフェメラルリソースの代わりにアウトオブバンドリソースを指定するため、ESCではインスタンス化の最中に、着信要求を使用してVNFD内のタグを照合できます。新しいデータ構造が既存の InstantiateVnfRequest に追加されます。

次の例を参考にしてください。

```
{
  "flavourId": "default",
  "instantiationLevelId": "default",
  "extVirtualLinks": [{}],
  "extManagedVirtualLinks": [{}],
  "extManagedVolumes": [
        {
            "virtualStorageDescId": "cf-cdr1-volume",
            "resourceId": "vol123"
        },
        {
            "virtualStorageDescId": "cf-boot1-volume",
            "resourceId": "vol456"
        }
    ],
  ...
}
```

セキュリティグループルールの作成

上記のボリュームの処理に従って、ESC は VNFD で設定する代わりに、アウトオブバンド セキュリティ グループを指定する機能を提供します。これは、標準のドキュメントでセキュリティグループを説明するために使用される動詞が、非常に複雑な設定に対しては単純すぎるためです。

次に例を示します。

```
    - NETWORK_ORCH_SEC_GRP_1:
        type: cisco.policies.nfv.SecurityGroupRule
        group_name: { get_input: VIM_NETWORK_ORCH_SEC_GRP_1 }
        targets: [ vdu1_nic0 ]
```

カスタム VM 名

シスコの拡張機能では、追加パラメータを使用して、展開時の VNFC（VM）名をカスタマイズできます。ESC ETSI には、VM 名をカスタマイズするための追加パラメータが含まれています。

VIMでVM名を設定するには、最初にデータタイプを定義してから、コンピューティングノードの Cisco ノードタイプを拡張する必要があります。

```
tosca_definitions_version: tosca_simple_yaml_1_2
```

```
data_types:
  cisco.datatypes.nfv.VnfcAdditionalConfigurableProperties:
    derived_from: tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties
    properties:
      vim_flavor:
        type: string
        required: true
      bootup_time:
        type: integer
        required: true
      vm_name_override:
        type: string
        required: false
```

これらの定義により、VNFD node_templates は入力を使用してコンピューティングノードにマッピングできます。

```
topology_template:
  inputs:
    ...

  node_templates:

#####################
# VDU configuration #
#####################
    c1:
      type: cisco.nodes.nfv.Vdu.Compute
      properties:
        name: control-function 1
        description: Vdu1 of an active:standby (1:1) redundant pair of CF VMs
        ...
        configurable_properties:
          additional_vnfc_configurable_properties:
            vim_flavor: { get_input: CF_FLAVOR }
            bootup_time: { get_input: BOOTUP_TIME_CF }
            vm_name_override: { get_input: VIM_C1_VM_NAME ] }
        ...
      capabilities:
        virtual_compute:
          properties:
            virtual_cpu:
              num_virtual_cpu: 8
            virtual_memory:
              virtual_mem_size: 16 GiB
      requirements:
        - virtual_storage:  cf-cdr1-volume
        - virtual_storage:  cf-boot1-volume
```

コンピューティングノードの設定可能なプロパティで、*vm_name_override* を指定します。*vm_name_override* 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。

ESCは、VNFC を表すコンピューティングノードに与えられたラベルと一致する vduId によって識別される VNFC の *VnfInstance.instantiatedVnfInfo.vnfcResourceInfo.metadata.vim_vm_name* に VNFC 固有の値を保存します。

ライフサイクル管理操作の詳細については、VNF ライフサイクルの管理 （25 ページ）を参照してください。

SR-IOV

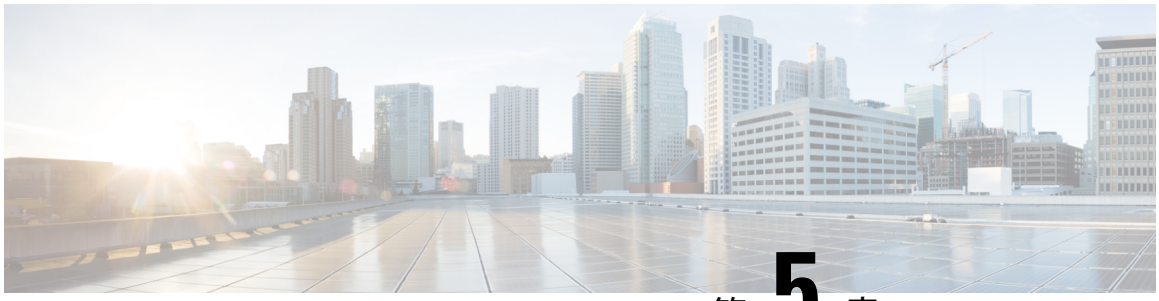
ESC ETSI NFV MANO は、Cisco データタイプを使用する SR-IOV プロパティをサポートします。VNFC を SR-IOV パススルーアダプタに関連付けるよう、インターフェイスを設定できます。

Cisco データタイプ：

```
cisco.datatypes.nfv.L2ProtocolData:
  derived_from: tosca.datatypes.nfv.L2ProtocolData
  properties:
    segmentation_id:
      type: integer
      required: false
```

VNFD の例：

```
  virtual_link_protocol_data:
  - associated_layer_protocol: ethernet
    l2_protocol_data:
      network_type: vlan
      physical_network: vlan_network
      segmentation_id: { get_input: VL1_SEG_ID }
```



第 5 章

VNF ライフサイクル操作の管理

VNF ライフサイクルの管理

NFVO は、VNF のライフサイクル管理に ETSI MANO API を使用して ESC と通信します。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 タイプの展開パラメータと運用動作を説明します。VNFD は、VNF を展開し、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。

VNF インスタンスのライフサイクル操作は次のとおりです。

1. **VNF ID の作成**：ESC は新しい VNF インスタンス ID（汎用一意識別子）を生成します。この ID は、その後の操作を実行するインスタンスを参照するハンドルとして使用されます。
2. **VNF のインスタンス化/展開**：VNF のインスタンス化の一環として、ESC は VIM の新しい VNF インスタンスをインスタンス化します。ESC は NFVO から VNF インスタンスをインスタンス化する要求を受信します。インスタンス化要求には、リソース要件、ネットワーキング、およびその他のサービス運用動作が含まれます。これらすべての要件と VNFD および付与情報は、VNF をインスタンス化するために必要なすべての情報を提供します。
3. **VNF の操作**：ESC を使用して、VNF インスタンスを開始および停止できます。リソースは解放も変更もされませんが、VIM の VNF インスタンスはこれら 2 つの状態の間で切り替わります。
4. **VNF のクエリ**：ESC が認識している 1 つ以上の VNF インスタンスをクエリします。これは、特定のインスタンスを検索するためにフィルタ処理できる特定の REST エンドポイントです。このインスタンスは、VNF インスタンス ID を使用してフィルタ処理できます。

   また、個別の REST エンドポイントにより、NFVO は VNF に関連付けられた 1 つ以上のライフサイクル操作オカレンスのステータスをクエリできます。ライフサイクル操作は、特定のオカレンス ID を使用してフィルタ処理できます。

5. **VNF の変更**：ESC では、1 つの VNF インスタンスのプロパティを変更できます。インスタンス化された VNF が更新され、ライフサイクル管理操作オカレンスが NFVO に VNF のステータスに関する通知を送信します。
6. **VNF のスケーリングとレベルへのスケーリング**：ESC では、2 つの方法で VNF をスケーリングできます。VNF は、段階的に、または特定のレベルにスケーリングできます。
7. **VNF の修復**：ESC は障害が発生したときに VNF を修復します。
8. **VNF の終了/展開解除**：VIM の VNF インスタンスを終了します。リソース自体は VNF インスタンス用に予約されたままですが、VNF 自体は展開解除されます。
9. **VNF ID の削除**：リソースは VIM および ESC で完全に解放され、関連付けられた VNF インスタンス ID も解放されます。

REST および NETCONF API を使用した VNF ライフサイクル操作については、『Cisco Elastic Services Controller User Guide』の「Configuring Deployment Parameters」を参照してください。

VNF ライフサイクル操作

VNFM の前提条件

VNF ライフサイクル操作では、次の前提条件を満たす必要があります。

- リソース定義はアウトオブバンドで作成する必要があり、VNFインスタンス化の前に使用できる必要があります。
- VIM への接続に関して、2 つのオプションがあります。VIM コネクタは、ESC が VIM に接続する方法を指定し、VNF を展開する前に作成して検証（および名前で識別）できます。または新しい vimConnectionInfo が指定された場合は、要求の一部として作成できます。VIM コネクタの概要 （13 ページ）を参照してください。

NFVO の前提条件

- インスタンス化する VNF は、ETSI 準拠の VNF パッケージ内で NFVO にオンボードする必要があります。
  - NFVO は、ETSI 準拠の VNF パッケージを ESC に提供する必要があります。
  - VNF パッケージには、VNF 記述子（VNFD）ファイルが含まれている必要があります。

  NFVO は、パッケージアーティファクトへのアクセスを許可するため、/vnf_packages API をサポートしている必要があります。詳細については、ETSI の Web サイトで *ETSI GS NFV-SOL 003* の仕様の第 10 章を参照してください。
- `/opt/cisco/esc/esc_database/` にあるプロパティファイル *etsi-product.properties* を更新します。プロパティファイルは、NFVO に関する詳細を ESC に提供します。

1 つのプロパティ *nfvo.apiRoot* では、NFVO のホストとポートを指定できます。たとえば、`nfvo.apiRoot` などです。

（注） ETSI MANO API の初期実装は、1 つの VIM のみをサポートします。テナント/プロジェクトは現在、resourceGroupId を使用して指定されます。

ETSI サービスで有効になっている HA モードの ESC に関する注意事項については、『Cisco Elastic Services Controller Install and Upgrade Guide』を参照してください。

展開要求

展開要求には、次のタスクが含まれます。

VNFD は次の構成の説明を提供します（詳細については、ETSI Web サイトの *ETSI GS NFV-SOL 001* の仕様を参照してください）。

- 展開フレーバや外部接続などの展開レベルの設定
- 適用可能なイメージを含む VDU 設定（コンピューティング）
- 内部接続ポイント（VduCp）
- 作成されるボリューム（適用可能なイメージ（VirtualBlockStorage）を含む）
- 内部仮想リンク（VnfVirtualLink）
- 配置、スケーリング、セキュリティに関するポリシーとグループ

InstantiateVnfRequest：

- 選択した展開フレーバ
- VIM 接続の詳細（vimConnectionInfo - Or-Vnfm のみ）
- 外部接続ポイント（extVirtualLinks）を接続する外部ネットワーク
- 内部仮想リンク（extManagedVirtualLinks）用にバインドできる外部ネットワーク
- 展開用に固有の変数を提供するキー値のペアのリスト（additionalParams）

NFVO からの付与（詳細については、ETSI Web サイトの *ETSI GS NFV-SOL 003* の仕様を参照）

- 追加、更新、または削除する、承認/更新されたリソース（UUID）
- 確認された配置情報

VNF ID の作成

VNF ID の作成は、あらゆる VNF インスタンスの最初の要求です。この ID は、ETSI API によってこれ以降実行されるすべての LCM 操作に使用されます。この段階では、リソースは作成も予約もされません。

ESC は POST 要求を送信して VNF インスタンスを作成します。

メソッドタイプ：

```
POST
```

VNFM エンドポイント：

```
 /vnf_instances/
```

HTTP 要求ヘッダー：

```
 Content-Type:application/json
```

要求ペイロード（ETSI データ構造：CreateVnfRequest）：

```
{
    "vnfInstanceName": "Test-VNf-Instance",
    "vnfdId": "vnfd-88c6a03e-019f-4525-ae63-de58ee89db74"
}
```

応答ヘッダー：

```
HTTP/1.1 201
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Application-Context: application:8250
Accept-Ranges: none
Location:
http://localhost:8250/vnflcm/v1/vnf_instances/14924fca-fb10-45da-bcf5-59c581d675d8
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 04 Jan 2018 12:18:13 GMT
```

応答本文（ETSI データ構造：VnfInstance）

```
{
    "id": "14924fca-fb10-45da-bcf5-59c581d675d8",
    "instantiationState": "NOT_INSTANTIATED",
    "onboardedVnfPkgInfoId": "vnfpkg-bb5601ef-cae8-4141-ba4f-e96b6cad0f74",
    "vnfInstanceName": "Test-VNf-Instance",
    "vnfProductName": "vnfd-1VDU",
    "vnfProvider": "Cisco",
    "vnfSoftwareVersion": "1.1",
    "vnfdId": "vnfd-88c6a03e-019f-4525-ae63-de58ee89db74",
    "vnfdVersion": "1.3",
    "_links": {
        "instantiate": {
            "href":
"http://localhost:8250/vnflcm/v1/vnf_instances/14924fca-fb10-45da-bcf5-59c581d675d8/instantiate"

        },
```

```
        "self": {
            "href":
"http://localhost:8250/vnflcm/v1/vnf_instances/14924fca-fb10-45da-bcf5-59c581d675d8"
        }
    }
}
```

VNF のインスタンス化については、仮想ネットワーク機能のインスタンス化 （29 ページ）を参照してください。

仮想ネットワーク機能のインスタンス化

インスタンス化要求によって多数のメッセージ交換がトリガーされ、VNF インスタンスをインスタンス化するためのコールフローが完了します。VNF インスタンスがインスタンス化されるときに、リソースが割り当てられます。これには、VNF 作成要求によって返され、要求がポストされる URL にエンコードされた VNF インスタンス ID が必要です。

フロー内のインスタンス化要求サブタスクは次のとおりです。

1. NFVO から VNF 記述子テンプレートを取得する。
2. NFVO から許可を要求する（双方向付与フロー）。詳細については、「付与経由での許可の要求」を参照してください。

メソッドタイプ：

```
POST
```

VNFM エンドポイント：

```
/vnf_instances/{vnfInstanceId}/instantiate
```

HTTP 要求ヘッダー：

```
Content-Type:application/json
```

要求ペイロード（ETSI データ構造：InstantiateVnfRequest）

```
{
    "flavourId": "default",
    "extManagedVirtualLinks": [
        {
            "id": "my-network",
            "resourceId": "93fb90ae-0ec1-4a6e-8700-bf109a0f4fba",
            "virtualLinkDescId": "VLD1"
        }
    ],
    "vimConnectionInfo": [
        {
            "accessInfo": {
                "password": "P@55w0rd!",
                "username": "admin",
                "vim_project": "tenantName"
            },
            "extra": {
                "name": "esc"
            },
            "id": "default_openstack_vim",
            "interfaceInfo": {
```

```
                "baseUrl": "http://localhost:8080"
            },
            "vimId": "default_openstack_vim",
            "vimType": "OPENSTACK"
        }
    ]
    "additionalParams": {
        "CPUS": 2,
        "MEM_SIZE": "512 MB",
        "VIM_FLAVOR": "Automation-Cirros-Flavor",
        "BOOTUP_TIME": "1800"
    }
}
```

flavourId 値は、VNFD で指定された 1 つの flavour_id と同じである必要があります。

（注） NFVO からの付与応答は、vimConnectionInfo を提供します。これは *SOL002* ペイロードでは提供されません。

VNFD テンプレートに変数を追加することで、インスタンス化の前に VNF をカスタマイズできます。LCM 要求の *additionalParams* フィールドに変数を指定します。値が文字列、数値、またはブール値のいずれかの場合、変数は名前と値のペアです。次の例では、get_input: <TOSCA method> を使用して、VNFD テンプレートの *cpus*、*mem_size*、および *additionalParams* が定義されます。

（注） 1 つの ETSI 展開で VNFD 内に複数の VM グループがある場合は、すべて同じ VIM を使用する必要があります。

このテンプレートが VNFM に送信されると、変数は同じ VNF インスタンスにマージされます。*additionalParams* 変数は VNF 変数とマージされ、変数の実際の値はインスタンス化中にのみ提供されます。

提供されるパラメータのリストは、VNFD のコンテンツによって決まります。要求で指定した additionalParams は、VNFD 内で get_input TOSCA メソッドを使用する VNFD によって使用されます。たとえば、cpus および mem_size 変数は VNFD 内のプレースホルダとマージされます。

```
tosca_definitions_version: tosca_simple_yaml_1_2

imports:
  - cisco_nfv_sol001_types.yaml
  - etsi_nfv_sol001_vnfd_0_10_0_types.yaml

metadata:
  template_name: Example
  template_author: Cisco Systems
  template_version: '1.0'

topology_template:
  inputs:

    CPUS:
      description: Number of CPUs
      type: string
```

```
      default: "2"
    MEM_SIZE:
      description: Memory size
      type: string
      default: "512 MB"
    VIM_FLAVOR:
      description: VIM Flavor
      type: string
      default: "Automation-Cirros-Flavour"
    BOOTUP_TIME:
      description: Time taken to boot the VNF
      type: string
      default: "1800"

node_templates:

    vdu1:
      type: cisco.nodes.nfv.Vdu.Compute
      properties:
        name: vdu1
        description: Example
        configurable_properties:
          additional_vnfc_configurable_properties:
            vim_flavor: { get_input: VIM_FLAVOR }
            bootup_time: { get_input: BOOTUP_TIME }
        vdu_profile:
          min_number_of_instances: 1
          max_number_of_instances: 1
      capabilities:
        virtual_compute:
          properties:
            virtual_cpu:
              num_virtual_cpu: { get_input: CPUS }
            virtual_memory:
              virtual_mem_size: { get_input: MEM_SIZE }
```

同じ VNF に対して *additionalParams* 変数を含むさらなる LCM 要求が送信されると、新しい変数が既存の変数を上書きします。VNFM は、インスタンス化に新しい変数を使用します。

内部リンクはエフェメラルになるように設計されていますが、一部の展開シナリオでは、VNF を超えた外部リンクにバインドできます。次の VNFD フラグメントの例を考えます。

```
     automation_net:
      type: tosca.nodes.nfv.VnfVirtualLink
      properties:
        connectivity_type:
          layer_protocols: [ ipv4 ]
        description: Internal Network VL
        vl_profile:
          max_bitrate_requirements:
            root: 10000
          min_bitrate_requirements:
            root: 0
```

VNF 展開で automation_net の代わりに使用する外部仮想リンクを指定するには、次のデータ構造をインスタンス化要求の一部として使用する必要があります。

```
...
"extManagedVirtualLinks": [
      {
            "id": "net-5ddc8435-9d85-4560-8b95-bfcd3369c5c2",
            "resourceId": "esc-net2",
```

```
            "vimConnectionId":"default_openstack_vim",
            "virtualLinkDescId": "automation_net"
        }
],
...
```

ETSI 仕様ではエフェメラルボリュームの概念しかサポートしていませんが、多くのベンダーは永続的なボリュームの仕様を求めているため、シスコはこれをサポートする拡張機能を実装しました。次の例に示すように、永続的なボリュームのリソース ID を additionalParam として指定し、オプションのプロパティを使用して VNFD のボリュームに関連付けることができます。

```
       example-volume:
      type: cisco.nodes.nfv.Vdu.VirtualBlockStorage
      properties:
        resource_id: { get_input: EX_VOL_UUID }
        virtual_block_storage_data:
          size_of_storage: 200 GB
          vdu_storage_requirements:
            vol_id: 1
            bus: ide
            type: LUKS
```

付与経由での許可の要求

ETSI API は、VNF インスタンスリソースのライフサイクル管理操作を完了するために NFVO からの許可を要求し、事前プロビジョニングされたリソースのリソース ID を取得します。GrantRequest の例を次に示します。

```
{
  "flavourId": "default",
  "instantiationLevelId": "default",
  "isAutomaticInvocation": false,
  "operation": "INSTANTIATE",
  "vnfInstanceId": "e426a94e-7963-430c-96ee-778dde5bd021",
  "vnfLc mOpOccId": "06fe989b-7b0b-40dc-afb3-de26c18651ae",
  "vnfdId": "6940B47B-B0D0-48CB-8920-86BC23F91B16",
  "addResources":
  [
    {
      "id": "res-1abb1609-a1f3-418a- a7a0-2692a5e53311",
      "resourceTemplateId": "vdu1",
      "type": "COMPUTE",
      "vduId": "vdu1"
    },
    {
      "id": "res-c5ece35c-89e3-4d29-b594-ee9f6591f061",
      "resourceTemplateI d": "node_1_nic0",
      "type": "LINKPORT",
      "vduId": "vdu1"
    },
    {
      "id": "res-e88d8461-5f5a-4dba-af14-def82ce894e5",
      "resourceTemplateId": "automation_net",
      "type": "VL"
    }
  ],
  "_links":
  {
    "vnfInstance":
```

```
    {
      "href": "https://172.16
.255.8:8251/vnflcm/v1/vnf_instances/14924fca-fb10-45da-bcf5-59c581d675d8"
    },
    "vnfLcmOpOcc":
    {
      "href":
"https://172.16.255.8:8251/vnflcm/v1/vnf_lcm_op_occs/457736f0-c877-4e07-8055-39dd406c616b"

    }
  }
}
```

返された対応する付与は、次のようになります。

```
{
    "id": "grant-0b7d3420-e6ee-4037-b116-18808dea4e2a",
    "vnfInstanceId": "14924fca-fb10-45da-bcf5-59c581d675d8",
    "vnfLcmOpOccId": "457736f0-c877-4e07-8055-39dd406c616b",
    "addResources": [
        {
            "resourceDefinitionId": "res-1abb1609-a1f3-418a-a7a0-2692a5e53311",
            "vimConnectionId": "esc-005e4412-e056-43a9-8bc0-d6699c968a3c"
        },
        {
            "resourceDefinitionId": "res-c5ece35c-89e3-4d29-b594-ee9f6591f061",
            "vimConnectionId": "esc-005e4412-e056-43a9-8bc0-d6699c968a3c"
        },
        {
            "resourceDefinitionId": "res-e88d8461-5f5a-4dba-af14-def82ce894e5",
            "vimConnectionId": "esc-005e4412-e056-43a9-8bc0-d6699c968a3c"
        }
    ],
    "vimAssets": {
        "computeResourceFlavours": [
            {
                "vimConnectionId": "esc-005e4412-e056-43a9-8bc0-d6699c968a3c",
                "vimFlavourId": "Automation-Cirros-Flavor",
                "vnfdVirtualComputeDescId": "vdu1"
            }
        ],
        "softwareImages": [
            {
                "vimConnectionId": "esc-005e4412-e056-43a9-8bc0-d6699c968a3c",
                "vimSoftwareImageId": "Automation-Cirros-DHCP-2-IF",
                "vnfdSoftwareImageId": "vdu1"
            }
        ]
    },
    "vimConnections": [
        {
            "id": "esc-005e4412-e056-43a9-8bc0-d6699c968a3c",
            "vimId": "default_openstack_vim",
            "vimType": "OPENSTACK",
            "accessInfo": {
                "vim_project": "admin"
            }
        }
    ],
    "zones": [
        {
            "id": "zone-c9f79460-7a23-43e4-bb6d-0683e2cdb3d4",
            "vimConnectionId": "default_openstack_vim",
            "zoneId": "default"
        },
```

```
        {
            "id": "zone-4039855e-a2cb-48f8-996d-b328cdf9889a",
            "vimConnectionId": "default_openstack_vim",
            "zoneId": "nova"
        }
    ],
    "_links": {
        "self": {
            "href":
"http://localhost:8280/grant/v1/grants/grant-0b7d3420-e6ee-4037-b116-18808dea4e2a"
        },
        "vnfInstance": {
            "href": "https://172.16
.255.8:8251/vnflcm/v1/vnf_instances/14924fca-fb10-45da-bcf5-59c581d675d8"
        },
        "vnfLcmOpOcc": {
            "href":
"https://172.16.255.8:8251/vnflcm/v1/vnf_lcm_op_occs/457736f0-c877-4e07-8055-39dd406c616b"

        }
    }
}
```

付与要求は、要求されたすべてのリソースが付与されている場合にのみ受け入れられます。そうでない場合、付与は拒否されます。

ESC からの展開記述子の取得

NFVO は、ESC データモデルインスタンスを展開記述子の形式で取得できます。NFVO は、インスタンス化の際に提供されたすべての入力と、後で展開記述子に加えられた変更を表示できます。

展開記述子を取得するには、次の手順を実行する必要があります。

- VNF の作成
- vnfinstanceId を指定します

メソッドタイプ

```
GET
```

VNFM エンドポイント

```
/vnflcm/v1/ext/vnfinstances/{vnfInstanceId}/deployment
```

HTTP 要求ヘッダー

```
content-Type:application/xml
```

要求ペイロード

該当なし。

仮想ネットワーク機能のクエリ

VNF のクエリは、VNF インスタンスの状態には影響を与えません。この操作は、既知のすべての VNF インスタンス、または特定の VNF インスタンスについて ESC にクエリするだけです。

メソッドタイプ：

```
GET
```

VNFM エンドポイント：

```
/vnf_instances/vnf_instances/{vnfInstanceId}
```

HTTP 要求ヘッダー：

```
Content-Type: application/json
```

要求ペイロード：

```
not applicable.
```

応答ヘッダー：

```
< HTTP/1.1 200
HTTP/1.1 200
< X-Content-Type-Options: nosniff
X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
X-XSS-Protection: 1; mode=block
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Pragma: no-cache
Pragma: no-cache
< Expires: 0
Expires: 0
< X-Frame-Options: DENY
X-Frame-Options: DENY
< Strict-Transport-Security: max-age=31536000 ; includeSubDomains
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
< X-Application-Context: application:8250
X-Application-Context: application:8250
< Accept-Ranges: none
Accept-Ranges: none
< ETag: "2"
ETag: "2"
< Content-Type: application/json;charset=UTF-8
Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
Transfer-Encoding: chunked
< Date: Thu, 04 Jan 2018 12:25:32 GMT
Date: Thu, 04 Jan 2018 12:25:32 GMT
```

1 つの VNF インスタンスの応答本文（ETSI データ構造：VnfInstance）

（注） ETag 応答ヘッダーは、1 つの VNF クエリ（VNF インスタンス ID が指定されたクエリ）に対してのみ返されます。ETag 値は、後続の VNF 変更操作中に条件付きで使用されます。

```
{
  "_links": {
    "instantiate": {
      "href":
"http://localhost:8250/vnflcm/v1/vnf_instances/14924fca-fb10-45da-bcf5-59c581d675d8/instantiate"

    },
    "self": {
      "href":
"http://localhost:8250/vnflcm/v1/vnf_instances/14924fca-fb10-45da-bcf5-59c581d675d8"
    }
  },
  "id": "14924fca-fb10-45da-bcf5-59c581d675d8",
  "instantiationState": "NOT_INSTANTIATED",
  "onboardedVnfPkgInfoId": "vnfpkg-bb5601ef-cae8-4141-ba4f-e96b6cad0f74",
  "vnfInstanceName": "Test-VNf-Instance",
  "vnfProductName": "vnfd-1VDU",
  "vnfProvider": "Cisco",
  "vnfSoftwareVersion": "1.1",
  "vnfdId": "vnfd-88c6a03e-019f-4525-ae63-de58ee89db74",
  "vnfdVersion": "2.1"
}
```


VNF のクエリ操作の出力は、VNF のインスタンス化された状態を示します。*InstantiatedVnfInfo* 要素は、すべての VNF の VIM リソース情報を示します。

次に例を示します。


```
{
"instantiatedVnfInfo": {
"extCpInfo": [
{
"cpProtocolInfo": [
{
"ipOverEthernet": {
"ipAddresses": [
{
"addresses": [
"172.16.235.19"
],
"isDynamic": false,
"type": "IPV4"
}
],
"macAddress": "fa:16:3e:4b:f8:03"
},
"layerProtocol": "IP_OVER_ETHERNET"
}
],
"cpdId": "anECP",
"id": "extCp-4143f7d4-f581-45fc-a730-568435dfdb4f"
}
],
"extManagedVirtualLinkInfo": [
{
"id": "net-d39bc4de-285c-4056-8113-24eccf821ebc",
"networkResource": {
"resourceId": "my-network",
"vimConnectionId": "esc-b616e5be-58ce-4cfc-8eee-e18783c5ae5d"
},
"vnfLinkPorts": [
{
```

```
"cpInstanceId": "vnfcCp-9b24c9e0-1b28-4aba-a9df-9bfc786bfaed",
"id": "vnfLP-9b24c9e0-1b28-4aba-a9df-9bfc786bfaed",
"resourceHandle": {
"resourceId": "926b7748-61d9-4295-b9ff-77fceb05589a",
"vimConnectionId": "esc-b616e5be-58ce-4cfc-8eee-e18783c5ae5d"
}
}
],
"vnfVirtualLinkDescId": "my-network"
}
],
"extVirtualLinkInfo": [
{
"extLinkPorts": [
{
"cpInstanceId": "extCp-4143f7d4-f581-45fc-a730-568435dfdb4f",
"id": "extLP-4143f7d4-f581-45fc-a730-568435dfdb4f",
"resourceHandle": {
"resourceId": "d6a4c231-e77c-4d1f-a6e2-d3f463c4ff72",
"vimConnectionId": "default_openstack_vim"
}
}
],
"id": "extVL-b9bd55a9-4bd9-4ad8-bf67-ba1e7b82aca6",
"resourceHandle": {
"resourceId": "anECP",
"vimConnectionId": "esc-b616e5be-58ce-4cfc-8eee-e18783c5ae5d"
}
}
],
"flavourId": "bronze",
"scaleStatus": [
{
"aspectId": "default_scaling_aspect",
"scaleLevel": 1
}
],
"vnfState": "STARTED",
"vnfcResourceInfo": [
{
"computeResource": {
"resourceId": "a21f0b15-ec4b-4968-adce-1ccfad118caa",
"vimConnectionId": "default_openstack_vim"
},
"id": "res-89a669bb-fef4-4099-b9fe-c8d2e465541b",
"vduId": "vdu_node_1",
"vnfcCpInfo": [
{
"cpProtocolInfo": [
{
"ipOverEthernet": {
"ipAddresses": [
{
"addresses": [
"172.16.235.19"
],
"isDynamic": false,
"type": "IPV4"
}
],
"macAddress": "fa:16:3e:4b:f8:03"
},
"layerProtocol": "IP_OVER_ETHERNET"
}
```

```
],
"cpdId": "node_1_nic0",
"id": "vnfcCp-c09d5cf2-8727-400e-8845-c4d5cb479db8",
"vnfExtCpId": "extCp-4143f7d4-f581-45fc-a730-568435dfdb4f"
},
{
"cpProtocolInfo": [
{
"ipOverEthernet": {
"ipAddresses": [
{
"addresses": [
"172.16.235.16"
],
"isDynamic": false,
"type": "IPV4"
}
],
"macAddress": "fa:16:3e:94:b3:91"
},
"layerProtocol": "IP_OVER_ETHERNET"
}
],
"cpdId": "node_1_nic1",
"id": "vnfcCp-9b24c9e0-1b28-4aba-a9df-9bfc786bfaed"
}
]
}
]
}
```


VNF クエリの属性の選択

属性セレクタを使用して、VNF クエリ応答に表示する属性を選択できます。属性をクエリに含めるか除外するかをマークできます。たとえば、基数の下限が 0 である属性（0..1、0..N など）や、必須ではない属性（特定の条件による）など、必要ない属性を除外できます。

クエリで必要な属性のみを選択すると、インターフェイス上で交換され、API コンシューマ アプリケーションによって処理されるデータ量が減少します。

次の表に、GET 要求の属性を選択するための URI クエリパラメータを示します。

表 *4: GET* 要求の属性の選択

パラメータ	定義
all_fields	exclude_default によって抑制された属性を含め、応答に含まれるすべての複合属性を要求します。これは *exclude_default* パラメータの反対です。API プロデューサは、特定のリソースで all_fields パラメータをサポートします。 （注）　複合属性は、構造化された属性または配列です。

パラメータ	定義
fields	リストされた複合属性のみを応答に含める要求。 パラメータは、属性名のリストとしてフォーマットされます。属性名は、属性の名前、または「/」で区切られた、親子関係を持つ複数の属性の名前で構成されるパスのいずれかです。リスト内の属性名は、カンマ（「,」）で区切ることができます。特定の GET 要求の有効な属性名は、基数の下限が 0 で、条件付きで必須ではない、予期される応答内のすべての複合属性の名前です。 API プロデューサは、特定のリソースでフィールドパラメータをサポートします。詳細は、実際のリソースを指定する句で定義されます。 属性セレクタの属性名の「/」および「~」文字は、IETF 標準に従ってエスケープされます。 属性セレクタの属性名の「,」文字は、「~a」に置き換えてエスケープされます。 さらに、パーセントエンコーディングは、IETF 標準に従って URI クエリパートで許可されない文字に適用されます。
exclude_fields	リストされた複合属性を応答から除外する要求。フォーマットについては、適格な属性と API プロデューサによるサポート、「fields」パラメータで定義される規定が適用されます。
exclude_default	複合属性のデフォルト設定を応答から除外する要求。すべてのリソースにデフォルト設定があるわけではありません。条件付きで必須ではなく、基数の下限が 0 の複合属性のみ、設定に含めることができます。 API プロデューサは、特定のリソースでこのパラメータをサポートします。 exclude_default パラメータはフラグであり、値はありません。 リソースが属性セレクタをサポートし、GET 要求で属性セレクタパラメータが指定されていない場合、**exclude_default** パラメータがデフォルトになります。GET 要求の元の動作をエミュレートするには、**all_fields** フラグを指定するか、ETSI プロパティの **attribute.selector.default.all_fields** を true に設定します。これにより、all_fields に 属性セレクタが指定されていない場合、動作が変わります。

GET 応答は、GET 要求のパラメータの組み合わせを検証します。表は、有効なパラメータの組み合わせの定義です。

表 *5: Get* 応答のパラメータの組み合わせ

パラメータの組み合わせ	GET 応答
（なし）	exclude_default と同じものが含まれます。
all_fields	すべての属性が含まれます。
fields=<list>	条件付きで必須ではなく、最小の基数が 0、**および**<list> で提供されないすべての複合属性を除く、すべての属性が含まれます。
exclude_fields=<list>	条件付きで必須ではなく、最小の基数が 0、**および** <list> で提供されるすべての複合属性を**除く**、すべての属性が含まれます。
exclude_default	条件付きで必須ではなく、最小の基数が 0、**および**特定のリソースの現在のドキュメントで定義された *default exclude set* の一部である複合属性を**除く**、すべての属性が含まれます。
exclude_default and fields=<list>	条件付きで必須ではなく、最小の基数が 0、**および**特定のリソースの現在のドキュメントで定義された *default exclude set* の一部で**あるが**、<list> の一部ではない複合属性を**除く**、すべての属性が含まれます。

VNF インスタンス、VNF LCM 操作オカレンス、PM ジョブなどのリソースに対する GET 要求は、属性の選択をサポートします。

表 *6:* 属性の選択をサポートするリソース

[名前（Name）]	Cardinality	説明
VNF インスタンス		

[名前（Name）]	Cardinality	説明
exclude_default	0..1	応答から次の複合属性を除外することを示します。 このパラメータが指定されている場合、またはパラメータ（all_fields、fields、exclude_fields、exclude_default）のいずれも指定されていない場合、次の属性が応答本文の VnfInstance 構造から除外されます。 • vnfConfigurableProperties • vimConnectionInfo • InstantiatedVnfInfo • メタデータ • 内線番号
VNF LCM 操作のオカレンス		
exclude_default	0..1	このパラメータが指定されている場合、またはパラメータ（all_fields、fields、exclude_fields、exclude_default）のいずれも指定されていない場合、次の属性が応答本文の VnfLcmOpOcc 構造から除外されます。 • operationParams • error • resourceChanges • changedInfo • changedExtConnectivity
PM Jobs		

[名前（Name）]	Cardinality	説明
exclude_default	0..1	このパラメータが指定されている場合、またはパラメータ（all_fields、fields、exclude_fields、exclude_default）のいずれも指定されていない場合、次の属性が応答本文の PmJob 構造から除外されます。 • レポート

VNF ライフサイクル操作の詳細については、VNF ライフサイクル操作 （26 ページ）を参照してください。

仮想ネットワーク機能の変更

VNF ライフサイクル変更操作を使用して、NOT_INSTANTIATED 状態の VNF インスタンスのプロパティを変更または更新できます。ESC は 1 つの VNF インスタンスを変更するため、NFVO からの PATCH 要求を受信します。

保存されたデータに対して入力ペイロードからJSONマージアルゴリズムが適用され、VNFインスタンスが変更されます。

（注） VNF変更操作によりプロパティのみが更新され、VNFの機能は更新されません。変更操作は、NOT_INSTANTIATED の VNF インスタンスリソースでのみ有効です。

既存の VNF インスタンスの次のプロパティを変更できます。

- vnfInstanceName
- vnfInstanceDescription
- onboardedVnfPkgInfoId（null 値は不可）
- vnfConfigurableProperties
- メタデータ
- 内線番号
- vimConnectionInfo

メソッドタイプ

```
PATCH
```

VNFM エンドポイント

```
/vnf_instances/{vnfInstanceId}
```

HTTP 要求ヘッダー

```
Content-Type: application/merge-patch+json
If-Match: ETag value
```

（注） ETag（指定されている場合）は、VNF インスタンスリソースに保存されている ETag 値に対して検証されます。値が一致しない場合、変更要求は拒否されます。

要求ペイロード（ETSI データ構造：VnfInfoModifications）

```
{
    "vnfInstanceName": "My NEW VNF Instance Name",
    "vnfInstanceDescription": "My NEW VNF Instance Description",
    "vnfPkgId": "pkg-xyzzy-123",
    "vnfConfigurableProperties": {
        "isAutoscaleEnabled": "true"
    },
    "metadata": {
        "serialRange": "ab123-cc331",
        "manufacturer": "Cisco"
    },
    "extensions": {
        "testAccess": "false",
        "ipv6Interface": "false"
    },
    "vimConnectionInfo": [
        {
            "id": "vci1",
            "vimType": "openstack",
            "interfaceInfo": {
                "uri": "http://172.16.14.27:35357/v3"
            },
            "accessInfo": {
                "domainName": "default",
                "projectName": "admin",
                "userName": "default"
            }
        }
    ]
}
```

（注） NFVO からの 付与応答は、*SOL002* ペイロードの代わりに vimConnectionInfo を提供します。*SOL002* 要求には、vnfcInfoModifications などのより細かい VNFC レベルで VNF リソースに影響を与える属性が含まれています。詳細については、*ETSI Web* サイトの *SOL002* を参照してください。

応答ヘッダー：

```
not applicable.
```

応答本文：

```
not applicable.
```

PATCH 操作が完了すると、VNF インスタンスが変更され、通知を通じて詳細が NFVO に送信されます。

仮想ネットワーク機能の操作

操作ライフサイクル管理操作を使用して、VNFインスタンスを開始または停止できます。VNFインスタンスは、猶予を与えて、または強制的に停止できます。

（注） OpenStack API は強制停止のみをサポートします。

VNF インスタンスを開始または停止するには、*changeStateTo* フィールドの要求ペイロードに値 STARTED または STOPPED が含まれている必要があります。

この操作には、NFVO（双方向付与フロー）からの権限も必要です。詳細については、「付与フローの要求」を参照してください。

メソッドタイプ：

```
POST
```

VNFM エンドポイント：

```
/vnf_instances/{vnfInstanceId}/operate
```

HTTP 要求ヘッダー：

```
コンテンツタイプ：application / json
```

応答ヘッダー：

```
HTTP/1.1 202
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: TEST
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Application-Context: application:8250
Accept-Ranges: none
Location:
http://localhost:8250/vnflcm/v1/vnf_lcm_op_occs/e775aad5-8683-4450-b260-43656b6b13e9
Content-Length: 0
Date: Thu, 04 Jan 2018 12:40:27 GMT
```

応答本文：

```
not applicable.
```

仮想ネットワーク機能の終了

VNF の終了要求により、VNF インスタンスが終了します。リソースは割り当て解除されますが、削除されるまでこのインスタンス用に予約されたままになります。この操作には、NFVO

からの権限（双方向付与フロー）が必要です。VNFインスタンスは、猶予を与えて、または強制的にデコミッションできます。

（注） OpenStack API は強制終了のみをサポートします。

VNF のインスタンス化要求に従い、VNF の終了要求には、要求がポストされる URL にエンコードされた VNF インスタンス ID が必要です。

方式タイプ：

```
POST
```

VNFM エンドポイント：

```
/vnf_instances/{vnfInstanceId}/terminate
```

HTTP 要求ヘッダー：

```
Content-Type:application/json
```

要求ペイロード（ETSI データ構造：TerminateVnfRequest）

```
{
  "terminationType":"FORCEFUL",
  }
```

応答ヘッダー：

```
HTTP/1.1 202
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: TEST
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Application-Context: application:8250
Accept-Ranges: none
Location:
http://localhost:8250/vnflcm/v1/vnf_lcm_op_occs/dae25dbc-fcde-4ff9-8fd6-31797d19dbc1
Content-Length: 0
Date: Thu, 04 Jan 2018 12:45:59 GMT
```

応答本文：

```
not applicable.
```

仮想ネットワーク機能リソース ID の削除

VNF 操作を削除すると、VNF インスタンス用に予約された VIM リソースが解放され、VNF インスタンス ID も削除されます。削除すると、VNF インスタンス ID は使用できなくなります。そのため、この ID を使用したライフサイクル管理操作はできなくなります。

メソッドタイプ：

```
DELETE
```

VNFM エンドポイント：

```
/vnf_instances/{vnfInstanceId}
```

HTTP 要求ヘッダー：

```
Content-Type:application/json
```

要求ペイロード：

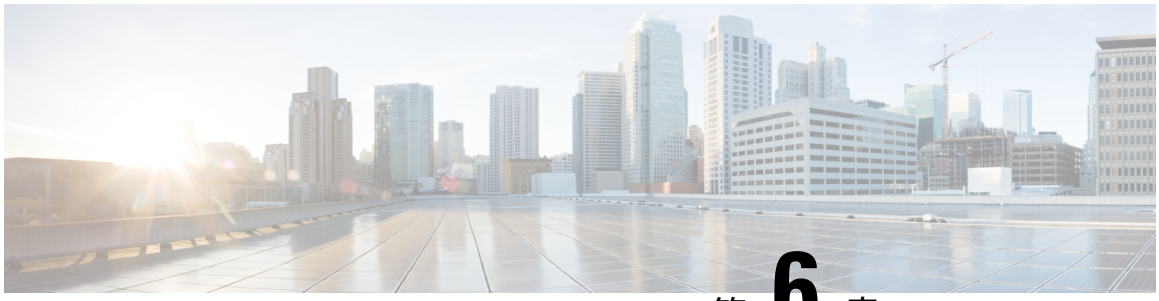
```
not applicable.
```

応答ヘッダー：

```
HTTP/1.1 204
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: TEST
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Application-Context: application:8250
Accept-Ranges: none
Date: Thu, 04 Jan 2018 12:48:59 GMT
```

応答本文：

```
not applicable.
```



第 6 章

仮想ネットワーク機能のモニタリング

ETSI API を使用した仮想ネットワーク機能のモニタリング

VNF の導入中に、ESC モニタリング エージェント コンポーネント（MONA）に VNF が正常かどうかを判断する方法を指示するメトリックを定義する必要があります。メトリックの定義は VNFD の重要業績評価指標（KPI）セクション内にあり、MONA は VNF を定期的にモニタして、VNFC ごとに定義された、その健全性とワークロードを確認できます。その後アクションはこれらの KPI に関連付けられ、適切な条件が満たされると実行されます。

ICMP Ping や SNMP など、いくつかの組み込みモニタリングメソッドがあります。構成 VNFC でモニタするメトリックには、次のものがあります。

- reachability
- リソース使用率（CPU、メモリ、ディスク、ネットワークスループットなど）

展開した VNFC 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。

- 展開した VNFC が動作している必要がある
- モニタリングが有効になっている
- KPI が設定されている必要がある

例：

```
vdu2:
      type: cisco.nodes.nfv.Vdu.Compute
      properties:
        name: Example VDU 2
        description:  Example VDU
        ...
        kpi_data:
          VM_ALIVE-1:
```

```
      event_name: 'VM_ALIVE-1'
      metric_value: 1
      metric_cond: 'GT'
      metric_type: 'UINT32'
      metric_occurrences_true: 1
      metric_occurrences_false: 30
      metric_collector:
        type: 'ICMPPing'
        nicid: 1
        poll_frequency: 10
        polling_unit: 'seconds'
        continuous_alarm: false
        property_list:
            - name: vmname
              value: vdu2
            - name: status
            value: ERROR
  admin_rules:
    VM_ALIVE-1:
      event_name: 'VM_ALIVE-1'
      action:
        - 'ALWAYS log'
        - 'FALSE recover autohealing'
        - 'TRUE esc_vm_alive_notification'
        property_list:
            - name: vmname
              value: vdu2
            - name: status
            value: SUCCESS
...
```

上記の kpi_data はデフォルトで必要な KPI で、最低でもすべての展開で必要です。これにより、VM_ALIVE メッセージが生成され、VNFC が正常に展開されたことが ESC Manager に通知されます。これは、KPI、その収集方法、およびKPIが満たされたときに実行されるアクションで構成されています。

シスコのデータ構造プロパティ

データ タイプ	プロパティ名	説明	値
cisco.datatypes.nfv.data.Kpi	KPI ラベル	ユーザ定義の一意の KPI 名	任意（Any）
cisco.datatypes.nfv.data.Kpi	event_name		
cisco.datatypes.nfv.data.Kpi	metric_value		
cisco.datatypes.nfv.data.Kpi	metric_cond		
cisco.datatypes.nfv.data.Kpi	metric_type		
cisco.datatypes.nfv.data.Kpi	metric_occurrences_true		
cisco.datatypes.nfv.data.Kpi	metric_occurrences_false		
cisco.datatypes.nfv.metric.Collector	type	『NETCONF API Guide』を参照してください。	『NETCONF API Guide』を参照してください。

データ タイプ	プロパティ名	説明	値
cisco.datatypes.nfv.metric.Collector	nicid		
cisco.datatypes.nfv.metric.Collector	poll_frequency		
cisco.datatypes.nfv.metric.Collector	polling_unit		
cisco.datatypes.nfv.metric.Collector	continuous_alarm		
cisco.datatypes.nfv.metric.Collector	property_list		
cisco.datatypes.nfv.data.Admin_rules	ルールラベル	一意のユーザ定義名	任意（Any）
cisco.datatypes.nfv.data.Admin_rules	event_name	この値は、Kpi event_name と一致する必要があります	
cisco.datatypes.nfv.data.Admin_rules	action		
cisco.datatypes.nfv.data.Admin_rules	property_list		

KPI とルールの詳細については、『*Cisco Elastic Services Controller User Guide*』を参照してください。

VM 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 をモニタするにはペイロードが必要です。

REST コード

```
POST <VNFM-api-root>/vnflcm/v1/ext/vnf_instances/{vnfInstanceId}/operations
```

指定した VM でモニタリング動作を開始および停止するには、`vnfcInstanceIds` を設定します

```
POST /v0/{internal_tenant_id}/deployments/vm/{vm_name}
```

以下のペイロードで：

```
{
    "vnfcInstanceIds": ["vnfcInstanceId1","vnfcInstanceId2",...,"vnfcInstaceIdN"],
## optional
    "operation": "ENABLE_MONITOR",
## mandatory ENABLE_MONITOR, DISABLE_MONITOR, REBOOT
    "additionalParams": []
## optional - for future use :-)
}
```

VNF 全体のモニタリング動作を開始および停止するには、`vnfcInstanceIds` を設定しないでください。

VM モニタリングを設定するには、enable_monitoring を指定し、VM モニタリングを設定解除するには操作フィールドで disable_monitoring を指定する必要があります。

（注） ユーザが ESC ETSI インターフェイスから 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。

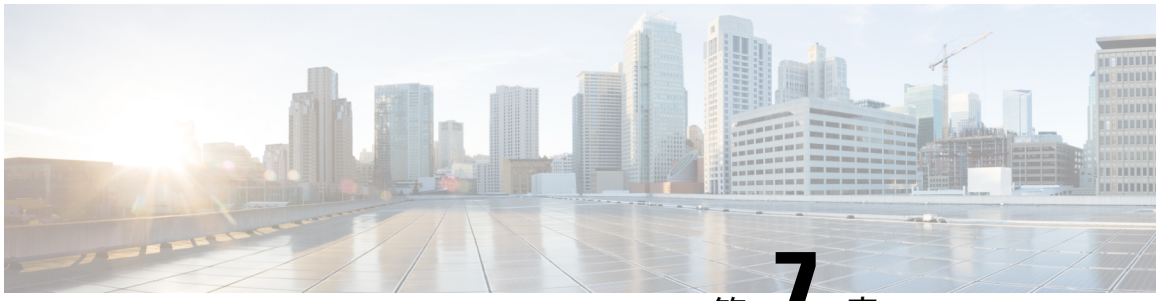
VM モニタリングステータスの通知

ETSI NFV MANO は、VM モニタリングのステータス通知を提供します。ペイロードを使用して、特定の VNF または VNF の特定の VM で、VM を有効化、無効化、再起動できます。

```
[operation]
--------------
enable a monitor for
disable a monitor for
reboot
```

VM を設定、設定解除、または再起動時に、ETSI NFV MANO は次の `[notifications-per-operation]` を送信します。

```
[notifications-per-operation]
-------------------------------------
VM_MONITOR_SET notification when enabling a monitor
VM_MONITOR_UNSET notification when disabling a monitor
VM_REBOOTED notification when rebooting
```



第 7 章

D-MONA を使用した VNF のモニタリング

D-MONA のオンボーディング

ETSI NFV MANO は、VNF の効果的なモニタリングのため、分散型モニタリングとアクション（D-MONA）をサポートします。D-MONA は、スタンドアロンのモニタリングアプリケーションです。詳細については、『Cisco Elastic Services Controller User Guide』の「Monitoring VNFs Using D-MONA」を参照してください。

D-MONA 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。

前提条件

- ESC と D-MONA 間の接続を確認します。
- D-MONA と展開された VNF 間の接続を確認します。
- D-MONA では ESC アクティブ/アクティブ展開のみサポートされます。

D-MONA の展開の詳細については、D-MONA の展開 （51 ページ）を参照してください。

D-MONA の展開

ESC は、VIM の 1:1 D-MONA 展開をサポートします。1 つの D-MONA インスタンスが、1 つの VIM の 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、それぞれのライブ状態をモニタします。

展開後、D-MONA は ESC VM で実行されているローカルの MONA によってモニタされます。

D-MONA を使用した VNF の展開については、D-MONA を使用した VNF の展開 （52 ページ）を参照してください。

D-MONA の設定

D-MONA は ESC 5.0 イメージを再利用します。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 展開から、もう 1 つは D-MONA が提供する機能で表示できます。

D-MONA Day 0 設定

D-MONA のランタイム動作は、展開時に VM に提供される day 0 の設定によって制御されます。

次に、D-MONA SSH アクセスの設定例を示します。

```
<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。

次に、D-MONA ESC 証明書の設定例を示します。

```
<configuration>
 <dst>/opt/cisco/esc/moan/dmona.crt</dst>
 <data>$DMONA_CERT</data>
</configuration>
```

D-MONA を使用したモニタリングについては、D-MONA を使用したモニタリング （53 ページ）を参照してください。

D-MONA を使用した VNF の展開

D-MONA を使用して VNF を展開するには、同じ `vim_connector` 内で、`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 を検出すると、VNF のモニタリングがその D-MONA に割り当てられます。それ以外の場合は、ローカルの MONA がモニタリングを処理します。

次に、D-MONA VNFD の例を示します。

```
tosca_definitions_version: tosca_simple_yaml_1_2
description: D-MONA VNFD (SOL001 v0.10.0)
```

```
imports:
  - cisco_nfv_sol001_types.yaml
  - etsi_nfv_sol001_vnfd_0_10_0_types.yaml

metadata:
  template_name: D-MONA
  template_author: Cisco Systems
  template_version: '1.0'

dsl_definitions:
  descriptor_id: &descriptor_id f5b37b47-d9bd-4605-afb0-30c0d659a3c2
  provider: &provider cisco
  product_name: &product_name D-MONA
  software_version: &software_version '1.0'
  descriptor_version: &descriptor_version '1.0'
  flavour_id: &flavour_id default
  flavour_description: &flavour_description 'Default VNF Deployment Flavour'
  vnfm: &vnfm '9:Cisco Elastic Services Controller:v04.04.01'
```

D-MONA を使用したモニタリング

D-MONA を使用して VNF をモニタするには、ESTI VNFD D-MONA を展開してから、D-MONA によってモニタされる ESTI VNFD を展開する必要があります。D-MONA の展開の詳細については、D-MONA を使用した VNF の展開 （52 ページ）を参照してください。

D-MONA パラメータは VNFD 内で定義されるか、またはインスタンス化 D-MONA VNF ペイロードで additionalparams として提供されます。

D-MONA の展開には、ETSI 準拠の VNFD が使用されます。

D-MONA 展開のインスタンス化には、入力パラメータ、KPI データ、および設定パラメータが必要です。

入力パラメータは VNFD 内で定義されるか、またはインスタンス化 D-MONA VNF ペイロードの additionalParams セクションとして提供されます。

表 *7 : D-MONA* 展開の入力パラメータ

パラメータ	説明
SW_IMAGE_NAME	ESC イメージの名前
DMONA_CERT	HTTPS 証明書
DMONA_AGENT_ID	VM をモニタするモニタリングエージェントの URL または ID
ADMIN_PASSWORD	管理ユーザのパスワード
SECURITY_BASIC_ENABLED	基本セキュリティが有効かどうかを示すフラグ
SECURITY_USER_NAME	ESCManager と通信するセキュリティユーザ

パラメータ	説明
SECURITY_USER_PASSWORD	ESCManager との通信に使用されるセキュリティユーザのパスワード

KPI データ：

- monitoring_agent：入力パラメータの DMONA_AGENT_ID で定義された値。
- property_list
  - name：プロトコル
  - value：https
  - name：ポート
  - value：8443
  - name：path
  - value：mona/v1/health/status

データ設定パラメータ：

- user-data.txt

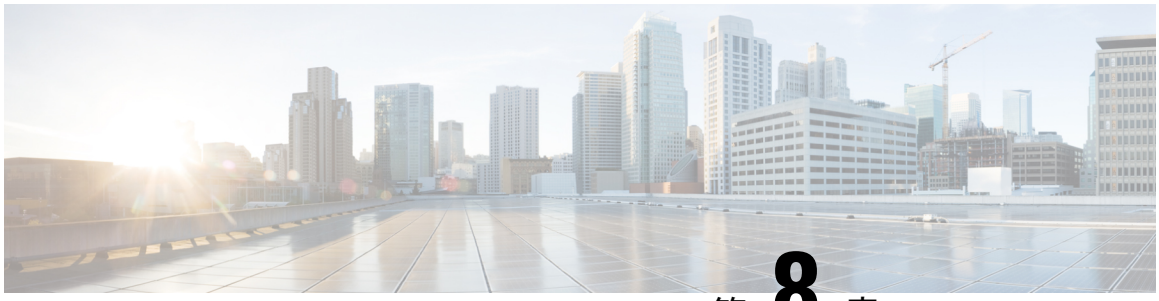
  admin_password：入力パラメータの ADMIN_PASSWORD で定義された値

- application：dmona.template
  - monitoring.agent：true
  - security_basic_enabled：入力パラメータの SECURITY_BASIC_ENABLED で定義された値
  - security_user_name：入力パラメータの SECURITY_USER_NAME で定義された値
  - security_user_password：入力パラメータの SECURITY_USER_PASSWORD で定義された値
  - monitoring.agent.vim.mapping：true

ペイロードの例：

```
config_data:
    '--user-data':
        file: ../Files/Scripts/user-data.txt
        variables:
             admin_password: { get_input: ADMIN_PASSWORD }
    '/opt/cisco/esc/mona/dmona.crt':
        data: { get_input: DMONA_CERT }
    '/opt/cisco/esc/mona/config/application-dmona.properties':
        file: ../Files/Scripts/application-dmona.template
        variables:
             monitoring.agent: true
```

```
 security_basic_enabled: { get_input: SECURITY_BASIC_ENABLED }
 security_user_name: { get_input: SECURITY_USER_NAME }
security_user_password: { get_input: SECURITY_USER_PASSWORD }
monitoring.agent.vim.mapping: true
```



第 8 章

仮想ネットワーク機能の修復

ETSI API を使用した仮想ネットワーク機能の修復

ESC は、ライフサイクル管理の一環として、障害が発生すると VNF を修復します。展開中に指定したリカバリポリシーがリカバリを制御します。ESCは、ポリシー主導型の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。『Cisco Elastic Services Controller User Guide』の「Configuring a Recovery Policy Using the Policy-driven Framework」を参照してください。

修復パラメータは、VNFを修復する通知をトリガーするためにモニタする動作を定義します。これらのパラメータは、ルールとともにVNFDの各コンピューティングノードのKPIセクションで設定されます。ルールでは、VNF を修復するためにこれらの KPI 条件の結果として実行されるアクション（トリガーされるイベントを含む）を定義します。

ESC ETSI は、次の 2 つのセクションを使用してモニタリングを設定します。

- kpi_data：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。
- admin_rules：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。

例：

```
vdu1:
  type: cisco.nodes.nfv.Vdu.Compute
  properties:
    name: Example VDU1
    description: Example VDU
    ...
    kpi_data:
      VM_ALIVE-1:
        event_name: 'VM_ALIVE-1'
        metric_value: 1
        metric_cond: 'GT'
        metric_type: 'UINT32'
        metric_occurrences_true: 1
```

```
        metric_occurrences_false: 30
        metric_collector:
          type: 'ICMPPing'
          nicid: 1
          poll_frequency: 10
          polling_unit: 'seconds'
          continuous_alarm: false
    admin_rules:
      VM_ALIVE-1:
        event_name: 'VM_ALIVE-1'
        action:
          - 'ALWAYS log'
          - 'FALSE recover autohealing'
          - 'TRUE esc_vm_alive_notification'
    ...
```

この例は、デフォルトのKPIと、ESCでの展開を完了するために必要なサービスアライブ通知をサポートするルールを示しています。VNFD で公開される KPI、ルール、および基盤となるデータモデルの詳細については、『Cisco Elastic Services Controller User Guide』の「KPIs、Rules and Metrics」を参照してください。

インスタンスに注意が必要なことを示すイベントを受信した場合、タイマーが期限切れになった、または手動のリカバリ要求を受信した場合のリカバリには、3 種類のアクションがあります。修復のワークフローは次のようになります。

- REBOOT_THEN_REDEPLOY：最初に、影響を受けたVNFCの再起動を試みます。これが失敗した場合、影響を受けた VNFC の再展開（同じホスト上で）を試みます
- REBOOT_ONLY：VM の再起動のみを試みます
- REDEPLOY_ONLY：VM の再展開のみを試みます

リカバリポリシーは VNF レベルで設定され、そこに含まれる VNFC ごとに適用されます。モニタリングエージェントが各VNFCをモニタし、リカバリ状況になると、メッセージがアラームに変換され、登録されたコンシューマ（NFVO または Element Manager）に送信されます。

VNF インスタンスで自動修復が有効になっている場合、ESC は展開時に設定されたリカバリポリシーに基づいて VNF のリカバリを自動的に試みます。これは、VNFD で設定、またはインスタンス化の前に VNF インスタンスにおいて変更できます。

VNF のリカバリは、影響を受けた VNFC に対するアクションを要求することです。初期展開操作がタイムアウトした後、ESCが定義されたポリシーを使用してサービスを回復できない場合、サービスが展開に失敗すると、ライフサイクル管理操作は失敗します。

自動修復フラグ（*isAutohealEnable*）VNF インスタンスリソースを変更するには、仮想ネットワーク機能の変更 （42 ページ）を参照してください。

自動修復が有効でない場合、アラームのみがすべてのサブスクライバにディスパッチされます。サブスクライバは手動の HealVnfRequest を開始できます。データ構造は、あらゆる VNF 固有のアクションに使用できます。必須パラメータはありません。

SOL003 の例：

```
Request Payload (ETSI data structure: HealVNFRequest)
POST /vnf_instances/{vnfInstanceId}/heal
{
```

```
    "cause": "b9909dde-e21e-45ec-9cc0-9e9ae413eee0",
}
```

SOL002 の例：

```
POST /vnf_instance/{vnfInstanceId}/heal
 {
   "vnfcInstanceId": ["b9909dde-e21e-45ec-9cc0-9e9ae413eee0"],
   "cause": "b9909dde-e21e-45ec-9cc0-9e9ae413eee0",
   "healScript": "REBOOT_ONLY"
 }
```

healScript は有効なリカバリポリシー名の列挙として実装されます。これにより、展開データモデルで設定されたポリシーを上書きできます。vnfcInstanceId のリストは、必要な VNFC が影響を受けることを許可しますが、このリストがない場合、要求は VNF 全体に適用されます。

追加のパラメータを使用して、展開時に設定されたポリシーに関係なく、上書きするリカバリポリシーを指定できます。

リカバリポリシーは、追加のパラメータを使用して VNFC レベルで指定できます。これにより、VNF レベルで設定された値が上書きされます。リカバリポリシーが VNFC レベルで指定されていない場合、ESC は VNF レベルのリカバリポリシーからプロパティを継承します。

オプションの追加パラメータが cisco.datatypes.nfv.VnfcAdditionalConfigurableProperties データタイプに追加され、VNFC レベルのリカバリをサポートします。

```
cisco.datatypes.nfv.VnfcAdditionalConfigurableProperties:
  derived_from: tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties
  properties:
    ...
    is_vnfc_autoheal_enabled:
      type: boolean
      description: It permits to enable (TRUE)/disable (FALSE) the auto-healing
functionality. If the properties is not present for configuring, then VNF-level property
 is used instead
      required: false
    recovery_action:
      type: string
      required: false
      constraints:
        - valid_values: [ REBOOT_THEN_REDEPLOY, REDEPLOY_ONLY, REBOOT_ONLY ]
```

モニタリングの詳細については、ETSI API を使用した仮想ネットワーク機能のモニタリング（47 ページ）を参照してください

修復中の既存の展開の更新

展開が正常に作成されたら、その中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、実行中の展開で実行できます。リソースは、リカバリプロセスの一環として更新されます。

修復ワークフロー中に、（ETSI NFV MANO API を介してプロビジョニングされた）既存の展開を更新できます。修復要求中に、既存のイメージと Day-0 パラメータが比較され、後続の修復要求の一部として提供される新しいパラメータに更新されます。

ヒーリングワークフローでは、次のことが可能です。

- 展開モデルを新しいイメージと Day-0 設定で更新する
- アップグレードされたイメージによる修復時に、新規または既存の設定データを VNFC に再適用する

（注） 変更が VIM で直接実行されない場合、データモデルの更新後に VNF を再展開する必要があります。

HealVnfRequest を介して新しい *additionalParams* を指定した後、（NFVO からの）付与応答も新しいイメージまたは新しい additionalParams を指定する場合、これもサービス更新をトリガーします。

展開を再展開の一環として移動させる必要があると NFVO が判断した場合、付与はリソースの新しい配置を反映するための新しい *zoneId* を提供します。

リカバリアクションは、サービスの更新が完了した後に実行されます。再展開の場合は、最新の展開モデルを考慮して、展開された更新が元に戻されないようにします。

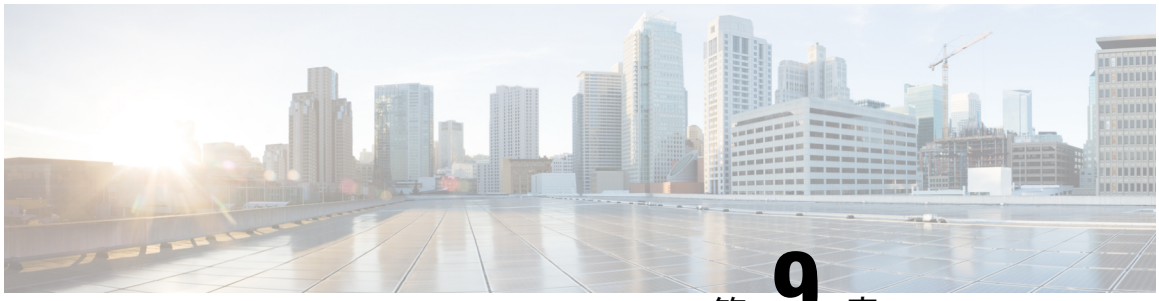
次の例は、新しい *additionalParams* や新しい *vimSoftwareImageId* でサービス更新をトリガーするために、NFVO が付与に返す詳細を示しています。

例：

```
{
  "headers" : {
    "Content-Type" : [ "application/json" ],
    "Location" : [
"http://{nfvoApiRoot}/sol003/default/grant/v1/grants/38ba2103-dab3-450e-992b-ee85aad6c899"
 ],
    "Content-Length" : [ "22935" ],
  },
  "body" : {
    "id" : "38ba2103-dab3-450e-992b-ee85aad6c899",
    "vnfInstanceId" : "6aaf527c-0093-49c3-ba2e-49fc6d8a4f71",
    "vnfLcmOpOccId" : "cdc5d9b3-81a0-400b-a4d9-97d1b3e117d9",
    "_links" : {
      "self" : {
        "href" :
"http://{nfvoApiRoot}/sol003default/grant/v1/grants/38ba2103-dab3-450e-992b-ee85aad6c899"

      },
      "vnfLcmOpOcc" : {
        "href" :
"https://{vnfmApiRoot}/vnflcm/v1/vnf_lcm_op_occs/cdc5d9b3-81a0-400b-a4d9-97d1b3e117d9"
      },
      "vnfInstance" : {
        "href" :
"https://{vnfmApiRoot}/vnflcm/v1/vnf_instances/6aaf527c-0093-49c3-ba2e-49fc6d8a4f71"
      }
```

```
    },
    "vimConnections" : [ {
      "id" : "myVimConnection",
      "vimType" : "OPENSTACK_V3",
      "vimId" : "595b0bc2-8dad-4087-abdf-ebe3b0b14d96",
      "interfaceInfo" : {
        "endpoint" : "https://{vimApiRoot}/v3"
      },
      "accessInfo" : {
        "password" : "********",
        "project" : "cisco",
        "projectDomain" : "demo",
        "region" : "RegionOne",
        "userDomain" : "demo",
        "username" : "********"
      }
    } ],
    "zones" : [{
      "id" : "1773873a-ab15-4a7b-b024-bc338425ed24",
      "zoneId" : "nova"
    },{
      "id" : "1773873a-ab15-4a7b-b024-bc555555ed55",
      "zoneId" : "nova2"
    }],
    "addResources" : [{
      "resourceDefinitionId" : "res-a6252dbf-b418-4f88-b8a9-14d8f3942938",
      "vimConnectionId" : "myVimConnection",
      "zoneId" : "1773873a-ab15-4a7b-b024-bc555555ed55"
    }],
    "vimAssets" : {
      "softwareImages" : [ {
        "vnfdSoftwareImageId" : "s3",
        "vimSoftwareImageId" : "3a609da7-e2b2-4e27-91b6-7bcabe902820",
        "vimConnectionId" : "myVimConnection"
      }, {
        "vnfdSoftwareImageId" : "s4",
        "vimSoftwareImageId" : "3a609da7-e2b2-4e27-91b6-7bcabe902820",
        "vimConnectionId" : "myVimConnection"
      } ]
    }
  },
  "additionalParams": [
    ...
    /* changed additionalParams */
    "CF_VIP_ADDR": "10.123.23.4",
    "SF_VIP_ADDR": "10.123.24.4",
    ...
  ],
  "statusCode" : "CREATED",
  "statusCodeValue" : 201
}
```

修復の詳細については、ETSI API を使用した仮想ネットワーク機能の修復 （57 ページ）を参照してください。



第 9 章

仮想ネットワーク機能のスケーリング

ETSI API を使用した仮想ネットワーク機能のスケーリング

ESC の主な利点の 1 つは、サービスを柔軟に拡張できることです。これにより、VNF 内で特定のロールまたはアスペクトを実行する VNFC が、要求を処理し、高い需要を満たすためにスケールアウトしたり、使用率が低い場合にスケールインしたりできます。このアスペクトは、複数の VNFC に広がる場合があります。

スケーリング要求は手動でも自動でもかまいません。スケーリングを実現するためのさまざまなアプローチについて、以下で詳しく説明します。

これらの概念と仕様の詳細については、*ETSI GS NFV-SOL 003* の Annex B を参照してください。

REST および NETCONF API を使用した VNF のスケーリングについては、『*Cisco Elastic Services Controller User Guide*』を参照してください。

拡張性

VNF のスケーリング要求は、VnfInstance リソースをクエリするときに instantiatedVnfInfo の一部として見つかる属性である *scaleStatus* を使用します。この属性は、VNF の各アスペクトの現在のスケールレベルを示します。次に例を示します。

```
"scaleInfo": [
  {
    "aspectId": "webserver", "scaleLevel": "4"
  },
  {
    "aspectId": "processing", "scaleLevel": "2"
  }
]
```

これは VNF のスケーリング要求の開始点を形成します。これにより、1 つのアスペクトを、VNF のその寸法において、現在の *scaleLevel* に対して水平方向にスケーリング（VNFC を追加

または削除）できます。アスペクトのスケーリング操作は、そのアスペクトをサポートする各 VNFC に適用されます。

（注） 現在の仕様では、垂直スケーリング（既存の VNFC インスタンスへのリソースの追加/削除）はサポートされていません。

要求ペイロード（ETSI データ構造：ScaleVNFRequest）

```
{
    "type": "SCALE_OUT",
    "aspectId": "processing",
    "numberOfSteps": 1,
    "additionalParams": {}
}
```

上記のペイロードにより、上記の *scaleStatus* の例は更新され、scaleLevel 3 にスケールアウトするために必要なこの手順において、VNFC の数が追加されます。

```
"scaleInfo": [
  {
    "aspectId": "webserver", "scaleLevel": "4"
  },
  {
    "aspectId": "processing", "scaleLevel": "3"
  }
]
```

スケーリング手順およびスケーリングをサポートするその他の関連ポリシーについては、「スケーリングの VNFD ポリシー」を参照してください。

レベルへのスケーリング

Scale VNF が提供する相対的なスケーリングではなく、VNF をレベルにスケーリングする要求は、求められる絶対的なスケーリング結果を指定します。その結果、一部のアスペクトはスケールアウトされ、その他のアスペクトはスケールインされます。このオプションは、スケーリングで必要な 2 つのアプローチのうちの 1 つを使用します。

- インスタンス化レベル
- スケールレベル

これらは相互に排他的であり、1 つの要求で複数のアスペクトをスケーリングできます。

インスタンス化レベル

インスタンス化レベルは各アスペクトに事前に定義されたサイズで、各レベルには各アスペクトに関連付けられたスケールレベルがあります。これ以上の細分性は提供されないため、VNF 全体（すなわちすべてのアスペクト）が、要求されるインスタンス化レベルに従ってスケーリングされます。

例：

要求ペイロード（ETSI データ構造：ScaleVNFToLevelRequest）

```
{
    "instantiationLevelId": "premium"
}
```

インスタンス化レベルの定義については、VNFD ポリシーを参照してください。

スケールレベル

スケールレベルもまた各アスペクトに事前定義されたサイズで、各アスペクトにはターゲット VNFC、定義された step_deltas（各スケーリングステップは均一ではない可能性があるため）、最大スケールレベルがあります。このオプションを定義するポリシーでは、ターゲットごとに異なるスケーリング結果を使用できます。

（注） スケールレベルは VM の数を表すものではありません。たとえば、scaleLevel=0 はターゲット VNFC 上のそのアスペクトのインスタンスの初期数（初期デルタ）を意味し、scaleLevel=1 は初期デルタに、そのアスペクトと VNFC タプルで定義した最初のスケーリングステップを加えたものです。

要求ペイロード（ETSI データ構造：ScaleVNFToLevelRequest）

```
{
    "scaleInfo": [
      {
            "aspectId": "processing",
            "scaleLevel": "2"
      },
      {
            "aspectId": "webserver",
            "scaleLevel": "3"
      }
    ]
}
```

スケールレベルの定義については、「スケーリングの VNFD ポリシー」を参照してください。

スケーリングの VNFD ポリシー

VNF の全体的なスケーリング動作を作るポリシーは多数あります。これらのポリシーは、上記のさまざまなスケーリングアプローチをサポートします。最初のポリシーは、スケーリングされる（またはスケーリングされない）アスペクトを定義します。

```
policies:
    - scaling_aspects:
        type: tosca.policies.nfv.ScalingAspects
        properties:
          aspects:
            webserver:
              name: 'webserver'
              description: 'The webserver cluster.'
              max_scale_level: 5
              step_deltas:
```

```
            - delta_1
        processing:
          name: 'processing'
          description: 'An example processing function'
          max_scale_level: 3
          step_deltas:
            - delta_1
            - delta_2
            - delta_1
        database:
          name: 'database'
          description: 'A test database'
          max_scale_level: 0
```

この例では、データベースアスペクトの max_scale_level が 0 であることがわかります。これはスケールアウトできないことを意味し、そのアスペクトのインスタンスが 0 であることを意味するわけではありません。理由については、以下のアルゴリズムを参照してください。Web サーバのアスペクトには 1 つの step_delta しかありません。つまり、すべてのスケーリングステップが均一であるのに対し、処理アスペクトにはスケーリングステップごとに異なる step_delta が指定されます。これは不均一スケーリングと呼ばれます。これはこの VNF のアスペクトの宣言にすぎず、これはスケーリング要求を受信したときに検証を実行するために使用されるポリシーの 1 つです。

次に、動作を制御するためにこれらを VNFC に適用する必要があります。

```
- db_initial_delta:
        type: tosca.policies.nfv.VduInitialDelta
        properties:
          initial_delta:
            number_of_instances: 1
        targets: [ vdu1 ]

    - ws_initial_delta:
        type: tosca.policies.nfv.VduInitialDelta
        properties:
          initial_delta:
            number_of_instances: 1
        targets: [ vdu2, vdu4 ]

    - pc_initial_delta:
        type: tosca.policies.nfv.VduInitialDelta
        properties:
          initial_delta:
            number_of_instances: 1
        targets: [ vdu3 ]

    - ws_scaling_aspect_deltas:
        type: tosca.policies.nfv.VduScalingAspectDeltas
        properties:
          aspect: webserver
          deltas:
            delta_1:
              number_of_instances: 1
        targets: [ vdu2, vdu4 ]

    - pc_scaling_aspect_deltas:
        type: tosca.policies.nfv.VduScalingAspectDeltas
        properties:
          aspect: processing
          deltas:
            delta_1:
              number_of_instances: 1
```

```
            delta_2:
              number_of_instances: 2
        targets: [ vdu2, vdu4 ]
```

上記の例では、VNFC がターゲットとして識別されています。アスペクトは VNFCS ごとに異なる動作の場合がありますが、ここでは示しません。スケーリング要求の検証と生成に使用される step_deltas の定義もここに示します（これらの手順は要求されるスケールレベルによって推測されます）。VNFC のインスタンスの最小数は常に 0 と仮定され、最大数は次のアルゴリズムによって計算されます。

initial_delta に max_scale_level までの逓増する各インスタンス数を足したもの。

これらのポリシーは、スケールレベルベースのスケーリングと見なされます。インスタンス化レベルに基づくスケーリングには、同様の構成が使用されます。

```
- instantiation_levels:
        type: tosca.policies.nfv.InstantiationLevels
        properties:
          levels:
            default:
              description: 'Default instantiation level'
              scale_info:
                database:
                  scale_level: 0
                webserver:
                  scale_level: 0
                processing:
                  scale_level: 0
            premium:
              description: 'Premium instantiation level'
              scale_info:
                database:
                  scale_level: 0
                webserver:
                  scale_level: 2
                processing:
                  scale_level: 3
          default_level: default
```

スケーリングアスペクトと同様に、インスタンス化レベルの定義の最初の部分は単なる宣言です。ここでは、各アスペクトはすでに宣言されている必要があり、その後、各アスペクトの scale_level はインスタンス化レベルで宣言されます。デフォルトのインスタンス化レベルは、他に何も指定されていない場合にも指定されます。各 VNFC の各 scale_level の意味は、VduInstantiationLevels ポリシーでさらに詳しく説明されています。次に例を示します。

```
- ws_instantiation_levels:
        type: tosca.policies.nfv.VduInstantiationLevels
        properties:
          levels:
            default:
              number_of_instances: 1
        targets: [ vdu2, vdu4 ]
```

したがって、これらのポリシーは、デフォルトのインスタンス化レベルが「default」であり、その結果 Web サーバのアスペクトが、scale_level 0（1 VNFC インスタンス）でインスタンス化されることを示します。

複数の IP アドレスへの依存

スタティック IP アドレス

VNFC に静的 IP アドレスが設定された接続ポイントがある場合、新たにスピンアップされた VNFC インスタンスの接続ポイントに割り当てる IP アドレスがないため、VNFC をスケーリングできません。代わりに、追加の静的 IP アドレスのプールを指定できます。これは ETSI の拡張仕様です。

次の例では、IP アドレスのリスト、IP 範囲、またはネットマスクを持つゲートウェイ（1 つまたは複数の組み合わせを指定可能）を使用して、静的 IP プールを作成する方法を説明します。

```
vdu2:
  type: cisco.nodes.nfv.Vdu.Compute
  properties:
    name: 'Webserver1'
    description: 'Webserver VNFC'
    vdu_profile:
      min_number_of_instances: 1
      max_number_of_instances: 6
      static_ip_address_pool:
        network: network1
        ip_addresses:
          - ip_address: 192.168.100.0
          - ip_address: 192.168.100.1
          - ip_address: 192.168.100.2
          - ip_address: 192.168.100.3
        ip_address_range:
          - start: 172.16.233.10
            end: 172.16.233.15
          - start: 172.16.233.20
            end: 172.16.233.25
        gateway: 172.10.11.0
        netmask: 255.255.255.0
```

静的 IP アドレスの接続ポイントを持つ、スケールアウトされた VNFC インスタンスがネットワークに割り当てられます。これは、スケールアウトされたインスタンスを展開するときに使用する IP アドレスプールを識別するためのキーです。静的 IP は、展開時に InstantiateVnfRequest の入力の一部として指定されます。VNF のインスタンス化の詳細については、「VNFのインスタンス化」を参照してください。

入力は、VNFD を介して additionalParams の一部として提供されます。

デイゼロ設定

VNF を展開後、展開サービスの VNFC インスタンスに day 0 の変数が設定されます。多くの場合、day 0 の設定値は一定です。それ以外の場合、day 0 のパラメータに指定される値のリソースプールがあり、新しい VNFC インスタンスに新しい値を割り当てられます。

VNFD の vendor_section 内の Day 0 の設定：

```
vdu3:
  type: cisco.nodes.nfv.Vdu.Compute
  properties:
    name: 'Processing1'
    description: 'Processing VNFC'
    vdu_profile:
```

```
          min_number_of_instances: 1
          max_number_of_instances: 5
        vendor_section:
          cisco_esc:
            config_data:
              '/tmp/OSRESTTestETSIDay0_Inline_data.cfg':
                data: |
                  NODE_NAME $NODE_NAME
                  NUM_OF_CPU $NUM_OF_CPU
                  MEM_SIZE $MEM_SIZE
                  PROXY_ADDRS $PROXY_ADDRS
                  SPECIAL_CHARS $SPECIAL_CHARS
                variables:
                  NODE_NAME: vdu_node_1
                  NUM_OF_CPU: 1
                  MEM_SIZE: 1GB
                  PROXY_ADDRS: ["1.1.1.1", "1.1.2.1", "1.1.3.1", "1.1.4.1", "1.1.5.1",
"1.1.6.1", "1.1.7.1"]
                  SPECIAL_CHARS: '`~!@#$%^&*()-_=+[{]}|;:<.>/?'
```

上記の例では、day 0 の設定はインラインで指定されており、速度変数はターゲット設定で定義されています。これらの各変数は、1 つ以上の値を持つ変数によってサポートされます。$PROXY_ADDRS 変数の複数の値をサポートするため、値のリストが提供されます。これらの値は、VNFC の新しいインスタンスの変数を後続で使用する際に事前入力するために使用されます。

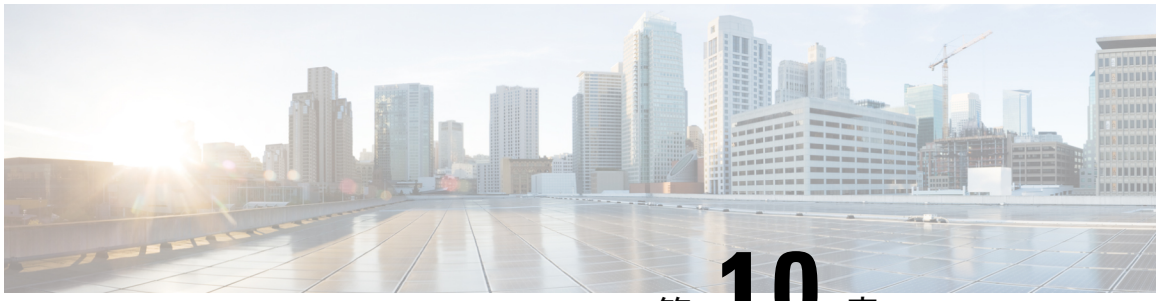
展開モデルの day 0 の設定の詳細については、『*Cisco Elastic Services Controller User Guide*』の「Day Zero Configuration」を参照してください。

VNF の自動スケーリング

VNFD で定義される KPI、ルール、およびアクションによって、スケーリングを考慮する必要がある条件が決まります。詳細については、「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。スケーリングポリシーは、許可されるスケーリング境界を制御するいくつかのポリシータイプを使用して、VNFD でも定義されます。次に、これらのポリシー項目について説明します。

展開後、ESC は各 VNFC をモニタするために、KPI を使用してモニタリングエージェント（これは集中管理型インスタンスまたは分散型インスタンスの場合があります）を設定します。KPI がしきい値に達すると、スケーリングワークフローが開始されます。定義されたアクションに基づいて、ESC はスケールインまたはスケールアウトを実行し、適切な通知とイベントログを生成します。これは、ログやオンボードスクリプトなど、指定できる一部の組み込み関数に従います。

ESC は、サブスクライブされたコンシューマに適切な通知を送信します。この時点で、ESC は *isAutoscaleEnabled* フラグについて、VNF インスタンスリソースに問い合わせます（これは最初に VNFD の値によって設定されますが、作成後に変更できます）。このフラグが true に設定されている場合、ESC はスケーリングワークフローを呼び出します（*ScaleVnfToLevelRequest* を使用して問い合わせ、1 つの要求で複数のアスペクトのスケーリングを要求します）。*isAutoscaleEnabled* が false に設定されている場合、上記の要求を使用して、制御は目的のアクションをトリガーするために、NFVO や EM などの外部システムを使用します。



第 10 章

エラー処理手順

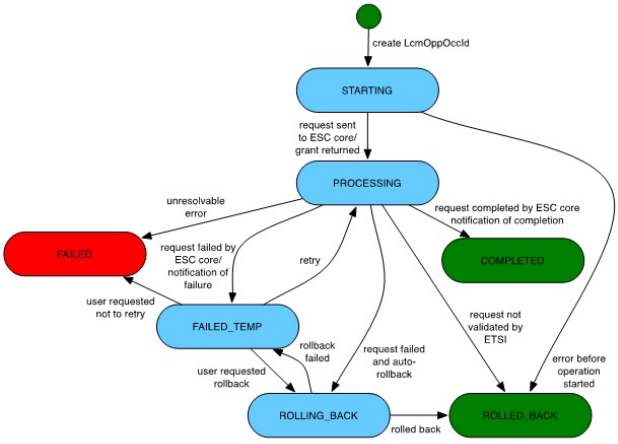
VNF ライフサイクル管理エラーの処理手順

ETSI は、すべての ETSI VNF ライフサイクル管理（LCM）操作に対して、次のエラー処理手順を呼び出します。

- Retry
- ロールバック
- 失敗
- キャンセル

次の図は、VNF ライフサイクル管理の運用オカレンスの遷移状態を表しています。

図 *2 : VNF* ライフサイクル管理の遷移状態



（注） *vnfLcmOpOccId* は、要求の詳細を取得するためのプライマリキーである URI にエンコードされます。

LCM 操作が FAILED_TEMP 以外の状態にある場合、再試行、ロールバック、および失敗の要求は拒否されます。このエラーは HTTP コード 409 を返します。

再試行、ロールバック、失敗、キャンセルの各要求は、特定の VNF の特定の VNF LCM 操作ではサポートされません。このエラーは HTTP コード 404 を返します。

vnfLcmOpOccId が ETSI データベースに存在しない場合、エラーが発生します。このエラーは HTTP コード 404 を返します。

Retry

再試行要求は、LCM 操作が成功する可能性がある場合に適用できます。この操作は、再試行要求の FAILED_TEMP 状態である必要があります（前提条件）。操作が FAILED_TEMP 状態である限り、複数の再試行要求を送信できます。

前提条件	FAILED_TEMP 状態
要求	`POST {api_root}/vnf_lcm_op_occs/{vnfLcmOpOccId}/retry()`
事後条件	PROCESSING 状態

再試行に成功すると、ESC は START または PROCESSING 通知を送信します。再試行要求が失敗すると、ESC は詳細を含む通知を NFVO に送信します。

ロールバック

ロールバック要求は、再試行要求の後でも操作が成功しない場合に実行されます。

rollback_required フラグを true に設定します。これが true に設定されない場合、ロールバックは実行されません。

前提条件	FAILED_TEMP 状態
要求	`POST {api_root}/vnf_lcm_op_occs/{vnfLcmOpOccId}/rollback()`
事後条件	ROLLED_BACK

ロールバックが成功すると、LCM 操作がロールバックされます。ロールバック要求が失敗すると、LCM 操作は failed_temp 状態に戻ります。

失敗

LCM 操作に再試行要求またはクリーンアップが必要ない場合、失敗した要求は後続の要求のためにリソースを解放します。

rollback_required フラグが true に設定されている場合、失敗要求は実行できません。

前提条件	FAILED_TEMP 状態
要求	`POST {api_root}/vnf_lcm_op_occs/{vnfLcmOpOccId}/fail()`
事後条件	FAILED 状態

この要求が正常に実行されると、LCM 操作は FAILED 状態になります。

キャンセル

操作が STARTING 状態の場合、キャンセル要求が可能です。

（注） 現在キャンセル要求は、インスタンス化の STARTING または PROCESSING 状態で可能ですが、他のすべての LCM 操作では STARTING のみです。

前提条件	STARTING 状態
要求	`POST {api_root}/vnf_lcm_op_occs/{vnfLcmOpOccId}/cancel(CancelMode)`
事後条件	ROLLED_BACK

キャンセル要求は強制的です。

（注） ETSI は、starting 状態でのみ LCM 操作のキャンセルをサポートします。処理中またはロールバック状態の LCM 操作のキャンセル要求は現在サポートされていません。

例 JSON ペイロード（CancelMode）：

```
{
  "cancelMode": "FORCEFUL",
  "action": "cancel"
}
```

VnfLcmOpOcc の *IsCancelPending* 属性を true に設定します。これにより、処理要求が停止し、LCM 操作が ROLLED_BACK 状態に移行します。

ETSI VNF ライフサイクル操作のエラー処理手順

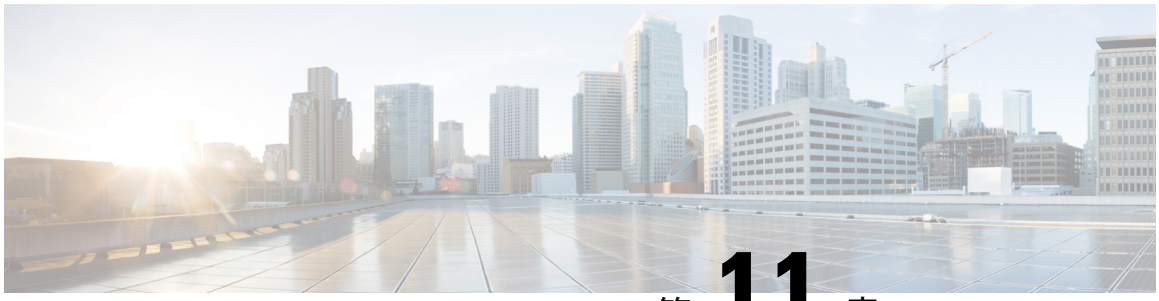
VNF インスタンスの LCM 操作が失敗すると、操作はステートマシンに従って FAILED_TEMP 状態に移行します。目的の操作を完了するには、再試行またはロールバック要求を実行する必要があります。

- VNF ID の作成に失敗した場合、それ以上のアクションは不要です。ロールバック要求はサポートされていません。

- VNF のインスタンス化が失敗すると、ESC は要求を終了し、新しいインスタンス化要求を送信します。
- VNF の操作が失敗した場合、それ以上のアクションは必要ありません。
- VNF の終了に失敗した場合は、ロールバックがサポートされていないため、操作を再試行する必要があります。
- VNF の削除操作が失敗した場合、それ以上のアクションは必要ありません。ロールバック要求はサポートされていません。

（注） エラー処理要求は、動作中の VNF ライフサイクル操作には影響を与えません。

VNF ライフサイクル操作の詳細については、VNF ライフサイクル操作 （26 ページ）を参照してください。



第 11 章

ETSI LCM 操作のアラームと通知

ETSI アラーム

ESC は、NFVO にアラームと通知を提供します。NFVO はこれらのアラームと通知に登録し、要求を ESC に送信する必要があります。

NFVO は、次の方法でアラームに関する情報を受信できます。

すべてのアラームをクエリする

NFVO は、アラームリソースからすべてのアラームのリストを取得できます。

メソッドタイプ：

```
GET
```

VNFM エンドポイント：

```
/vnffm/v1/alarms
```

HTTP 要求ヘッダー：

```
Accept:application/json
```

たとえば、イベントタイプが ENVIRONMENTAL_ALARM のすべてのアラームをクエリする場合

メソッドタイプ：

```
GET
```

VNFM エンドポイント：

```
http://localhost:8250/vnffm/v1/alarms?eventType="ENVIRONMENTAL_ALARM"
```

HTTP 要求ヘッダー：

```
Accept:application/json
```

複数アラームのクエリ中に、NFVOはURIクエリパラメータを使用して結果をフィルタ処理できます。アラームの URI クエリでは、次の属性名がサポートされています。

- id
- managedObjectId
- rootCauseFaultyResource.faultyResourceType
- eventType
- perceivedSeverity
- probableCause

（注） URI クエリパラメータは、複数のアラームのクエリ専用です。

個々のアラームのクエリ

NFVO は、*alarmId* リソースから特定のアラームをクエリできます。

メソッドタイプ：

```
GET
```

VNFM エンドポイント

```
/vnffm/v1/alarms/{alarmId}
```

HTTP 要求ヘッダー：

```
Accept:application/json
```

個々のアラームの変更

アラームを変更するには、NFVO が *AlarmModifications* リソースに PATCH 要求を送信する必要があります。

メソッドタイプ：

```
PATCH
```

VNFM エンドポイント：

```
/vnffm/v1/alarms/{alarmId}
```

HTTP 要求ヘッダー：

```
Content-Type: application/merge-patch+json
If-Match: ETag value
```

（注） **If-Match:** はオプションです。指定した場合、その値は VNF に保存された ETag 値に対して検証されます（1 つの VNF クエリから返されます）。

サポートされる属性はackStateで、サポートされる属性値はACKNOWLEDGEです。他のすべての変更ペイロードは拒否されます。

VNF 障害および負荷アラーム

次のアラームは、ETSI VNF 障害および負荷通知用に作成されます。

- 障害アラーム：ESC は、VFND の VM_ALIVE KPI 設定に基づいて VNF 内のコンピューティングリソースの 1 つが到達不能になると、障害アラームを生成します。詳細については、「VNF の ETSI 障害および負荷の通知 」を参照してください。

  例：

  メソッドタイプ

  ```
  POST
  ```

  VNFM エンドポイント

  ```
  /vnffm/v1/extension/alarms
  ```

  HTTP 要求ヘッダー

  ```
  Content-Type:application/json
  ```

  要求ペイロード：

  ```
  {
    "externalAlarmId" : "26bf1e3d-cefa-4f59-88ea-210a29358a5c", #generated value
    "alarmSource" : "MONA", #hard-coded
    "managedObjectId" : "08733ef2-319b-46ce-9d8d-95730306bd1a", #external_deployment_id

    "rootCauseFaultyResource" : "chrimann-dep_g1_0_212da327-0573-421b-ae37-057f6b1a6aef",
   #vm_name
    "alarmRaisedTime" : "$timestamp", #generated value
    "ackState" : "UNACKNOWLEDGED", #hard-coded
    "perceivedSeverity" : "CRITICAL", #hard-coded
    "eventTime" : "2018-05-08T00:59:32.571+00:00", #do we have the eventTime?
    "eventType" : "EQUIPMENT_ALARM", #hard-coded
    "faultType" : "COMPUTE", #hard-coded
    "probableCause" : "VM_MANUAL_RECOVERY_NEEDED", #event_name
    "isRootCause" : "TRUE", #hard-coded
    "links" : {
      "objectInstance" :
  "{http_scheme}://{api_root}/vnflcm/v1/vnf_instances/08733ef2-319b-46ce-9d8d-95730306bd1a"

    }
  }
  ```

- 負荷アラーム：ESCは、VFNDの関連KPI設定に基づいて、VNF内のコンピューティングリソースの1つが過負荷または過小負荷になると、負荷アラームを生成します。ESCは、NFVOから通知を受信した後にこれらのアラームを作成します。詳細については、「VNF の ETSI 障害および負荷の通知 」を参照してください。

  例：

  メソッドタイプ

  ```
  POST
  ```

  VNFM エンドポイント

```
/vnffm/v1/extension/alarms
```

HTTP 要求ヘッダー

```
Content-Type:application/json
```

要求ペイロード

アラーム拡張

ETSI は、サードパーティツールとやり取りするアラームの拡張機能を提供します。アラームを作成するには、POST 要求を送信する必要があります。

メソッドタイプ

```
POST
```

VNFM エンドポイント

```
/vnffm/v1/extension/alarms
```

HTTP 要求ヘッダー

```
Content-Type:application/json
```

要求ペイロード

```
[admin@davwebst-esc-4-2-0-49-keep ETSI]$ cat CreateAlarm.json
{
    "id": "alm87032",
    "externalAlarmId": "ext-id-xx11214",
    "managedObjectId": "930fb087-c1b9-4660-bec8-2a8d97dc1df5",
    "rootCauseFaultyResource": {
        "id": "fres7629",
        "faultyResource": {
            "resourceId": "res7727"
        },
        "faultyResourceType": "NETWORK"
    },
    "alarmRaisedTime": "2018-05-30T13:55:15.645000+00",
    "ackState": "UNACKNOWLEDGED",
    "perceivedSeverity": "MAJOR",
    "eventTime": "2018-05-30T13:55:15.645000+00",
    "eventType": "ENVIRONMENTAL_ALARM",
    "probableCause": "Server room overheading",
    "isRootCause": "false"
}
```

通知への登録

NFVO は、ESC からの障害管理に関連した ETSI 通知に登録できます。

サブスクリプションの作成

NFVO は、通知に登録するための POST 要求を送信します。

メソッドタイプ：

```
POST
```

VNFM エンドポイント：

```
/vnffm/v1/subscriptions
```

応答ペイロード：

```
{
  "filter" : {
    "notificationTypes" : [
      "AlarmNotification",
      "AlarmClearedNotification",
      "AlarmListRebuiltNotification"
    ],
    "perceivedSeverities" : [
      "CRITICAL",
      "MAJOR"
    ]
  },
  "callbackUri" : "https://nfvo.endpoint.listener",
  "authentication" : {
    "authType" : "BASIC",
    "paramsBasic" : {
      "userName" : "admin",
      "password" : "pass123"
    }
  }
}
```

これにより、新しい登録リソースと新しい ID が作成されます。必須パラメータは callbackUri だけです。その他はすべてオプションです。GET 要求を送信することで、callbackuri が有効で到達可能かどうかを確認できます。

すべてのサブスクリプションのクエリ

NFVO は、サブスクリプションリソースに GET 要求を送信することで、そのサブスクリプションに関する情報をクエリできます。

メソッドタイプ：

```
GET
```

VNFM エンドポイント：

```
/vnffm/v1/subscriptions
```

HTTP 要求ヘッダー：

```
Accept:application/json
```

たとえば、callbackUri が以下の場合、すべてのアラートサブスクリプションをクエリするには

```
http://10.10.1.44:9202/alerts/subscriptions/callback
```

```
GET
```

VNFM エンドポイント

```
http://localhost:8250/vnffm/v1/subscriptions?callbackUri="http://10.10.1.44:9202/alerts/subscriptions/callback"
```

HTTP 要求ヘッダー

```
Accept:application/json
```

NFVOはURIクエリパラメータを使用して、結果をフィルタ処理できます。サブスクリプションの URI クエリでは、次の属性名がサポートされています。

- id
- filter
- callbackUri

（注） URI クエリパラメータは、複数のサブスクリプションのクエリ専用です。

個々のサブスクリプションのクエリ

個々のサブスクリプションをクエリするには、サブスクリプション ID を知っている必要があります。

メソッドタイプ：

```
GET
```

VNFM エンドポイント：

```
/vnffm/v1/subscriptions/{subscriptionId}
```

HTTP 要求ヘッダー：

```
Accept:application/json
```

サブスクリプションの削除

NFVO が必要としていないサブスクリプションを削除できます。個々のサブスクリプションに削除要求を送信します。

メソッドタイプ：

```
DELETE
```

VNFM エンドポイント：

```
/vnffm/v1/subscriptions/{subscriptionId}
```

HTTP 要求ヘッダー：

```
http://localhost:8250/vnffm/v1/subscriptions/682791f8-34ad-487e-811a-553036bf49b2
```

VNF の ETSI 障害および負荷の通知

ESC は次の通知を生成します。

- **VM の障害**

  展開された VNF 内の VM に障害が発生すると、NFVO は ESC から障害通知を受信します。通知を受信すると、アラームが生成されます。アラームの詳細については、ETSI アラーム （75 ページ）を参照してください。

NFVO は通知のために ESC に登録する必要があります。

例：

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_event xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <deployment_name>sample-dep</deployment_name>
  <event_name>MY_VM_UNDERLOADED</event_name>
  <event_type>VM_UNDERLOADED</event_type>
  <external_deployment_id>e911eecf-5f3f-456c-9c80-d99aca2416da</external_deployment_id>

  <external_tenant_id>etsi_tenant</external_tenant_id>
  <internal_deployment_id>99f7629f-98d3-40f5-ad68-7addcfe07006</internal_deployment_id>

  <internal_tenant_id>etsi_tenant</internal_tenant_id>
  <vm_source>

<generated_vm_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</generated_vm_name>

    <interfaces>
      <addresses>
        <address>
          <address_id>0</address_id>
          <gateway>172.16.0.1</gateway>
          <ip_address>172.16.0.0</ip_address>
          <dhcp_enabled>true</dhcp_enabled>
          <prefix>20</prefix>
          <subnet>365a0884-fdb3-424c-afe9-2deb3b39baae</subnet>
        </address>
      </addresses>
      <network_uuid>c7fafeca-aa53-4349-9b60-1f4b92605420</network_uuid>
      <mac_address>fa:16:3e:38:1d:6c</mac_address>
      <nic_id>0</nic_id>
      <port_forwarding/>
      <port_uuid>0aeb9585-5190-4f3b-b1aa-495e09c56b7d</port_uuid>
      <security_groups/>
      <subnet_uuid>none</subnet_uuid>
      <type>virtual</type>

<vim_interface_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</vim_interface_name>

    </interfaces>
    <vim_id>default_openstack_vim</vim_id>
    <vim_project>admin</vim_project>
    <vim_project_id>c12f013306d849e5b1bbf257c54d5891</vim_project_id>
    <host_uuid>6b8cf361c5ff08a5a886e26f591b8087dadcf2d2b34fb3b5d2772a8d</host_uuid>

    <host_name>my-server</host_name>
    <vm_uuid>9fea3fe7-9417-4734-b962-b24340941ef3</vm_uuid>
    <vm_group_name>vm1</vm_group_name>
    <vm_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</vm_name>
  </vm_source>
</esc_event>
```

- **VM のオーバーロードとアンダーロード**

  同様に、NFVO は VM のオーバーロードまたはアンダーロードの通知を受信します。

  スケーリングが自動的に有効になっていない場合、ESC は VM の状態に応じて通知を生成します。

  次に例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_event xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <deployment_name>sample-dep</deployment_name>
  <event_name>MY_VM_UNDERLOADED</event_name>
  <event_type>VM_UNDERLOADED</event_type>
  <external_deployment_id>e911eecf-5f3f-456c-9c80-d99aca2416da</external_deployment_id>

  <external_tenant_id>etsi_tenant</external_tenant_id>
  <internal_deployment_id>99f7629f-98d3-40f5-ad68-7addcfe07006</internal_deployment_id>

  <internal_tenant_id>etsi_tenant</internal_tenant_id>
  <vm_source>

<generated_vm_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</generated_vm_name>

    <interfaces>
      <addresses>
        <address>
          <address_id>0</address_id>
          <gateway>172.16.0.1</gateway>
          <ip_address>172.16.0.0</ip_address>
          <dhcp_enabled>true</dhcp_enabled>
          <prefix>20</prefix>
          <subnet>365a0884-fdb3-424c-afe9-2deb3b39baae</subnet>
        </address>
      </addresses>
      <network_uuid>c7fafeca-aa53-4349-9b60-1f4b92605420</network_uuid>
      <mac_address>fa:16:3e:38:1d:6c</mac_address>
      <nic_id>0</nic_id>
      <port_forwarding/>
      <port_uuid>0aeb9585-5190-4f3b-b1aa-495e09c56b7d</port_uuid>
      <security_groups/>
      <subnet_uuid>none</subnet_uuid>
      <type>virtual</type>

<vim_interface_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</vim_interface_name>

    </interfaces>
    <vim_id>default_openstack_vim</vim_id>
    <vim_project>admin</vim_project>
    <vim_project_id>c12f013306d849e5b1bbf257c54d5891</vim_project_id>
    <host_uuid>6b8cf361c5ff08a5a886e26f591b8087dadcf2d2b34fb3b5d2772a8d</host_uuid>

    <host_name>my-server</host_name>
    <vm_uuid>9fea3fe7-9417-4734-b962-b24340941ef3</vm_uuid>
    <vm_group_name>vm1</vm_group_name>
    <vm_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</vm_name>
  </vm_source>
</esc_event>
```

VM のアンダーロードの例：

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_event xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <deployment_name>sample-dep</deployment_name>
  <event_name>MY_VM_OVERLOADED</event_name>
  <event_type>VM_OVERLOADED</event_type>
  <external_deployment_id>e911eecf-5f3f-456c-9c80-d99aca2416da</external_deployment_id>

  <external_tenant_id>etsi_tenant</external_tenant_id>
  <internal_deployment_id>99f7629f-98d3-40f5-ad68-7addcfe07006</internal_deployment_id>

  <internal_tenant_id>etsi_tenant</internal_tenant_id>
  <vm_source>
```

```
<generated_vm_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</generated_vm_name>

    <interfaces>
      <addresses>
        <address>
          <address_id>0</address_id>
          <gateway>172.16.0.1</gateway>
          <ip_address>172.16.0.0</ip_address>
          <dhcp_enabled>true</dhcp_enabled>
          <prefix>20</prefix>
          <subnet>365a0884-fdb3-424c-afe9-2deb3b39baae</subnet>
        </address>
      </addresses>
      <network_uuid>c7fafeca-aa53-4349-9b60-1f4b92605420</network_uuid>
      <mac_address>fa:16:3e:38:1d:6c</mac_address>
      <nic_id>0</nic_id>
      <port_forwarding/>
      <port_uuid>0aeb9585-5190-4f3b-b1aa-495e09c56b7d</port_uuid>
      <security_groups/>
      <subnet_uuid>none</subnet_uuid>
      <type>virtual</type>

<vim_interface_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</vim_interface_name>

    </interfaces>
    <vim_id>default_openstack_vim</vim_id>
    <vim_project>admin</vim_project>
    <vim_project_id>c12f013306d849e5b1bbf257c54d5891</vim_project_id>
    <host_uuid>6b8cf361c5ff08a5a886e26f591b8087dadcf2d2b34fb3b5d2772a8d</host_uuid>

    <host_name>my-server</host_name>
    <vm_uuid>9fea3fe7-9417-4734-b962-b24340941ef3</vm_uuid>
    <vm_group_name>vm1</vm_group_name>
    <vm_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</vm_name>
  </vm_source>
</esc_event>
```

KPI 手順を使用した VNF の自動スケーリング

ESC は、KPI 手順を使用して VM を自動スケーリングできます。スケーリングワークフローは、VNF インスタンスがインスタンス化された状態のときに開始されます。NFVO は、VNF の *isAutoscaleEnabled* 設定可能プロパティを変更しながら、自動スケーリングを有効または無効にできます。

次に、ETSI 準拠の自動スケールをトリガーするイベントを示します。これには、*ScaleVnfToLevelRequest:* 機能の導入が必要です。

- **オーバーロードとアンダーロード**

  VM の状態が変化し、VM がオーバーロードまたはアンダーロードの場合、ESC はスケーリングが自動的に有効になっているかどうかを判断する通知を受け取ります。そうでない場合、ESC は VNF の状態を確認するために ETSI-VNFM コンポーネントへの通知を生成します。

  次の例は、ESC からのアンダーロード通知を示しています。

```
Headers:
  esc-status-code = 200
  esc-status-message = VM [sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de]
```

```
underloaded.
Body:
<?xml version="1.0" encoding="UTF-8"?>
<esc_event xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <deployment_name>sample-dep</deployment_name>
  <event_name>MY_VM_UNDERLOADED</event_name>
  <event_type>VM_UNDERLOADED</event_type>
  <external_deployment_id>e911eecf-5f3f-456c-9c80-d99aca2416da</external_deployment_id>

  <external_tenant_id>etsi_tenant</external_tenant_id>
  <internal_deployment_id>99f7629f-98d3-40f5-ad68-7addcfe07006</internal_deployment_id>

  <internal_tenant_id>etsi_tenant</internal_tenant_id>
  <vm_source>

<generated_vm_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</generated_vm_name>

    <interfaces>
      <addresses>
        <address>
          <address_id>0</address_id>
          <gateway>172.24.0.1</gateway>
          <ip_address>172.24.0.37</ip_address>
          <dhcp_enabled>true</dhcp_enabled>
          <prefix>20</prefix>
          <subnet>365a0884-fdb3-424c-afe9-2deb3b39baae</subnet>
        </address>
      </addresses>
      <network_uuid>c7fafeca-aa53-4349-9b60-1f4b92605420</network_uuid>
      <mac_address>fa:16:3e:38:1d:6c</mac_address>
      <nic_id>0</nic_id>
      <port_forwarding/>
      <port_uuid>0aeb9585-5190-4f3b-b1aa-495e09c56b7d</port_uuid>
      <security_groups/>
      <subnet_uuid>none</subnet_uuid>
      <type>virtual</type>

<vim_interface_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</vim_interface_name>

    </interfaces>
    <vim_id>default_openstack_vim</vim_id>
    <vim_project>admin</vim_project>
    <vim_project_id>c12f013306d849e5b1bbf257c54d5891</vim_project_id>
    <host_uuid>6b8cf361c5ff08a5a886e26f591b8087dadcf2d2b34fb3b5d2772a8d</host_uuid>

    <host_name>my-server-65</host_name>
    <vm_uuid>9fea3fe7-9417-4734-b962-b24340941ef3</vm_uuid>
    <vm_group_name>vm1</vm_group_name>
    <vm_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</vm_name>
  </vm_source>
</esc_event>
```

次の例は、ESC からのオーバーロード通知を示しています。

```
Headers:
  esc-status-code = 200
  esc-status-message = VM [sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de]
overloaded.
Body:
<?xml version="1.0" encoding="UTF-8"?>
<esc_event xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <deployment_name>sample-dep</deployment_name>
  <event_name>MY_VM_OVERLOADED</event_name>
  <event_type>VM_OVERLOADED</event_type>
  <external_deployment_id>e911eecf-5f3f-456c-9c80-d99aca2416da</external_deployment_id>
```

```
  <external_tenant_id>etsi_tenant</external_tenant_id>
  <internal_deployment_id>99f7629f-98d3-40f5-ad68-7addcfe07006</internal_deployment_id>

  <internal_tenant_id>etsi_tenant</internal_tenant_id>
  <vm_source>

<generated_vm_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</generated_vm_name>

    <interfaces>
      <addresses>
        <address>
          <address_id>0</address_id>
          <gateway>172.24.0.1</gateway>
          <ip_address>172.24.0.37</ip_address>
          <dhcp_enabled>true</dhcp_enabled>
          <prefix>20</prefix>
          <subnet>365a0884-fdb3-424c-afe9-2deb3b39baae</subnet>
        </address>
      </addresses>
      <network_uuid>c7fafeca-aa53-4349-9b60-1f4b92605420</network_uuid>
      <mac_address>fa:16:3e:38:1d:6c</mac_address>
      <nic_id>0</nic_id>
      <port_forwarding/>
      <port_uuid>0aeb9585-5190-4f3b-b1aa-495e09c56b7d</port_uuid>
      <security_groups/>
      <subnet_uuid>none</subnet_uuid>
      <type>virtual</type>

<vim_interface_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</vim_interface_name>

    </interfaces>
    <vim_id>default_openstack_vim</vim_id>
    <vim_project>admin</vim_project>
    <vim_project_id>c12f013306d849e5b1bbf257c54d5891</vim_project_id>
    <host_uuid>6b8cf361c5ff08a5a886e26f591b8087dadcf2d2b34fb3b5d2772a8d</host_uuid>

    <host_name>my-server-65</host_name>
    <vm_uuid>9fea3fe7-9417-4734-b962-b24340941ef3</vm_uuid>
    <vm_group_name>vm1</vm_group_name>
    <vm_name>sample-dep_vm1_0_fbc3da46-e0c6-40dc-91c8-70b1a88857de</vm_name>
  </vm_source>
</esc_event>
```

- **VNFD**

  VNFD 通知には、VNF 操作フローの *isAutoscaleEnabled* 設定可能プロパティに必要なスケールアクションの手順が含まれています。

  スケーリングが自動的に有効になっていない場合は、KPI 手順を使用して手動 LCM 操作を実行できます。これは、ESC 通知ストリームを処理することで実行されます。KPI イベントを受信したら、通知を検証する必要があります。

  次のアクションを実行する必要があります。

  - 一致する VNF インスタンスを検索する
  - 適切な設定プロパティが自動操作を有効にするよう設定されていることを確認する

  検証に合格した場合、適切な操作の実行および関連する通知を生成するための操作フローを開始するよう要求できます。スケーリングでは、指定した KPI データによってスケーリ

ングパラメータが決定されます。プロパティファイルには、次の新しい属性が含まれます。

```
external.scaling.decision = 1
  #external.scaling.window = 120
  external.healing.decision = 1
  #external.healing.window = 120
```

- **VnfInstance リソース**

  VNFD は、現在の scaleStatus を使用してスケールレベルを決定します。要求の処理によって、ESCManager から要求する VM の数が決まります。要求は、増分の相対数（SCALE_IN または SCALE_OUT）のみを指定します。

  スケーリングする VNF の vnfInstance リソースから、*vnfInstanceId* を使用して、次のペイロードで *ScaleVnfToLevel* エンドポイントを呼び出すことができます。

  VnfLcmOpOcc.isAutomaticInvocation に true が設定されていることを確認します。

  次の例は JSON ペイロードを示しています。

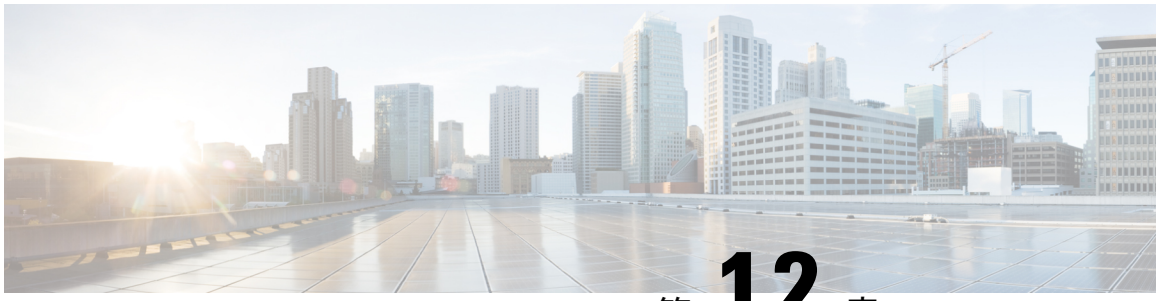
```
{
  /* "instantiationLevelId":"id111", */
  "scaleInfo": [
    { "aspectId":"processing", "scaleLevel":"3" },
    { "aspectId":"database",   "scaleLevel":"2" }
  ]
  "additionalParams": {
    "password": "pass1234",
    "username": "admin"
  },
  "action": "scale_to_level"
}
```

KPI 手順を使用した VNF の修復

ESC は、KPI 手順を使用して VM を自動修復できます。NFVO は、VNF の *isAutohealEnabled* 設定可能プロパティの変更中に自動修復を有効または無効にします。

isAutohealEnabled プロパティは、自動修復機能を有効（TRUE）または無効（FALSE）にすることを許可します。

-



第 12 章

ESC の管理

ETSI パフォーマンスレポート

ESC では、パフォーマンス管理ジョブ機能を使用して、メトリックや通知などの VNF のパフォーマンス情報を収集できます。最初に、パフォーマンス管理（PM）ジョブを作成する必要があります。PM job を作成した後、次のタスクを実行できます。

- パフォーマンス管理ジョブをクエリ、削除、または通知する
- 個々のレポートを読む、またはパフォーマンスレポートを取得する
- パフォーマンス管理ジョブのしきい値を設定する
- パフォーマンス管理ジョブのしきい値をクエリ、削除、または通知する
- サブスクリプションを管理、サブスクリプションをクエリ、サブスクライブ、または終了する

パフォーマンス管理ジョブ

このセクションでは、パフォーマンス管理ジョブについて説明します。

パフォーマンス管理ジョブの作成

さらにクエリを実行してレポートを実行するには、パフォーマンス管理ジョブを作成する必要があります。

メソッドタイプ：

POST

VNFM エンドポイント：

```
{api_root}/vnfpm/v1/pm_jobs (Data structure=CreatePmJobRequest)
```

要求ペイロード：

```
{
  "objectInstanceIds": [
    "cc6a34e5-0463-459a-b367-493ba997775f"
  ],
  "criteria": {
    "performanceMetric": [
      "default"
    ],
    "performanceMetricGroup": [
      "default"
    ],
    "collectionPeriod": 3600,
    "reportingPeriod": 14400
  }
}
```

応答ペイロード：

```
{
  "id": "13963644-11b0-4302-a13b-26ca3d9eb8f8",
  "objectInstanceIds": [
    "cc6a34e5-0463-459a-b367-493ba997775f "
  ],
  "criteria": {
    "performanceMetric": [
      "default"
    ],
    "performanceMetricGroup": [
      "default"
    ],
    "collectionPeriod": 3600,
    "reportingPeriod": 14400
  },
  "_links": {
    "self": {
      "href": "http://host:port/vnfpm/v1/pm_jobs/13963644-11b0-4302-a13b-26ca3d9eb8f8"
    },
    "objects": [
      {
        "href":
"http://host:port/vnflcm/v1/vnf_instances/cc6a34e5-0463-459a-b367-493ba997775f"
      }
    ]
  }
}
```

個々のパフォーマンス管理ジョブのクエリ

NFVO は、個々のパフォーマンス管理ジョブをクエリします。

メソッドタイプ：

GET

VNFM エンドポイント：

{api_root}/vnfpm/v1/pm_jobs/{pmJobId} or GET {api_root}/vnfpm/v1/pm_jobs/{pmJobId}

要求ペイロード：

該当なし。

応答ペイロード：

```
{
  "id": "13963644-11b0-4302-a13b-26ca3d9eb8f8",
  "objectInstanceIds": [
    "cc6a34e5-0463-459a-b367-493ba997775f "
  ],
  "criteria": {
    "performanceMetric": [
      "default"
    ],
    "performanceMetricGroup": [
      "default"
    ],
    "collectionPeriod": 3600,
    "reportingPeriod": 14400,
    "reports": [
      {
        "href": "uri_where_report_can_be_obtained",
        "readyTime": "2018-08-20T06:17:35.081+0000",
        "expiryTime": "2018-10-20T06:17:35.081+0000",
        "fileSize": "5000"
      }
    ]
  },
  "_links": {
    "self": {
      "href": "http://host:port/vnfpm/v1/pm_jobs/13963644-11b0-4302-a13b-26ca3d9eb8f8"
    },
    "objects": [
      {
        "href":
"http://host:port/vnflcm/v1/vnf_instances/cc6a34e5-0463-459a-b367-493ba997775f"
      }
    ]
  }
}
```

（注） レポートが使用可能な場合にのみ、レポートセクションが応答ペイロードに追加されます（上記を参照）。

すべての属性名と、応答ペイロードの属性名から参照されるデータタイプは、属性ベースのフィルタ処理でサポートされます。

すべてのパフォーマンス管理ジョブのクエリ

NFVO は、すべてのパフォーマンス管理ジョブのリストを取得します。

メソッドタイプ：

GET

VNFM エンドポイント：

```
{api_root}/vnfpm/v1/pm_jobs or GET {api_root}/vnfpm/v1/pm_jobs
```

要求ペイロード：

該当なし。

応答ペイロード：

```
{
  "id": "13963644-11b0-4302-a13b-26ca3d9eb8f8",
  "objectInstanceIds": [
    "cc6a34e5-0463-459a-b367-493ba997775f "
  ],
  "criteria": {
    "performanceMetric": [
      "default"
    ],
    "performanceMetricGroup": [
      "default"
    ],
    "collectionPeriod": 3600,
    "reportingPeriod": 14400,
    "reports": [
      {
        "href": "uri_where_report_can_be_obtained",
        "readyTime": "2018-08-20T06:17:35.081+0000",
        "expiryTime": "2018-10-20T06:17:35.081+0000",
        "fileSize": "5000"
      }
    ]
  },
  "_links": {
    "self": {
      "href": "http://host:port/vnfpm/v1/pm_jobs/13963644-11b0-4302-a13b-26ca3d9eb8f8"
    },
    "objects": [
      {
        "href":
"http://host:port/vnflcm/v1/vnf_instances/cc6a34e5-0463-459a-b367-493ba997775f"
      }
    ]
  }
}
```

（注） レポートが使用可能な場合にのみ、レポートセクションが応答ペイロードに追加されます（上記を参照）。

応答ペイロードのすべての属性名と、属性名から参照されるデータタイプは、属性ベースのフィルタ処理でサポートされます。

パフォーマンス管理ジョブの削除

NFVO は既存のパフォーマンス管理ジョブに削除要求を送信します。

```
DELETE {api_root}/vnfpm/v1/pm_jobs/{pmJobId}
```

NFVO は、PerformanceInformationAvailableNotification 通知を使って通知されます。

パフォーマンス管理ジョブのしきい値の設定

このセクションでは、パフォーマンス管理ジョブのしきい値を設定する方法について説明します。

しきい値の作成

NFVO は、パフォーマンス管理ジョブのしきい値を作成するための作成要求を送信します。

メソッドタイプ：

POST

VNFM エンドポイント：

```
{api_root}/vnfpm/v1/thresholds (Datastructure=CreateThresholdRequest)
```

要求ペイロード：

```
{
  "objectInstanceId": "cc6a34e5-0463-459a-b367-493ba997775f",
  "criteria": {
    "performanceMetric": "default",
    "thresholdType": "SIMPLE",
    "simpleThresholdDetails": {
      "thresholdValue": 0.8,
      "hysteresis": 0.9
    }
  }
}
```

応答ペイロード：

```
{
  "id": "23f52511-9f72-4797-881b-c0f72e60a052",
  "objectInstanceId": "cc6a34e5-0463-459a-b367-493ba997775f",
  "criteria": {
    "performanceMetric": "default",
    "thresholdType": "SIMPLE",
    "simpleThresholdDetails": {
      "thresholdValue": 0.8,
      "hysteresis": 0.9
    }
  },
  "_links": {
    "self": {
     "href": "http://host:port/vnfpm/v1/thresholds/23f52511-9f72-4797-881b-c0f72e60a052"

    },
    "object": [
      {
        "href":
"http://host:port/vnflcm/v1/vnf_instances/cc6a34e5-0463-459a-b367-493ba997775f"
      }
    ]
```

```
  }
}
```

個々のしきい値のクエリ

NFVO は、パフォーマンス管理ジョブのしきい値をクエリできます。

GET

VNFM エンドポイント：

```
{api_root}/vnfpm/v1/thresholds/{thresholdId}
```

要求ペイロード：NA

応答ペイロード：

```
{
  "id": "23f52511-9f72-4797-881b-c0f72e60a052",
  "objectInstanceId": "cc6a34e5-0463-459a-b367-493ba997775f",
  "criteria": {
    "performanceMetric": "default",
    "thresholdType": "SIMPLE",
    "simpleThresholdDetails": {
      "thresholdValue": 0.8,
      "hysteresis": 0.9
    }
  },
  "_links": {
    "self": {
     "href": "http://host:port/vnfpm/v1/thresholds/23f52511-9f72-4797-881b-c0f72e60a052"

    },
    "object": [
      {
        "href":
"http://host:port/vnflcm/v1/vnf_instances/cc6a34e5-0463-459a-b367-493ba997775f"
      }
    ]
  }
}
```

（注） しきい値 ID を指定する場合、属性ベースのフィルタ処理はできません。

すべてのしきい値のクエリ

NFVO は、パフォーマンス管理ジョブのしきい値をクエリできます。

メソッドタイプ：

GET

VNFM エンドポイント：

```
{api_root}/vnfpm/v1/thresholds
```

要求ペイロード：NA

応答ペイロード：

```
{
  "id": "23f52511-9f72-4797-881b-c0f72e60a052",
  "objectInstanceId": "cc6a34e5-0463-459a-b367-493ba997775f",
  "criteria": {
    "performanceMetric": "default",
    "thresholdType": "SIMPLE",
    "simpleThresholdDetails": {
      "thresholdValue": 0.8,
      "hysteresis": 0.9
    }
  },
  "_links": {
    "self": {
     "href": "http://host:port/vnfpm/v1/thresholds/23f52511-9f72-4797-881b-c0f72e60a052"

    },
    "object": [
      {
        "href":
"http://host:port/vnflcm/v1/vnf_instances/cc6a34e5-0463-459a-b367-493ba997775f"
      }
    ]
  }
}
```

（注） 応答ペイロードのすべての属性名と、属性名から参照されるデータタイプは、属性ベースのフィルタ処理でサポートされます。

しきい値の削除

NFVO は、既存のパフォーマンス管理ジョブのしきい値設定を削除する削除要求を送信します。

```
DELETE {api_root}/vnfpm/v1/thresholds/{thresholdId}
```

ESC が設定されたしきい値を超えると、NFVO は `ThresholdCrossedNotification` を受信します。

パフォーマンス管理ジョブへの登録

このセクションでは、パフォーマンス管理ジョブの登録について説明します。

パフォーマンス管理サブスクリプションの作成

NFVO はパフォーマンス管理ジョブに登録できます。

メソッドタイプ：

POST

VNFM エンドポイント：

```
{api_root}/vnfpm/v1/subscriptions(DataStructure=PmSubscriptionRequest)
```

例 1：

要求ペイロード：

```
{
           "callbackUri": "http://host:port/notification",
           "filter": {
                      "notificationTypes": ["ThresholdCrossedNotification",
"PerformanceInformationAvailableNotification"],
                      "vnfInstanceSubscriptionFilter": {
                            "vnfdIds": ["25ec9e1c-ad9e-4613-9280-411920f3649a"],
                             "vnfInstanceIds": ["cc6a34e5-0463-459a-b367-493ba997775f"]
                          }
            }
}
```

応答ペイロード：

```
                      {
                             "id": "4fba7dcb-e015-4674-9c50-8cee7059eb91"
                             "callbackUri": "http://host:port/notification",
                                 "filter": {
                                                "notificationTypes": ["ThresholdCrossedNotification",
                      PerformanceInformationAvailableNotification"],
                                                 "vnfInstanceSubscriptionFilter": {
                                                          "vnfdIds": ["25ec9e1c-ad9e-4613-9280-411920f3649a"],

                                                             "vnfInstanceIds":
                      ["cc6a34e5-0463-459a-b367-493ba997775f"]                 }
                                },
                               "_links": {
                                          "self": {
                                                                "href":
                      "http://host:port/vnfpm/v1/subscriptions/4fba7dcb-e015-4674-9c50-8cee7059eb91"
                                           }
                                }
                      }
```

例 2：

要求ペイロード：

```
                      {
                            "callbackUri": "http://host:port/notification",
                            "filter": {
                                        "notificationTypes": ["ThresholdCrossedNotification",
                      "PerformanceInformationAvailableNotification"],
                                         "vnfInstanceSubscriptionFilter": {
                                                        "vnfProductsFromProviders": [{
                                                                          "vnfProvider": "Cisco",
                                                                           "vnfProducts": [{
                                                                                    "vnfProductName":
                      "vnfd-1VDU",
                                                                                          "versions": [{

                      "vnfSoftwareVersion": "1.3.1",

                      "vnfdVersions": ["1.0", "1.1", "1.2"]
                                                                                                  }]
                                                                            }]
                                                     }],
                                                  "vnfInstanceNames":
```

```
["kaswaczy-TestETSIPmSubscriptionGet-114113"]
                  }
                }
}
```

応答ペイロード：

```
{
      "id": "4fba7dcb-e015-4674-9c50-8cee7059eb92"
       "callbackUri": "http://host:port/notification",
         "filter": {
                     "notificationTypes": ["ThresholdCrossedNotification",
"PerformanceInformationAvailableNotification"],
                      "vnfInstanceSubscriptionFilter": {
                      "vnfProductsFromProviders": [{
                                                          "vnfProvider": "Cisco",
                                                           "vnfProducts": [{
                                                                          "vnfProductName":
 "vnfd-1VDU",
                                                                             "versions": [{

 "vnfSoftwareVersion": "1.3.1",

  "vnfdVersions": ["1.0", 1.1", 1.2"]
                                                                          }]
                                                             }]
                                       }],
                                  "vnfInstanceNames":
["kaswaczy-TestETSIPmSubscriptionGet-114113"]
                           }
             },
            "_links": {
                         "self": {
                            "href":
"http://host:port/vnfpm/v1/subscriptions/4fba7dcb-e015-4674-9c50-8cee7059eb92"
                        }
           }
}
```

（注）

- vnfdIds 属性と vnfProductsFromProviders 属性は相互に排他的です。1 つの作成要求で指定できるのはどちらか 1 つだけです。
- vnfInstanceIds 属性と vnfInstanceNames 属性は相互に排他的です。1 つの作成要求で指定できるのはどちらか 1 つだけです。
- 指定した callbackUri とフィルタが既存のサブスクリプションと完全に一致する場合、サブスクリプションの重複は許可されないことを示すエラーメッセージと共に、サブスクリプションの作成操作は失敗します。

個々のパフォーマンス管理サブスクリプションのクエリ

メソッドタイプ：

GET

VNFM エンドポイント：

```
{api_root}/vnfpm/v1/subscriptions/{subscriptionId}
```

要求ペイロード：NA

応答ペイロード：

```
{
                "id": "4fba7dcb-e015-4674-9c50-8cee7059eb91"
                "callbackUri": "http://host:port/notification",
                "filter": {
                                "notificationTypes": ["ThresholdCrossedNotification",
"PerformanceInformationAvailableNotification"],
                                "vnfInstanceSubscriptionFilter": {
                                                "vnfdIds":
["25ec9e1c-ad9e-4613-9280-411920f3649a"],
                                                "vnfInstanceIds":
["cc6a34e5-0463-459a-b367-493ba997775f"]                }
                },
                "_links": {
                                "self": {
                                                "href":
"http://host:port/vnfpm/v1/subscriptions/4fba7dcb-e015-4674-9c50-8cee7059eb91"
                                }
                }
}
```

（注） サブスクリプション ID を指定する場合、属性ベースのフィルタ処理はできません。

すべてのパフォーマンス管理サブスクリプションのクエリ

メソッドタイプ：

GET

```
{api_root}/vnfpm/v1/subscriptions
```

要求ペイロード：NA

応答ペイロード：

```
{
                "_embedded": {
                                "pmSubscriptions": [{
                                                "id":
"4fba7dcb-e015-4674-9c50-8cee7059eb91""callbackUri": "http://host:port/notification",

                                                "filter": {
                                                                "notificationTypes":
["ThresholdCrossedNotification", "PerformanceInformationAvailableNotification"],

"vnfInstanceSubscriptionFilter": {

"vnfdIds": ["25ec9e1c-ad9e-4613-9280-411920f3649a"],

"vnfInstanceIds": ["cc6a34e5-0463-459a-b367-493ba997775f"]
                                                                }
```

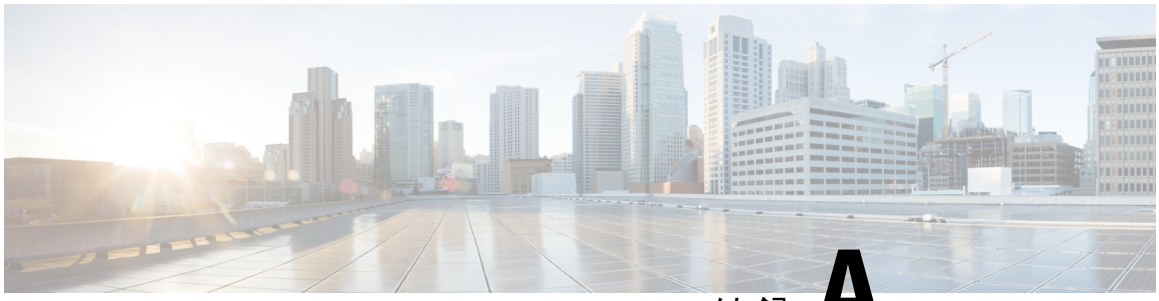
```
                                                },
                                                "_links": {
                                                                "self": {
                                                                                "href":
 "http://host:port/vnfpm/v1/subscriptions/4fba7dcb-e015-4674-9c50-8cee7059eb91"
                                                                }
                                                }
                                },
                                {
                                                "id":
"4fba7dcb-e015-4674-9c50-8cee7059eb92""callbackUri": "http://host:port/notification",

                                                "filter": {
                                                                "notificationTypes":
["ThresholdCrossedNotification", "PerformanceInformationAvailableNotification"],

"vnfInstanceSubscriptionFilter": {

"vnfProductsFromProviders": [{

          "vnfProvider": "Cisco",

          "vnfProducts": [{

                          "vnfProductName": "vnfd-1VDU",

                          "versions": [{

                                          "vnfSoftwareVersion": "1.3.1",

                                          "vnfdVersions": ["1.0", "1.1", "1.2"]

                          }]

          }]
                                                                                }],

"vnfInstanceNames": ["kaswaczy-TestETSIPmSubscriptionGet-114113"]
                                                                }
                                                },
                                                "_links": {
                                                                "self": {
                                                                                "href":
 "http://host:port/vnfpm/v1/subscriptions/4fba7dcb-e015-4674-9c50-8cee7059eb92"
                                                                }
                                                }
                                }]
                }
}
```


（注） 応答ペイロードで参照されるすべての属性名とデータタイプは、パラメータの属性ベースのフィルタ処理でサポートされます。

パフォーマンス管理サブスクリプションの終了

NFVO はサブスクリプションを終了できます。

```
DELETE {api_root}/vnfpm/v1/subscriptions/{subscriptionId}
```



付録 A

ETSI 製品のプロパティ

ETSI 製品のプロパティ

ESC の動作を決定するために設定できるプロパティは多数あります。これらのプロパティにより、システムアーキテクチャの ESC と NFVO を統合できます。

プロパティファイルには次の場所からアクセスできます。

```
/opt/cisco/esc/esc_database/etsi-production.properties
```

次の表に、ETSI NFV MANO スタック内で VNFM として動作する ESC の動作を制御するために使用できるパラメータを示します。

表 *8 : ETSI* 製品のプロパティ

プロパティ名	説明	タイプ	デフォルト値（Default Value）
`server.host`	ETSI サービスが存在するホスト IP アドレス。これは、サーバに複数の IP アドレスがある場合、または展開が高可用性に設定されている場合（その後 VIP に設定する必要あり）に必須のプロパティです。	文字列	

プロパティ名	説明	タイプ	デフォルト値（Default Value）
server.host.preferInet6	サーバに複数の IP アドレスタイプが割り当てられている場合は、任意の IPv4 アドレスではなく、この IPv6 アドレスを使用します。	ブール値	false
server.port	HTTP を介した通信に使用されるポート。	整数（Integer）	8250
server.port.https	HTTPS を介した通信に使用されるポート。	整数（Integer）	8251
certificate.validation	HTTPS を使用するときに提示される証明書のホストを検証するかどうかを決定します。より緩やかな検証が可能で、特にテストの際に役立ちます。	ブール値	true
notification.maxThreads	通知サービスに使用されるスレッドの最大数。	整数（Integer）	3
notification.subscription.test	新しいサブスクリプションを作成したら、テストするかどうかを決定します	ブール値	true
notification.links.httpScheme	通知用に NFVO との通信に使用される HTTP スキーム。有効な値：http、https。	列挙体	https
notification.retry.maxAttempt	通知再試行メカニズムの再試行回数。	整数（Integer）	5

プロパティ名	説明	タイプ	デフォルト値（Default Value）
notification.retry.backOff.delay	通知再試行メカニズムの間隔。	整数（Integer）	1000
security.user.name	**必須**。これは REST API のユーザ名です。これは、sudo escadm etsi set --rest_user <username>:<password> によって設定され、ここで同期する必要があります。	文字列	
nfvo.apiRoot	**必須**。NFVO の apiRoot。	文字列	localhost：8280
nfvo.httpScheme	NFVO との通信に使用される HTTP スキーム。有効な値：http、https。	列挙体	http
nfvo.isPackageNotificationSupported	VNFM がパッケージ通知への登録を試行するかどうかを決定します。	ブール値	true
nfvo.callback.httpScheme	応答へのポーリング時に NFVO との通信に使用される HTTP スキーム。有効な値：http、https。	列挙体	https
nfvo.userName	NFVO ログイン情報のユーザ名。	文字列	
nfvo.password	プレーンテキストで必要な NFVO ログイン情報のパスワード。	文字列	
retryTemplate.expotential.retryPolicy.maxAttempt	指数的再試行メカニズムの再試行回数。	整数（Integer）	1000

ETSI 製品のプロパティ

プロパティ名	説明	タイプ	デフォルト値（Default Value）
retryTemplate.expotential.backOffPolicy.interval.initial	指数的再試行メカニズムの開始間隔。	整数（Integer）	1000
retry.simple.maxAttempt	単純な再試行メカニズムの再試行回数。	整数（Integer）	50
retry.simple.backOff.delay	単純な再試行メカニズムの間隔。	整数（Integer）	1000
nfvo.allPackagesFilter	パッケージのクエリ時に NFVO のパッケージをフィルタ処理するために使用する値。	文字列	
mapping.vimConnectionInfo.accessInfo.username	accessInfo でユーザ名を指定する場合に、代替属性名を指定します。	文字列	username
mapping.vimConnectionInfo.accessInfo.password	accessInfo でパスワードを指定する場合に、代替属性名を指定します。	文字列	パスワード
mapping.vimConnectionInfo.accessInfo.project	accessInfo でプロジェクトを指定する場合に、代替属性名を指定します。	文字列	プロジェクト
mapping.vimConnectionInfo.accessInfo.projectDomain	accessInfo で projectDomain を指定する場合に、代替属性名を指定します。	文字列	projectDomain
mapping.vimConnectionInfo.accessInfo.userDomain	accessInfo で userDomain を指定する場合に、代替属性名を指定します。	文字列	userDomain

プロパティ名	説明	タイプ	デフォルト値（Default Value）
mapping.vimConnectionInfo.accessInfo.vim_project	accessInfo で vim_project を指定する場合に、代替属性名を指定します。	文字列	vim_project
mapping.vimConnectionInfo.accessInfo.vim_vdc	accessInfo で vim_vdc を指定する場合に、代替属性名を指定します。	文字列	vim_vdc
nfvo.grantRequest.retry.maxAttempt	失敗した GrantRequest 試行の再試行回数。	整数（Integer）	5
nfvo.grantRequest.retry.backOff.delay	失敗した GrantRequest 試行の再試行間隔。	整数（Integer）	1000
spring.jackson.date-format	さまざまな NFVO 実装で日付を正しく読み取るための日付形式を表す文字列。	文字列	yyyy-MM-dd'T'HH:mm:ss.SSSXXX
nfvo.authenticationType	使用されている NFVO の認証タイプを設定します。必須プロパティ。有効なオプションは「BASIC」、「OAUTH2」、「OFF」です。他のすべての文字列は「OFF」と同様に扱われます。これを使用して、基本認証と OAuth2 認証を有効にします。	文字列	
nfvo.clientID	NFVO OAuth2 認証の場合。クライアント ID。	文字列	

プロパティ名	説明	タイプ	デフォルト値（Default Value）
nfvo.clientSecret	NFVO OAuth2 認証の場合。クライアントシークレット。	文字列	
nfvo.tokenEndpoint	NFVO OAuth2 認証の場合。NFVO から OAuth2 トークンを取得する ETSI のエンドポイント。	文字列	
rate.limit.capacity.read	ETSI REST API への読み取り（GET）要求のバケット容量を設定します。 デフォルトでは無効になっています。	整数（Integer）	
rate.limit.perSecond.read	ETSI REST API への読み取り（GET）要求に対してバケットが空になるレート（1 秒あたり）を設定します。 デフォルトでは無効になっています。	倍精度浮動小数点型	
rate.limit.capacity.write	ETSI REST API への書き込み（POST、PUT、PATCH、DELETE）要求のバケット容量を設定します。 デフォルトでは無効になっています。	整数（Integer）	

プロパティ名	説明	タイプ	デフォルト値（Default Value）
rate.limit.perSecond.write	ETSI REST API への書き込み（POST、PUT、PATCH、DELETE）要求に対してバケットが空になるレート（1 秒あたり）を設定します。 デフォルトでは無効になっています。	倍精度浮動小数点型	
log.multiple.query	複数の VNF インスタンスのクエリに対する応答と、複数の VNF ライフサイクル管理操作のクエリに対する応答のロギングを有効にするフラグ。	ブール値	false
scheduled.cleanup[vnfLcmOpOcc].interval.value	VnfLcmOpOcc クリーンアップタスクの間隔値を設定します。 interval.value と interval.unit の組み合わせによって、クリーンアップタスクが実行される頻度が決まります。	整数（Integer）	1

プロパティ名	説明	タイプ	デフォルト値（**Default Value**）
scheduled.cleanup[vnfLcmOpOcc].interval.unit	VnfLcmOpOcc クリーンアップタスクの間隔単位を設定します。 interval.value と interval.unit の組み合わせによって、クリーンアップタスクが実行される頻度が決まります。 有効な値は、次のとおりです。 `NANOS、MICROS、MILLIS、SECONDS、MINUTES、HOURS、HALF_DAYS、DAYS`		DAYS
scheduled.cleanup[vnfLcmOpOcc].age.value	VnfLcmOpOcc クリーンアップタスクの経過時間の値を設定します。 age.value と age.unit の組み合わせによって、削除する孤立レコードの経過時間が決まります。	整数（Integer）	60

プロパティ名	説明	タイプ	デフォルト値（**Default Value**）
scheduled.cleanup[vnfLcmOpOcc].age.unit	VnfLcmOpOcc クリーンアップタスクの経過時間の単位を設定します。 age.value と age.unit の組み合わせによって、削除する孤立レコードの経過時間が決まります。 有効な値は、次のとおりです。 `NANOS、MICROS、MILLIS、SECONDS、MINUTES、HOURS、HALF_DAYS、DAYS`		DAYS

プロパティ名	説明	タイプ	デフォルト値（Default Value）
paging.size	0 より大きい値を設定すると、クエリエンドポイントのページングがオンになります。 この値は、1 ページあたりに含まれる結果の数を表します。 応答がページングされ、さらにページがある場合、応答には「Link」という名前のヘッダーと rel="next" が含まれます。次に例を示します。 <http://example.com /resources?nextpage_opaque _marker=abc123>; rel="next" リンクの URL は次のページを取得します。 取得するページがそれ以上ない場合、リンクヘッダーは省略されます。	整数（Integer）	0
attribute.selector.default.all_fields	値を「true」に設定すると、属性セレクタが指定されていない場合（all_fields）、属性の完全なセットを返すよう、ETSI クエリエンドポイントの動作が変更されます。	ブール値	false

リソース定義の詳細については、ETSI API のリソース定義 （3 ページ）を参照してください。

ETSI 製品のプロパティ