# 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 （5 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 *1: 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。<br>（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。<br>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。<br>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。<br>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。<br>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。<br>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。<br>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。<br>exclude_default パラメータはフラグであり、値はありません。<br>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 *2: 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 *3:* 属性の選択をサポートするリソース

| 名前 | 基数 | 説明 |
|---|---|---|
| VNF インスタンス | | |

| 名前 | 基数 | 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。<br>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。<br>• vnfConfigurableProperties<br>• vimConnectionInfo<br>• InstantiatedVnfInfo<br>• metadata<br>• extension |
| 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。<br>• operationParams<br>• error<br>• resourceChanges<br>• changedInfo<br>• changedExtConnectivity |
| PM Jobs | | |

| 名前 | 基数 | 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。<br>• 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 （2 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
- metadata
- extensions
- 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：

```
Content-Type: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。

メソッドタイプ：

```
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.
```