Elastic Services Controller インターフェイス

Elastic Services Controller インターフェイス

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

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

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

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

Elastic Services Controller NB API

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

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

NETCONF/YANG ノースバウンド API

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

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

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

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

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

次に、NETCONF/YANG API の例を示します。

テナントを作成する NETCONF 要求

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

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

テナントが正常に作成された後の NETCONF 通知：

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

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

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

NETCONF 応答

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

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

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

サンプルエラーメッセージ（拒否された要求）は次のとおりです。

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

```
        <error-path xmlns:esc="http://www.cisco.com/esc/esc"

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

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

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

```
<networks>
    <network>
        <name>mgmt-net</name>
        <subnet>
            <name>mgmt-net-subnet</name>
            <ipversion>ipv4</ipversion>
            <dhcp>false</dhcp>
            <address>10.0.0.0</address>
            <no_gateway>true</no_gateway>
            <!-- DISABLE GATEWAY -->
            <gateway>10.0.0.1</gateway>
            <netmask>255.255.255.0</netmask>
        </subnet>
    </network>
</networks>
```

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

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

Openstack

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

VMware vCenter

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

• vcenter_username：vCenter ユーザが表示されます。

複数リソースを設定するための NETCONF 要求

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

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

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

ESC には 2 種類の依存関係があります。

1. 参照型依存関係
2. 階層型依存関係

参照型依存関係

参照型依存関係では、1 つの設定に別の設定への参照があります。

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

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

階層型依存関係

階層型依存関係では、1 つの設定が別の設定の中にあります。

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

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

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

REST ノースバウンド API

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

API モデルには、これらのプログラマチック エンティティが含まれます。

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

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

REST API ドキュメント

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

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

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

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

REST API の例：

REST を使用してテナントを作成するには、次の手順を実行します。

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

テナントが正常に作成された後の REST 応答：

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

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

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

ETSI NFV MANO Northbound API

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

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

ETSI API ドキュメント

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

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

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

Elastic Services Controller ポータル

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

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

ESC ポータルを開始、停止、および再起動するには、次の手順を実行します。

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

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