Cisco APIC レイヤ 4 から レイヤ 7 のデバイス パッケージ開発ガイド

初版：2013 年 10 月 31 日

最終更新：2014 年 08 月 01 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。

目 次

はじめに

この前書きは、次の項で構成されています。

対象読者

このガイドは、次の 1 つ以上を担い、その専門知識を備えているデータセンターの管理者を主な対象にしています。

- 仮想マシンのインストールと管理
- サーバ管理
- スイッチおよびネットワーク管理

表記法

コマンドの説明には、次のような表記法が使用されます。

表記法	説明
bold	太字の文字は、表示どおりにユーザが入力するコマンドおよびキーワードです。
italic	イタリック体の文字は、ユーザが値を入力する引数です。

表記法	説明
[x]	省略可能な要素（キーワードまたは引数）は、角カッコで囲んで示しています。
[x | y]	いずれか 1 つを選択できる省略可能なキーワードや引数は、角カッコで囲み、縦棒で区切って示しています。
{x | y}	必ずいずれか1つを選択しなければならない必須キーワードや引数は、波カッコで囲み、縦棒で区切って示しています。
[x {y | z}]	角カッコまたは波カッコが入れ子になっている箇所は、任意または必須の要素内の任意または必須の選択肢であることを表します。 角カッコ内の波カッコと縦棒は、省略可能な要素内で選択すべき必須の要素を示しています。
`variable`	ユーザが値を入力する変数であることを表します。イタリック体が使用できない場合に使用されます。
string	引用符を付けない一組の文字。 string の前後には引用符を使用しません。引用符を使用すると、その引用符も含めて string とみなされます。

例では、次の表記法を使用しています。

表記法	説明
`screen フォント`	スイッチが表示する端末セッションおよび情報は、screen フォントで示しています。
`太字の screen フォント`	ユーザが入力しなければならない情報は、太字の screen フォントで示しています。
イタリック体の *screen* フォント	ユーザが値を指定する引数は、イタリック体の screen フォントで示しています。
< >	パスワードのように出力されない文字は、山カッコ（< >）で囲んで示しています。
[ ]	システム プロンプトに対するデフォルトの応答は、角カッコで囲んで示しています。
!、#	コードの先頭に感嘆符（!）またはポンド記号（#）がある場合には、コメント行であることを示します。

このマニュアルでは、次の表記法を使用しています。

（注） 「注釈」です。役立つ情報やこのマニュアルに記載されていない参照資料を紹介しています。

注意 「要注意」の意味です。機器の損傷またはデータ損失を予防するための注意事項が記述されています。

警告 安全上の重要事項

「危険」の意味です。人身事故を予防するための注意事項が記述されています。機器の取り扱い作業を行うときは、電気回路の危険性に注意し、一般的な事故防止対策に留意してください。各警告の最後に記載されているステートメント番号を基に、装置に付属の安全についての警告を参照してください。

これらの注意事項を保管しておいてください。

関連資料

アプリケーション セントリック インフラストラクチャのマニュアル セットには次のマニュアルがあります。

Web ベースのマニュアル

- 『*Cisco APIC Management Information Model Reference*』
- 『*Cisco APIC Online Help Reference*』
- 『*Cisco ACI MIB Support List*』

ダウンロード可能なマニュアル

- 『*Cisco Application Centric Infrastructure Release Notes*』
- 『*Cisco Application Centric Infrastructure Fundamentals Guide*』
- 『*Cisco APIC Getting Started Guide*』
- 『*Cisco APIC REST API User Guide*』
- 『*Cisco APIC Command Line Interface User Guide*』
- 『*Cisco APIC Faults, Events, and System Message Guide*』
- 『*Cisco APIC レイヤ 4 から レイヤ 7 のデバイス パッケージ開発ガイド*』
- 『*Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*』

- 『*Cisco ACI Firmware Management Guide*』
- 『*Cisco ACI Troubleshooting Guide*』
- 『*Cisco ACI NX-OS Syslog Reference Guide*』
- 『*Cisco ACI Switch Command Reference, NX-OS Release 11.0*』
- 『*Cisco ACI MIB Quick Reference*』
- 『*Cisco Nexus CLI to Cisco APIC Mapping Guide*』
- 『*Installing the Cisco Application Virtual Switch with the Cisco APIC*』
- 『*Configuring the Cisco Application Virtual Switch using the Cisco APIC*』
- 『*Application Centric Infrastructure Fabric Hardware Installation Guide*』

マニュアルに関するフィードバック

このマニュアルに関する技術的なフィードバック、または誤りや記載もれなどお気づきの点がございましたら、apic-docfeedback@cisco.com までご連絡ください。 ご協力をよろしくお願いいたします。

マニュアルの入手方法およびテクニカル サポート

マニュアルの入手方法、Cisco Bug Search Tool（BST）の使用、サービス要求の送信、および追加情報の収集に関する情報については、http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html で『*What's New in Cisco Product Documentation*』を参照してください。

『*What's New in Cisco Product Documentation*』では、シスコの新規および改訂版の技術マニュアルの一覧を、RSS フィードとして購読できます。また、リーダー アプリケーションを使用して、コンテンツをデスクトップに配信することもできます。 RSS フィードは無料のサービスです。

第 1 章

概要

Application Policy Infrastructure Controller とのサービス統合の概要

Application Policy Infrastructure Controller（APIC）は、セキュア ソケット レイヤ（SSL）のオフロード、サーバ ロード バランシング（SLB）、Web アプリケーション ファイアウォール（WAF）、および従来のファイアウォールなどのネットワーク サービスの挿入とプロビジョニングを自動化します。 ネットワーク サービスは、アプリケーション デリバリ コントローラ（ADC）やファイアウォールなどのサービス アプライアンスで行われます。 サービス アプライアンスは、1 つ以上のサービス機能を実行できます。

APIC によってサービス グラフを定義できます。 サービス グラフの各ノードはネットワーク機能を表します。 グラフは、ユーザ定義のポリシーに基づくサービス機能を定義します。 サービス アプライアンス（デバイス）は、グラフ内のサービス機能を実行します。 1 つ以上のサービス アプライアンスが、グラフに必要なサービスを実行できます。 1つ以上のサービス機能が単一のサービス デバイスで実行できます。

APIC には、ネットワーク サービスのアプライアンス（デバイス）にネットワーク サービス機能を挿入し、設定するためのデバイス パッケージが必要です。 デバイス パッケージは以下を含む zip ファイルです。

- `DeviceModel.xml`：デバイス パッケージには、`DeviceModel.xml` という デバイス仕様である 1 つのXML ファイルを含める必要があります。 デバイス仕様は各機能の設定など、デバイスの階層的説明を提供する XML ファイルで、APIC の管理オブジェクトにマッピングされています。 デバイス仕様は、以下を定義します。
  - デバイス機能

- 各機能を設定するときに APIC が必要とするパラメータ
- 各機能のインターフェイスおよびネットワーク接続情報

- `DeviceScript.py`：デバイス パッケージには、`DeviceScript.py` という1 つの Python ファイルを含める必要があります。 この Python ファイルでは、APIC とデバイス間でやりとりするための API を定義する必要があります。 デバイス仕様 XML ファイルが、デバイス スクリプトをこの Python ファイルに関連付けます。

  デバイス スクリプトは、APIC とデバイス間の通信を管理します。 また、一般的な API からのコールをデバイス固有のコールに変換することで、APIC イベントとデバイスの相互作用を表す機能コール間のマッピングを定義します。

  APIC にデバイス パッケージをアップロードすると、APIC はデバイスを表す管理対象オブジェクトの階層を作成し、デバイスのスクリプトを検証します。

- Python またはテキスト ファイルを含む追加ファイルおよびディレクトリ。 デバイス パッケージには、デバイスのやりとりと設定をサポートする Python ライブラリを含めることができます。 サポート Python ファイルは複数のディレクトリに分割できます。 またパッケージにはサポート テキスト ファイルも含めることができます。 デバイス パッケージには、サポート Python egg ファイルを含めることができます。

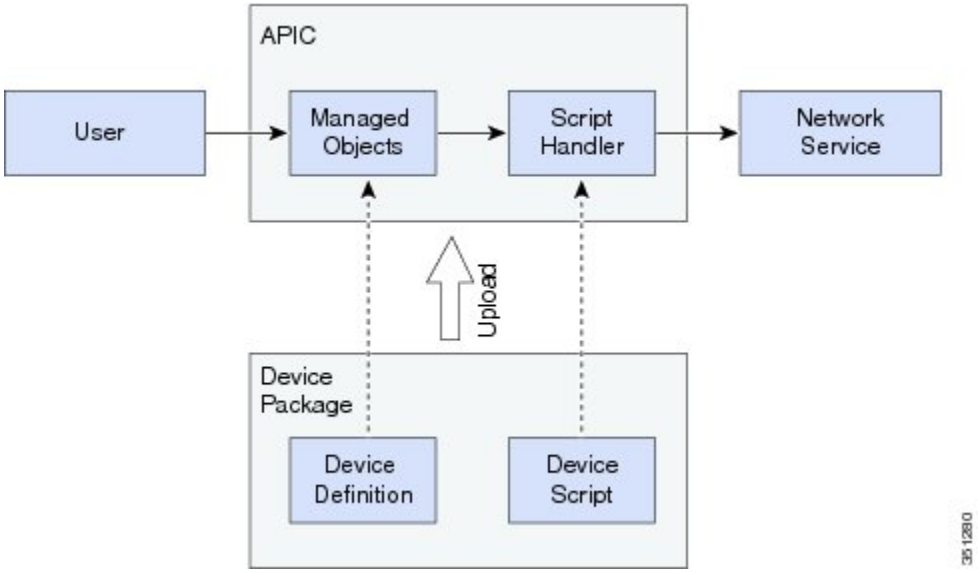
- `images` ディレクトリ：デバイス パッケージには、`images` ディレクトリを含め、またディレクトリには *`vendor_name.gif`* という 1 つのファイルを含める必要があります。 イメージ サイズは 28 ピクセル X 28 ピクセルにする必要があります。

次に、Insieme というベンダーからのパッケージ zip ファイルのリストの例を示します。

```
bash-4.1$ unzip -l insiemeDevicePackage.zip
Archive:  insiemeDevicePackage.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
   309597  03-17-2014 17:39   DeviceModel.xml
     1597  03-17-2014 17:39   DeviceScript.py
        0  01-30-2014 15:36   common/
     1919  02-06-2014 11:35   common/deviceInterface.py
        0  01-30-2014 15:36   feature/
    21919  02-06-2014 11:35   feature/functionCommon.py
     6485  10-31-2013 06:32   feature/function2.py
     7747  10-31-2013 06:32   feature/function1.py
        0  10-31-2013 06:32   feature/__init__.py
        0  01-30-2014 15:36   lib/
     1919  02-06-2014 11:35   lib/
        0  01-30-2014 15:36   util/
    21919  02-06-2014 11:35   util/logging.py
        0  10-31-2013 06:32   parser/configParser.py
        0  02-12-2014 10:07   images/
     1380  02-12-2014 10:07   images/insieme.gif
```

次の図では、デバイス パッケージと APIC の関係について説明します。

図 *1*：*APIC* パッケージのアップロード



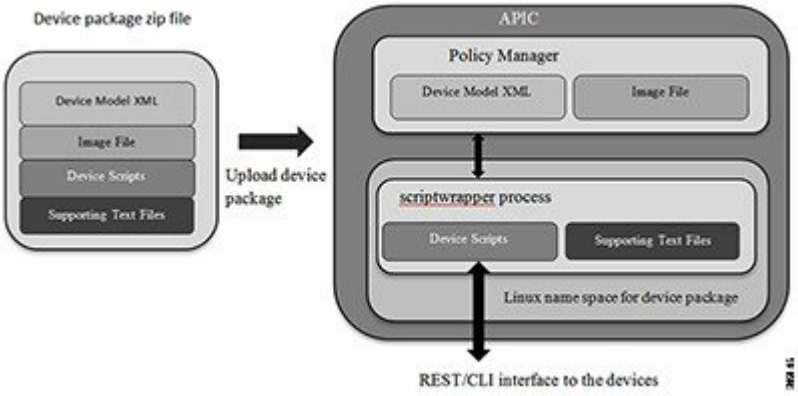
デバイス パッケージを開発すると、以下を管理できます。

- デバイスのアタッチ
- エンドポイントのアタッチ
- サービス グラフの実行
- ヘルス モニタリング
- エラー
- カウンタ

デバイス パッケージ アーキテクチャの概要

次の図では、デバイス パッケージによる Application Policy Infrastructure Controller （APIC）サービスの自動化および挿入アーキテクチャを示します。

図 *2*：デバイス パッケージ アーキテクチャ



GUI またはノースバウンド APIC インターフェイスを介してデバイス パッケージをアップロードすると、APIC がそれぞれ固有なデバイス パッケージのネームスペースを作成します。 デバイス パッケージのコンテンツが解凍され、ネームスペースにコピーされます。 デバイスパッケージのネームスペースのため作成されるファイル構造は次のとおりです。

```
root@apic1:/# ls
bin  dbin  dev  etc  fwk  install images lib  lib64  logs  pipe  sbin  tmp  usr util

root@apic1:/install# ls
DeviceScript.py DeviceSpecification.xml feature common images  lib  util.py
```

デバイス パッケージのコンテンツは `install` ディレクトリにコピーされます。

APIC はデバイス モデルを解析します。 XML ファイルで定義される管理対象オブジェクトは、ポリシー マネージャが保持する APIC の管理オブジェクト ツリーに追加されます。

デバイス パッケージで定義される Python スクリプトは、ネームスペースのスクリプトラッパープロセス内で起動されます。 ファイル システムへのアクセスは制限されます。 Python スクリプトは `/tmp` に一時ファイルを作成し、デバイス パッケージの一部としてバンドルされたすべてのテキストファイルにアクセスすることができます。ただし、ファイルで固定データを作成または保存する Python スクリプトは作成しないでください。

ログは 2 つのファイル、debug.log および periodic.log に書き込まれます。 設定 API イベント ログはすべて debug.log に書き込まれ、定期ポーリング API ログは periodic.log に書き込まれます。 ロギング フレームワークは Python ロギング フレームワークに似ています。

ログ ファイルは、ファブリック管理者として APIC にログインすることでアクセスできます。 ログ ファイルは `/data/devicescript/<vendorname-model-pkgversion>/logs` にあります。

デバイス パッケージの各バージョンはそれぞれのネームスペースで動作するため、APIC ではデバイスパッケージの複数のバージョンが共存できます。それによって、一連のデバイスを管理する特定のバージョンを選択できます。

デバッグ ログの概要

Application Policy Infrastructure Controller（APIC）はデバイス スクリプトのデバッグに使用できるログ ファイルを保持します。 ログ ファイルは、次のディレクトリ内に保持されます。

ディレクトリ	ログ ファイル
/data/devicescript	debug.log および periodic.log
/var/log/dme	• DME ログ：表示するにはルート権限が必要です。 • コア ファイル：コア ファイルのプロセス スタックを確認するバックトレースを使うにはルート権限が必要です。 • svc_ifc_*.log：表示するにはルート権限が必要です。 APIC で問題が生じた場合にのみ、これらのログ ファイルを表示する必要があります。 ログ ファイルのエクスポートの詳細については、『*Cisco ACI Troubleshooting Guide*』を参照してください。

これらのディレクトリにアクセスするには管理者権限が必要です。

第 2 章

デバイス仕様の開発

デバイス タイプの概要

Application Policy Infrastructure Controller（APIC）はネットワーク サービス デバイスを 2 つのタイプに分類します。

- GoTo：レイヤ 3 （L3）が添付されたデバイスを表します。 パケット内の送信先 MAC または送信先 IP がデバイスを識別するため、パケットは GoTo デバイスに送信されます。 通常、アプリケーション デリバリ コントローラ（ADC）または L3 ファイアウォールが GoTo デバイスを表します。
- GoThrough：トランスペアレント デバイスを表します。 送信先 MAC または送信先 IP アドレスはデバイスに対してアドレス指定されませんが、パケットは VLAN スティッチングによりデバイスを介して方向付けされます。 通常、レイヤ 2 （L2）ファイアウォールまたは侵入検知システム（IDS）デバイスが GoThrough デバイスを表します。 パケットを交換するエンド ステーションは、パス内の GoThrough （トランスペアレント デバイス）の存在を認識しません。

APIC はさらに、APIC に登録されたデバイスのインスタンスを 2 つのカテゴリに分類します。

- コンクリート デバイス：サービス デバイスのインスタンスを識別する vnsCDev で表されます。 コンクリート デバイスは物理デバイスまたは仮想デバイスです。 コンクリート デバイスには、APIC を介して設定、およびモニタリングする独自の管理 IP アドレスがあります。

- ロジカル デバイス：vnsLDevVip で表されます。 vnsLDevVip は 1 つまたは複数のコンクリート デバイスのクラスタを識別します。 ロジカル デバイスはクラスタに割り当てられた管理 IP アドレスを介してアドレス指定および管理されます。 サービス デバイスが提供するサービス機能は、ロジカルデバイスで常に実行されます。 通常、ロジカルデバイスは、アクティブ/アクティブ モードまたはアクティブ/スタンドバイ高可用性モードで実行されるデバイスのクラスタを表します。 スタンドアロン モードのデバイスを実行する場合は、ロジカル デバイスには1つだけのコンクリートデバイスが含まれます。ロジカルデバイスとコンクリート デバイスの管理 IP アドレスは同じです。 すべてのサービス動作は、常にロジカル デバイスのインスタンスで行われます。

デバイスの APIC 登録については、『*Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*』を参照してください。

サービス デバイスでは、シングル コンテキストまたはマルチ コンテキストを指定できます。 マルチ コンテキストのデバイスは複数のルーティング ドメインをサポートするため、重複する IP アドレスをサポートするデバイスは異なるルーティング コンテキストで設定されます。

シングルコンテキストのデバイスは特定のテナントに登録する必要があります。 シングルコンテキストのデバイスは複数のテナントで共有できません。マルチコンテキストのデバイスは共通のテナントの下に登録し、複数のテナントで共有することができます。

デバイス仕様の概要

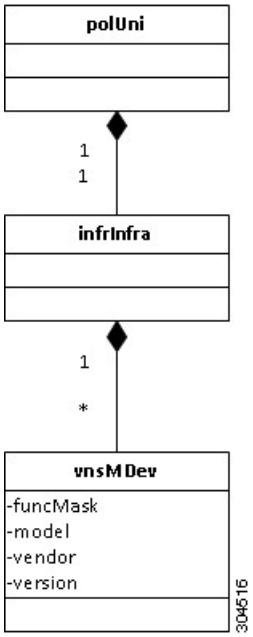
Application Policy Infrastructure Controller（APIC）の設定は、多数の管理対象オブジェクト（MO）で構成されるオブジェクト モデルとして表されます。 デバイス タイプはメタ デバイス（MDev）をルートに持つ管理対象オブジェクトのツリーによって定義されます。 新しい MDev を定義することで、デバイス固有の XML ファイルが APIC の管理対象オブジェクト モデルを拡張します。

デバイス固有のファイルはメタ デバイス（`vnsMDev`）オブジェクトを定義する必要があります。 `vnsMDev` オブジェクトには、ベンダー名、デバイス パッケージのバージョン、サポートされるデバイスのバージョン、デバイス スプリクト バインディング、そしてこれらの機能をデバイスで実行するために必要な機能とパラメータを説明するデバイス モデルといったベンダー固有の情報を記述するメタデータがあります。

デバイス パッケージの各固有バージョンは、`vnsMDev` オブジェクトのインスタンスと APIC ポリシー マネジャーとの 1 つのインスタンスの作成を生じさせます。 APIC は、`vnsMDev` オブジェクトの多数のインスタンスをサポートします。 `vnsMDev` オブジェクトは、APIC グローバル ポリシーの下のインフラ ポリシー（infraInfra で表される）内に含まれます。 グローバル ポリシーは、polUni

として表されるポリシーの集合体です。 次の図では、vnsMDev の APIC の管理対象オブジェクト階層との関係について説明します。

図 *3*：*vnsMDV* の *APIC* の管理対象オブジェクト階層との関係



デバイス モデルは vnsMDev オブジェクトに抑制されます。 デバイス仕様ファイルは次の構造を必要とします。

```
<poliUni>
   <infraInfra>
      <vnsMDev>
         <!-- device Sepcification-->
      </vnsMdev>
   </infraInfra>
</poliUni>
```

vnsMDev には次の属性がなければいけません。

- vendor：デバイス パッケージのベンダーを識別します。
- model：デバイス仕様によって管理されるデバイス モデルを識別します。
- version：デバイス パッケージのバージョンを識別します。 APIC ではデバイス パッケージの 1 つまたは複数のバージョンをアップロードできます。 APIC では、APIC に登録されたデバイス インスタンスの管理に使用するデバイス パッケージを選択することができます。

  デバイス パッケージに付加機能を追加したり、デバイスの最新リビジョンを管理するためにデバイス パッケージを更新する場合、デバイス パッケージのバージョンを増分することができます。 新しい機能を追加すると、デバイス パッケージのメジャー バージョンを増分する必要があります。 また、デバイス パッケージにバグ修正やマイナー拡張をすると、マイナー バージョンを増分する必要があります。
- funcMask：GoTo または GoThrough モードで展開されるサービス機能を、デバイス パッケージがサポートするかどうかを示します。 デバイス パッケージは、サービス挿入の GoTo モー

ドおよびGoThroughモードの両方をサポートします。 両方のモードがサポートされる場合、funcMask をカンマ区切りリストとして次の形式で定義してください。

`GoTo,GoThrough`

デバイスのサービス機能は、デバイスパッケージがこうした設定をサポートする場合のみ、GoTo または GoThrough とし展開できます。 通常、ファイアウォール デバイス パッケージの funcMask は、ファイアウォールがルート モードまたはトランスペアレント ブリッジ モードで展開されるよう、GoTo モードと GoThrough モードの両方をサポートしています。

次に、`vnsMDev` 属性の例を示します。

```
<vnsMDev vendor="Insieme"
  model="SampleDevice"
  version="1.0"
  funcMask="GoTo,GoThrough">
```

`vnsMDev` オブジェクトのインスタンスは、<vendor-model-version> 文字列によって識別されます。APIC は、一意の各 <vendor-model-version> 文字列の `vnsMDev` インスタンスを作成します。

デバイス モデルは、次の要素で構成されています。

- 一般的要素：デバイスに関する一般的な情報を定義します。 次のオブジェクトで構成されています。
  - デバイスのクレデンシャル
  - インターフェイスのラベル
- クラスタとデバイス設定要素：クラスタまたはデバイス固有の設定を定義します。 次のオブジェクトで構成されています。
  - クラスタ設定
  - デバイス設定
- 機能要素：サービス機能とその設定を示します。 設定は次のオブジェクトの下に分割されます。
  - グローバル機能デバイス設定
  - グループ設定
  - 機能設定

デバイス スクリプト

デバイスのスクリプト情報は <vnsDevScript> オブジェクトによって定義されます。 デバイスのスクリプト オブジェクトは、APIC API を定義する Python ファイルを関連付けます。 APIC はこれらの Python API を呼び出し、デバイス パッケージが定義するサービス機能をインスタンス化します。

デバイスのスクリプト オブジェクトには以下の属性が含まれます。

属性	タイプ	説明
ctrlrVersion	文字列	コントローラ API バージョンの互換性を識別します。 文字列です。 APIC API バージョンと一致する必要があります。現在の許容値は "1.0" です。
minorversion	文字列（512 文字）	スクリプトのバージョンを識別します。 デバイス モデルを変更せずそのデバイス スクリプト用だけに作られたリビジョンをトレースするには、デバイス パッケージの開発者はこのバージョン文字列を使用する必要があります。
versionExpr	文字列（512 文字）	APIC は deviceValidate コール中、この versionExpr 文字列をスクリプトに渡します。 このデバイス パッケージがサポートするデバイスのバージョンを示すために、デバイスパッケージの開発者は文字列（標準の表現で可）を定義します。

minorversion 文字列は、デバイス パッケージの無停止アップグレードを提供します。 デバイスのスクリプトだけ変更され、モデルに変更がない場合、デバイス パッケージの開発者はマイナーバージョン文字列だけアップグレードする必要があります。 マイナーバージョンだけ変更され、デバイス パッケージのバージョンが増分されなかったことを APIC が識別すると、パッケージに関連するスクリプトが再起動され、デバイス パッケージに新た一連のなファイルがバンドルされます。 管理対象オブジェクトモデルは、デバイス パッケージに指定されたデバイス モデルによって更新されません。 これによって、デバイスパッケージを使用したグラフが再レンダリングを発生させずに、スクリプトの効率的なアップグレードが可能となります。

デバイスのクレデンシャル

デバイス クレデンシャルのオブジェクトによって、デバイスと通信中でも、ベンダーはApplication Policy Infrastructure Controller（APIC）が認証用にデバイスのスクリプトに渡すクレデンシャルのタイプを指定できます。 現在は、ユーザ名とパスワードによる認証のみサポートされています。 デバイス仕様ファイルは、次のオブジェクトを定義する必要があります。

```
<vnsMCred name=”username” key=”username”/>
<vnsMCredSecret name=”password” key=”password”/>
```

デバイス仕様ファイルは、vnsMCred と vnsMCredSecret の 1 つだけのインスタンスを定義する必要があります。 デバイスの登録時に、ユーザ名およびパスワードのオブジェクトの値を入力します。 詳しくは、『*Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*』を参照してください。

インターフェイスのラベル

デバイスのインターフェイスは抽象的な方法でラベル付けする必要があります。 サービス機能におけるパケットの論理フローを表すために、機能をインターフェイスに関連付けます。 たとえば、ファイアウォール デバイスはインターフェイスを、信頼できるインターフェイス、信頼できないインターフェイス、クラスタ インターフェイス、および管理インターフェイスとしてラベル付けします。 信頼できないインターフェイスから受信したパケットをファイアウォール機能に送り、信頼できるインターフェイスから送出することも可能です。 別の例としては、デバイスがそのインターフェイスを外部、内部、HA、そして管理インターフェイスとしてラベル付けすることも可能です。 ロードバランシング機能によって、外部インターフェイスからパケットを受信し、内部インターフェイスを通じてプールにロード バランシングすることも可能です。 1 つの物理インターフェイス（仮想サービスの場合は vNIC）に、1 つまたは複数のラベルを割り当てることができます。 ラベルは、ロジカル デバイスおよびコンクリート デバイスの登録時に、デバイスのインターフェイスに割り当てられます。 1 つのアーム展開用の 1 つのインターフェイスに、複数のラベルを割り当てることができます。デバイス モデルは、インターフェイスのラベルを指定する必要があります。 ラベルは vnsMIfLb オブジェクト タイプを使用して定義します。

次に、ラベルの定義の例を示します。

```
<vnsMIfLbl name="external" shortName="ext"/>
<vnsMIfLbl name="internal" shortName="int"/>
<vnsMIfLbl name="management" shortName="mgmt" />
```

vnsMIfLbl オブジェクトには、name 属性および shortName 属性を含める必要があります。 省略名は 4 文字以下にする必要があります。 デバイス仕様は vnsMIfLbl オブジェクトの 1 つまたは複数のタイプを定義することができます。

クラスタとデバイス設定の概要

Application Policy Infrastructure Controller（APIC）によって、デバイスをスタンドアロン モード、高可用性アクティブ/スタンバイ モード、またはクラスタ イン アクティブ /アクティブ モードで展開することができます。 クラスタとデバイスの設定セクションを使用して、ベンダーはHA モードに関係なく、クラスタ内のクラスタまたは固有ノードに対応するあらゆる設定を指定することができます。 クラスタとデバイスの指定は必須ではありません。

クラスタ設定

デバイス仕様ファイルは、vnsClusterCfg と呼ばれるクラスタ設定オブジェクトを 1 つだけ定義できます。 クラスタ設定には、クラスタ全体の設定が含まれます。 クラスタに適用される設定は 1 つまたは複数のタイプ vnsMParam のオブジェクトによって示され、さらに 1 つまたは複数の vnsMFolder オブジェクトの下でロジカルにグループ化することができます。

Application Policy Infrastructure Controller（APIC）に登録されたロジカル デバイスのクラスタ設定の下で定義されたパラメータとフォルダをインスタンス化することができます。 クラスタ設定で定義された設定は、clusterModify() または clusterAudit() コール中にのみ、デバイスのスクリプトに渡されます。クラスタの下で定義された設定はサービス機能によって参照できません。クラスタ設定は、serviceModify()、serviceAudit()、serviceHealth()、または serviceCounters() API コール中にはスクリプトに渡されません。

デバイス パッケージの開発者は、vnsClusterCfg オブジェクト内のクラスタ レベル設定をすべて定義する必要があります。たとえば、クラスタ設定にはネットワーク タイムプロトコル（NTP）サーバ設定と syslog サーバ IP アドレスを含めることができます。

次に、クラスタ設定の例を示します。

```
<vnsClusterCfg name="ClusterConfig">
  <vnsMFolder key="SyslogConfig">
    <vnsMParam key="ipaddress"
      description="Syslog Server IP address"
      dType="str"
      validation="isIPAddress"/>
  </vnsMFolder>

  <vnsMFolder key="NTPConfig">
    <vnsMParam key="ipaddress"
      description="NTP Server IP address"
      dType="str"
      validation="isIPAddress"/>
  </vnsMFolder>
</vnsClusterCfg>
```

デバイス設定

デバイス仕様ファイルは、デバイス固有の設定を含む vnsDevCfg の 1 つのインスタンスを含むことができます。 vnsDevCfg は vnsClusterCfg に含まれています。 デバイス固有の設定は、1 つまたは複数の vnsMParam で表され、これをさらに 1 つまたは複数の vnsMFolder 下でグループ化できます。

デバイス設定の下で定義された設定は、ロジカル デバイス内のコンクリート デバイス登録時のユーザによってインスタンス化されます。 デバイス設定は、deviceAudit()、deviceModify()、deviceHealth()、deviceCounters() のコール時にのみ、デバイスのスクリプトに渡されます。 デバイス設定は、clusterModify() のコール、または clusterAudit() のコール時にはサービス機能から参照できません。

デバイス設定は、デバイスの HA モード、クラスタのピア IP アドレス、またはクラスタ内の特定のデバイスにプッシュされるポート チャネル（LACP）設定などの設定を含むことができます。

次に、デバイスの設定の例を示します。

```
<vnsClusterCfg name="ClusterConfig">
  <vnsDevCfg name="DevCfg">
    <vnsMFolder key="HighAvailabilityCfg" cardinality="n">
      <vnsMParam key="peerIP"
        description="HA Pair peer IP address"
        dType="str"
        validation="isIPAddress"/>
    </vnsMFolder>
  </vnsDevCfg>
</vnsClusterCfg>
```

機能的設定の概要

デバイスパッケージおよびデバイスはさまざまなサービス機能をサポートできます。通常、デバイスのパケット転送を変換したり、それに作用する機能はサービス機能として表すことができます。 たとえば、SSL オフロード、VPN、サーバ ロード バランシングおよび Web アプリケーション フィルタリングは、デバイスがサポートする機能としてモデル化できます。 1 つまたは複数のこうした機能を、デバイス仕様ファイルでモデル化することができます。

機能は vnsMFunc オブジェクトで表されます。 vnsMFunc オブジェクトには name 属性があります。デバイスパッケージ内で定義される各機能は、一意の名前でなければなりません。その名前が、MDev のインスタンスの下で定義される機能の検索に使用されます。

vnsMFunc オブジェクトには、次のオブジェクトが含まれている必要があります。

- vnsMConn
- vnsImage

特定のサービス機能を実行するのに必要なパラメータは、次のカテゴリの下で定義できます。

- 機能
- グループ
- デバイス グローバル

次に、機能設定の構造の例を示します。

```
<poliUni>
   <infraInfra>
      <vnsMDev>
        <!-- Generic Part -->

        <!-- Device Credentials -->
        <vnsMCred name=”username” key=”username”/>
        <vnsMCredSecret name=”password” key=”password”/>

        <!-- Interface Labels -->
        <vnsMIfLbl name="external" shortName="ext"/>

        <!-- Cluster Configuration -->
        <vnsClusterCfg name="ClusterCfg">

          <!-- Device Configuration -->
          <vnsDevCfg name="DeviceConfig">

          </vnsDevCfg>
        </vnsClusterCfg>

        <!-- Functional Configuration -->

        <!-- Global Functional Device Configuration -->
        <vnsMDevCfg>
        </vnsMDevCfg>

        <!-- Group Configuration -->
        <vnsGrpCfg>
        </vnsGrpCfg>

        <!—Function configuration: Could be one or more such configuration -->
        <vnsMFunc>
```

```
        </vnsMFunc>
      </vnsMdev>
   </infraInfra>
</poliUni>
```

コネクタ オブジェクト

1 つの機能には少なくとも 1 つのコネクタ オブジェクト、vnsMConn が必要です。 1 つまたは複数の機能をリンクし、サービス グラフを形成するためにコネクタ オブジェクトが使用されます。機能がトランジット機能の場合は、少なくとも 2 つのコネクタが必要です。 機能がコレクタのようなスタブ機能の場合は、1 つのコネクタでかまいません。 通常、パッシブ モードであり、デバイスにコピーされるパケットをキャプチャする IDS デバイスだけが、キャプチャ機能に定義された1 つのコネクタを持っています。 ファイアウォール、ロード バランサ、SSL オフロードなど、ほかの機能にはすべて 2 つ以上のコネクタがあります。 現在、Application Policy Infrastructure Controller（APIC）は 1 機能ごとに最大 2 つのコネクタをサポートしているので、すべてのトランジット機能で入力コネクタと出力コネクタを定義することが可能です。

コネクタは次の属性を持ちます。

属性	必須	説明
name	Yes	コネクタの名前を指定します。機能内のすべてのコネクタの名前は一意でなければなりません。
encType	Yes	コネクタのカプセル化タイプを指定します。 この属性はコネクタのトラフィックに使用されるカプセル化で、vlan または vxlan の値として指定されます。 値によって、パケットがカプセル化されてネットワークからデバイス VLAN または VXLA（カプセル化）に送られるかが指定されます。 仮想デバイスではカプセル化が仮想スイッチによって削除されることがあり、VLAN または VXLAN のカプセル化ヘッダーが仮想サービス デバイスによって認識されない場合があります。現在、APIC がサポートするのは VLAN カプセル化だけです。
dir	No	コネクタの方向を指定します。方向は input としても output としても指定できます。

属性	必須	説明
cardinality	No	ある機能が任意のコネクタ タイプの複数のインスタンスをサポートする場合、デバイス モデルは濃度を n に設定することでこれを明示的に指定できます。 デフォルトでは、濃度は `1` です。
notification	No	エンドポイントまたはネットワークの接続/切断通知が、その機能のために作成されるのを許可します。 この属性は、コネクタに直接的または間接的に接続されたエンドポイント グループ（EPG）のためにエンドポイントまたはサブネットの関連付けが変化した場合、APIC がデバイスのスクリプトを呼び出すかどうかを指定するのに使用します。 通知は、次の値を指定できます。 • `none` • `subnet` • `endpoint` 通知属性が指定されていない場合は、デフォルトにより `none` になります。つまり、APIC はネットワークまたはエンドポイント API の接続も切断も行いません。

1 つのコネクタには、1 つの `vnsRsInterface` オブジェクトだけが含まれていなければいけません。このオブジェクトは、`vnsMIfLbl` を使って定義されたラベルによって識別される特定のインターフェイスにコネクタを関連付けます。 APIC はサービス機能の実行中、この関係を使って特定のインターフェイス情報を渡します。 詳細については、ファブリック接続, （75 ページ） を参照してください。

イメージ

機能は、デバイス パッケージにバンドルされている gif イメージ ファイルにリンクする必要があります。 Application Policy Infrastructure Controller（APIC）はグラフ内の機能のため GUI でイメージを表示します。 このイメージは、特定のベンダデバイスで実行されるグラフでの特定の機能のグラフィカル ID となります。

イメージへのリンクは、次の例に示すように、1 つの vnsMImage オブジェクトをインスタンス化することで実行されます。

```
<vnsMImage
name="insieme.gif"/>
```

機能設定

デバイス パッケージの開発者は、機能オブジェクトの下でサービス機能を設定するために必要なパラメータを定義できます。 vnsMFunc 下で定義されるすべてのパラメータは、特定の機能の下で範囲が決定されます。 機能下で定義されたパラメータは、1 つまたは複数のフォルダでさらにロジカルにグループ化することができます。

機能下で定義されたパラメータおよびフォルダは、機能のインスタンスが持続する場合は持続します。 Application Policy Infrastructure Controller（APIC）は、機能のインスタンスが削除されると、機能下で定義されたパラメータおよびフォルダを削除します。

機能下にあるパラメータおよびフォルダは、同じグラフまたは同じデバイスで実行される別のグラフ内の他の機能と共有したり、参照したりすることはできません。 機能下で定義されたパラメータおよびフォルダには、各機能のインスタンスごとに、デバイスにおける一意のインスタンスがなければいけません。 パラメータ範囲および機能のコンテキスト内に制限されたフォルダは、C 言語のローカル変数と類似しています。

次に、サービス機能のパラメータの定義の例を示します。

```
<vnsMFunc name="SLB">
     <vnsMImage name="insieme.png"/>

      <vnsMConn name="external"
                dir="input"
                encType="vlan"
                 notifications="endpoint">
               <vnsRsInterface tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mIfLbl-external"
 />
       </vnsMConn>

       <vnsMConn name="internal"
                 dir="output"
                 encType="vlan"
                 notifications="endpoint">
                <vnsRsInterface tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mIfLbl-internal"
 />
        </vnsMConn>

        <vnsMFolder key="VServer"
                    scopedBy="epg">
                <vnsMParam key="vservername"
                           description="Name of VServer"
                           mandatory="true"
                           dType="str"
```

```
                validation="isAlpha"/>
            <vnsMParam key="port"
                description="Port for Virtual server"
                validation="isL4Port"/>
            <vnsMParam key="persistencetype"
                description="persistencetype"/>

            <vnsMParam key="servicename"
                    description="Service bound to this vServer"/>
            <vnsMParam key="servicetype"
                    description="Service bound to this vServer"
                dType="str"
                    validation="isProtocol"/>
            <vnsMParam key="clttimeout"
                description="Client timeout"/>

        </vnsMFolder>
</vnsMFunc>
```

グループ設定

グループ設定下で定義されるパラメータとフォルダはすべて、グラフ内の複数の機能上で共有することができます。 デバイス パッケージの開発者は、グループ設定下にある 1 つのグラフ内の 1 つのデバイスで実行される複数の機能上で共有されるよう、パラメータとフォルダを定義することができます。

グループ設定内のパラメータとフォルダは、グラフのインスタンスの下にあります。 グラフのインスタンス内にある機能はすべて、設定を共有し、参照することができます。

グループ設定下で定義されるオブジェクトは、グラフのインスタンスが持続するかぎり持続します。 Application Policy Infrastructure Controller（APIC）は、グラフのインスタンスが削除されると、グループ設定下で定義されるパラメータおよびフォルダを削除します。 グループ設定下で定義されるパラメータにはすべて、デバイスのグラフごとに一意のインスタンスがなければいけません。パラメータを同じデバイスで実行される他のグラフのインスタンスで共有したり、参照したりすることはできません。

グループ設定は `vnsGrpCfg` オブジェクトで表されます。 `vnsMDev` の下には、`vnsGrpCfg` の 1 つの定義しかありません。 あるグループの下にあるすべてのグループ パラメータとフォルダは、`vnsGrpCfg` オブジェクト内に含まれなければいけません。

グループ設定下で定義されるパラメータおよびフォルダは、C 言語の静的変数と類似しています。変数は機能を超えて持続します。

グローバル機能の設定

`vnsMDev` 設定下で定義されるパラメータおよびフォルダはすべて、複数グラフ上で複数の機能を共有することができます。 デバイス パッケージの開発者は、`vnsMDevCfg` の下で単一のデバイスで実行される、複数グラフ上の複数機能上で共有されるよう、パラメータおよびフォルダを定義することができます。

`vnsMDev` 設定下で定義されるオブジェクトは、パラメータまたはフォルダに参照するグラフのインスタンスが少なくとも 1 つあれば、持続します。 Application Policy Infrastructure Controller（APIC）

は、すべてのグラフのインスタンス上のすべての機能が特定のデバイスから削除されると、vnsMDev 下で定義されるパラメータおよびフォルダを削除します。

マルチ コンテキストのデバイスでは、グローバル設定にはコンテキストごとに一意のインスタンスがなければいけません。 vnsMDev 下で定義されるパラメータおよびフォルダは、マルチ コンテキスト上では共有できません。

vnsMDev 設定下で定義されるパラメータおよびフォルダは、C 言語のグローバル変数と類似しています。

通常、インターフェイスで設定された IP アドレス、ルート、サブネットなどのネットワーク属性にはグローバル スコープがあります。 APIC によって割り当てられたカプセル化のタグはグローバル スコープで、複数のパラレルな機能を複数のグラフ上の同じネットワーク上で展開することが可能です。

関係

サービス機能はグループまたは vnsMDevCfg 下で定義される特定のパラメータまたはフォルダを参照することができ、それによってグループまたはデバイス スコープ下で定義されるパラメータまたはフォルダのインスタンスを機能が使用可能になります。 フォルダとの関係は vnsMRel オブジェクトを使用して定義されます。 vnsMRel オブジェクトは vnsMFolder オブジェクト内にのみ存在できます。 1 つのフォルダでは、1 つまたは複数の関係オブジェクトを定義できます。

vnsMRel オブジェクトには次の属性があります。

属性	必須	説明
key	Yes	オブジェクトを特定します。各オブジェクト キーは、含む側のオブジェクト内に一意の値が必要です。キーには英数字、「_」または「-」のみを使用できます。 キーでは他の文字は使えません。 Application Policy Infrastructure Controller（APIC）はキーを、含む側のオブジェクト内の特定のオブジェクトを検索するために使用します。 キー サイズは 512 文字以内に制限されています。

属性	必須	説明
Description	Yes	この設定項目の説明を保持します。 APIC GUI で使用する説明フィールドで、ユーザにヘルプを提供します。 デバイス パッケージの開発者は、関係の正確な説明と目的を入力する必要があります。 説明フィールドのサイズは 512 文字以内に制限されています。
mandatory	No	この関係が必須かどうかを指定します。 このプロパティはブール値（Yes または No）です。 デフォルトでは、関係は明示的に特定されないかぎり必須ではなく、ユーザが関係オブジェクトを特定する必要はありません。 デバイスで機能を実行するための任意の関係は必要ありません。
cardinality	No	この関係の発生数を指定します。 デフォルトでは、含まれているオブジェクトの下では関係のインスタンスは 1 つだけ許可されています。関係オブジェクトの 1 つ以上のインスタンスをユーザがインスタンス化することが許されている場合、デバイス仕様ファイルで関係を `cardinality="n"` で定義する必要があります。

`vnsMRel` オブジェクトは、オブジェクトがどの関係を参照しているかを識別する vnsRsTarget オブジェクトを含みます。 ターゲットは、デバイス仕様ファイルで定義されるオブジェクトの完全修飾キーです。 `vnsMRel` オブジェクトには、`vnsRsTarget` オブジェクトのインスタンスを 1 つだけ含めることができます。

次に、関係オブジェクトの定義の例を示します。

```
<vnsMRel key="ServerConfig">
   <vnsRsTarget tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mDevCfg/mFolder-Server"/>
 </vnsMRel>
```

上記の例は、機能内で定義された ServerConfig に、vnsMDevCfg 下で定義されたサーバ フォルダのインスタンスとの関係があることを示します。デバイス設定下のフルパスで認定されたターゲットフォルダのインスタンス名を指定することで、関係をインスタンス化できます。サービス機能がデバイスで実行されると、APIC が関係によって参照されるフォルダの特定のインスタンスを検索します。APIC が一致するインスタンスを見つけると、サービス API コールで渡される設定ディクショナリにそのフォルダを含めます。 APIC はまた、機能設定ディクショナリの一部として関係のインスタンスを渡します。 API で渡される設定ディクショナリの例については、デバイス スクリプトの開発, （45 ページ）を参照してください。

パラメータの範囲および API 設定ディクショナリ

vnsMDevCfg、vnsGrpCfg、または vnsMFunc 下で定義されるパラメータおよびフォルダはすべて、serviceAudit()、serviceModify()、serviceHealth()、および serviceCounters() 機能コール時にのみ、デバイスのスクリプトに渡されます。 vnsMDevCfg オブジェクトで定義されるパラメータおよびフォルダは、そのパラメータおよびフォルダを参照する関係オブジェクトのサービス機能がある場合のみ、サービス API コールで渡されます。

パラメータ オブジェクトおよびフォルダについて

クラスタ、デバイス、機能設定は、1 つ以上の vnsMParam オブジェクトによって定義されます。これらのオブジェクトは、vnsMFolder のオブジェクトとして表される 1 つまたは複数のフォルダでロジカルにグループ化できます。

パラメータのオブジェクト

設定パラメータは vnsMParam オブジェクト タイプで表されます。 デバイス パッケージは 1 つまたは複数の vnsMParam オブジェクトを持つことができます。 パラメータのオブジェクトには次の属性が含まれます。

属性	必須	説明
key	Yes	メタ パラメータのキーを指定します。 このプロパティは、パラメータを一意に識別します。 各パラメータ キーは、含む側のオブジェクト内に一意の値が必要です。 キーには英数字、「_」または「-」のみを使用できます。 キーでは他の文字は使えません。 Application Policy Infrastructure Controller（APIC）はキーを、通常 `vnsMFolder` オブジェクトであるコンテナ オブジェクト内の特定のオブジェクトを検索するために使用します。 キー サイズは 512 文字以内に制限されています。
Description	Yes	この設定項目の説明を保持します。 APIC GUI で使用する説明フィールドで、ユーザにヘルプを提供します。 デバイス パッケージの開発者は、パラメータの正確な説明と目的を入力する必要があります。 説明フィールドのサイズは 512 文字以内に制限されています。
mandatory	No	このパラメータが必須かどうかを指定します。 このプロパティはブール値（Yes または No）です。 デフォルトでは、パラメータは明示的に特定されないかぎり必須ではありません。

属性	必須	説明
dType	No	このパラメータのデータ タイプを指定します。 これは、次の値を指定できます。 • `int` • `real` • `str` dType が指定されていない場合、パラメータのデフォルトは `str` です。
validation	No	このパラメータの値を APIC が検証するために使う検証式を指定します。 検証文字列は255 文字を超えることはできません。 検証が指定されている場合、dType が `str` である必要はありません。 検証文字列は複合または比較のオブジェクト名を参照します。 詳細については、パラメータの検証, （28 ページ）を参照してください。

属性	必須	説明
cardinality	No	このパラメータの発生数を指定します。 デフォルトでは、含まれているオブジェクトの下でパラメータのインスタンスは1つだけ許可されています。 パラメータ オブジェクトの 1 つ以上のインスタンスをユーザがインスタンス化することが許されている場合、デバイス固有ファイルがパラメータを `cardinality="n"` で定義する必要があります。 たとえば、ルートと呼ばれるパラメータ オブジェクトを持つデバイスで複数の静的ルータをインスタンス化できる場合、ルートのパラメータの濃度を `cardinality="n"` に設定してください。

次に、パラメータ オブジェクトの定義の例を示します。

```
 <vnsMParam key="vservername"
  description="Name of VServer"
  mandatory="true"
  dType="str"
  validation="isAlpha"/>

<vnsMParam key="subnetipaddress"
  description="Subnet IPAddress of the Device"
  dType="str"
  cardinality=”n”
  validation="isIPAddress"/>
```

フォルダ

設定パラメータは、フォルダの下でロジカルにグループ化できます。 フォルダは、1 つまたは複数のフォルダおよびパラメータを含むことができます。 フォルダは vnsMFolder オブジェクトによって表され、次の属性があります。

属性	必須	説明
key	Yes	メタ フォルダのキーを指定します。このプロパティは、フォルダを一意に識別します。 各フォルダのキーは、含む側のオブジェクト内で固有の値でなければいけません。 キーには英数字、「_」または「-」のみを使用できます。 キーでは他の文字は使えません。 Application Policy Infrastructure Controller（APIC）はキーを、含む側のオブジェクト内の特定のオブジェクトを検索するために使用します。 キー サイズは 512 文字以内に制限されています。 <vnsMDevCfg>、<vnsGrpCfg>、または <vnsMfunc> 下で定義される最上位フォルダのキーは、デバイス パッケージ内で一意である必要があります。
Description	Yes	この設定項目の説明を保持します。 APIC GUI で使用する説明フィールドで、ユーザにヘルプを提供します。 デバイス パッケージの開発者は、フォルダの正確な説明と目的を入力する必要があります。 説明フィールドのサイズは512文字以内に制限されています。

属性	必須	説明
scopedBy	No	

属性	必須	説明
		この設定フォルダの範囲を指定します。 この属性は、機能のインスタンス化時に管理情報ツリー（MIT）のどこでこのフォルダの値を検索するかを指定します。 APIC は、異なるオブジェクト下で定義されたインスタンスがどのグラフに関連しているかを検索し、値を解決します。 scopedBy 属性には次の値を含めることができます。 • tenant：フォルダがテナント下だけでインスタンス化されます。 • ap：フォルダがアプリケーション プロファイルまたはテナント下だけでインスタンス化されます。 • bd：フォルダが bd またはテナント下だけでインスタンス化されます。 • epg：フォルダがエンドポイントのグループ（EPG）、ブリッジ ドメイン、アプリケーション プロファイル、またはテナント下だけでインスタンス化されます。 • none デバイス パッケージの開発者は、解決を上位レベルに制限できます。 デフォルトでは、scopedByは「none」と定義されています。つまり、特定のフォルダをインスタンス化できる場所でデバイス パッケージは制限を設けません。 APIC のユーザは、テナント、アプリケーション プロファイル、ブリッジ ドメイン、または EPG 下で

属性	必須	説明
		フォルダのインスタンスを定義できます。
cardinality	No	このフォルダの発生数を指定します。 デフォルトでは、含まれているオブジェクトの下ではフォルダのインスタンスは1つだけ許可されています。 フォルダのオブジェクトの1つまたは複数のインスタンスをユーザがインスタンス化することが許されている場合、デバイス仕様ファイルでフォルダを `cardinality="n"` で定義する必要があります。

次に、フォルダのオブジェクトの定義の例を示します。

```
<vnsMFolder key="Server"
  scopedBy="epg">
  <vnsMParam key="servername"
    description="Server Name"
    dType="str"
    validation="isAlpha"/>
  <vnsMParam key="domain"
    description="Domain name of the server"/>
  <vnsMParam key="ipaddress"
    description="Server IP address"
    dType="str"
    validation="isIPAddress"/>
</vnsMFolder>
```

パラメータの検証

Application Policy Infrastructure Controller（APIC）は、`vnsComparison` および `vnsComposite` オブジェクトを使用してパラメータ検証を行うことができます。 デバイスパッケージの開発者は基本または複合比較を使って、すべての文字列タイプのパラメータの検証を定義し、関連付けることができます。

基本比較（`vnsComparison`）は次の演算を行うことができます。

- 等しい：`eq`（デフォルト）
- 等しくない：`ne`
- 未満：`lt`
- より大きい：`gt`
- 以上：`ge`

- 以下：le
- 一致：match（標準形式が必要）

比較オブジェクトの vnsComparison は、vnsMDev、vnsMFunc、vnsMFolder、vnsMParam、または vnsComposite オブジェクト下で定義されています。 vnsComparison オブジェクトには次の属性があります。

属性	必須	説明
name	Yes	比較アサーションの名前を保持します。
cmp	Yes	比較演算子を定義します。 • eq：等しい、デフォルト。 • ne：等しくない。 • lt：より小さい。 • gt：より大きい。 • ge：以上 • le：以下 • match：一致。 match 比較には、正規表現が必要です。

次の例では、正規表現一致を使って、パラメータが IP アドレスを検証します。

```
<vnsMParam key="vipaddress"
           description="VIP IPAddress"
           dType="str"
           validation="isIPAddress"
/>

<vnsComparison name="isIPAddress"
               cmp="match"

value="([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])"
/>
```

複合比較（vnsComposite）では、次のタイプの比較が実行されます。

- すべて一致（デフォルト）：複合オブジェクトで定義される比較オブジェクトすべてがパラメータ値と一致したとき、検証がパスします。
- どれか一致：複合オブジェクト内で定義される比較オブジェクトの 1 つがパラメータ値と一致したとき、検証がパスします。
- 1つだけ一致：比較オブジェクトの1つがパラメータ値と一致したとき、検証がパスします。

複合オブジェクトは、1 つまたは複数の vnsComparison オブジェクトを含むことができます。 複合オブジェクトは、vnsMDev、vnsMFunc、vnsMFolder または vnsMParam 下で定義できます。 vnsComposite は次の属性を持ちます。

属性	必須	説明
name	Yes	複合の名前を保持します。
cmp	Yes	実行する比較のタイプを定義します。 次の値が必要です。 • and：検証で成功として返すには、複合内に含まれるすべての比較文字列が一致する必要があります。 and タイプはデフォルトの比較です。 • or：複合内に含まれる比較文字列のいずれかが一致すると、検証で成功として返されます。 • one：複合内に含まれる比較文字列が 1 つのみ一致すると、検証で成功として返されます。 この演算子によって、パッケージの開発者は相互排除を定義できます。

次の例では、含まれる値のいずれかの一致を要素が定義します。

```
<vnsComposite name="isProtocol" comp="or">
    <vnsComparison name="ip" cmp="eq" value="IP" />
    <vnsComparison name="tcp" cmp="eq" value="TCP" />
    <vnsComparison name="udp" cmp="eq" value="UDP" />
    <vnsComparison name="http" cmp="eq" value="HTTP" />
 </vnsComposite>

 <vnsComposite name="yesNo" comp="one">
     <vnsComparison name="yes" cmp="eq" value="YES" />
     <vnsComparison name="No" cmp="eq" value="NO" />
 </vnsComposite>
```

エラー コード

デバイス仕様ファイルで、エラーの本質と可能な修正措置を説明する、ヘルプ文字列付きのエラーコードを定義することができます。 デバイスのスクリプトで、パラメータまたはフォルダによる

機能実行時の問題が検出されると、スクリプトは特定のエラー コードに問題のあるオブジェクトのパスを返すことができます。 Application Policy Infrastructure Controller（APIC）は、デバイス仕様ファイルで定義されるエラー コードを参照し、エラーを表示すると同時に説明および対処方法を選択します。エラーコードを定義するとエラーの原因と修正措置が説明され、ユーザがエラーを解決することができます。

エラー コードはvnsMDfctsオブジェクト下で定義されます。 デバイス仕様には、vnsMDevの下にvnsMDfctsのインスタンスが1つあります。 vnsMDfctsオブジェクトは、vnsMDfctオブジェクトに記述されている1つまたは複数のエラー コードを含むことができます。

vnsMDfctオブジェクトには次の属性を含みます。

属性	必須	説明
code	Yes	一意の問題を識別するaAユニット16値を指定します。
Description	Yes	問題について説明します。APIC GUIで使用する説明フィールドで、ユーザにヘルプを提供します。 デバイス パッケージの開発者は、正確な説明を入力する必要があります。 フィールド サイズは512文字以内に制限されています。
htmlFile	No	ユーザが問題を把握し、解決するためのオンライン ヘルプへのURLリンクを指定します。 フィールド サイズは512文字以内に制限されています。
recAct	Yes	推奨処置を指定します。 APIC GUIは、このフィールドを使って、ユーザに推奨処置を提供します。 デバイス パッケージの開発者は、問題を解決するための正確な推奨処置を入力する必要があります。 フィールド サイズは512文字以内に制限されています。

次に、エラーのオブジェクトの定義の例を示します。

```
<vnsMDfcts>
     <vnsMDfct code="100"
               recAct="Configure a Netmask for the vipaddress"
```

```
            descr="VIP requires vipaddress and NetMask"/>
     <vnsMDfct code="200"
            recAct="Configure a relation to VIP Folder"
            descr="A function should have a valid relations to a VIP folder that is
specifying the VIP Address and Netmask"/>
</vnsMDfcts>
```

機能プロファイル

APIC によって、デバイス パッケージの開発者はデバイス モデル内の機能プロファイルを定義することができます。 機能プロファイルは、特定のアプリケーションに適した 1 つまたは複数の機能用のテンプレートです。 機能プロファイルは、グラフを定義する機能に有効なデフォルトによって、デバイスパッケージ内の抽象的グラフを定義することに相当します。 ユーザはサービスグラフの定義時にデバイス パッケージでビルトイン機能プロファイルを参照し、ビルトイン機能プロファイルを活用できます。 機能プロファイルを使用すれば、特定のアプリケーション用のサービス機能をインスタンス化するためにユーザが入力しなければならないパラメータの数が減少します。デバイスパッケージの開発者は、適用できるだけの数の機能プロファイルを含めることができます。

次に、デバイス パッケージの機能プロファイルの定義の例を示します。

```
<vnsAbsFuncProfContr name = "FunctionProfiles">

         <vnsAbsFuncProfGrp name = "Function Profiles for Service graph
                             for an Application 1 ">
             <vnsAbsFuncProf name = "Function 1 Name">
                 <vnsRsProfToMFunc
                        tDn="uni/infra/mDev-<vendor-model-version>/mFunc-function1"/>

                 <vnsAbsDevCfg>
                     <vnsAbsFolder key="Folder_Key"
                                   name="Folder Instance name" scopedBy="epg">
                         <vnsAbsParam name="Param Instance name"
                                      "key="Param Name" value="Value"/>
                         …
                     </vnsAbsFolder>
                      …

                 </vnsAbsDevCfg>

                 <vnsAbsFuncCfg>

                      <vnsAbsFolder key=”Folder_Key"
                                    name="Folder Instance name" scopedBy="epg">
                       <vnsAbsCfgRel key="relation_key"
                                     name=”rel name” targetName="targetValue"/>
                     </vnsAbsFolder>
                  …
                 <vnsAbsFuncCfg>

             </vnsAbsFuncProf>

             <vnsAbsFuncProf name = "Function 2 Name">
                 <vnsRsProfToMFunc
                        tDn="uni/infra/mDev-<vendor-model-version>/mFunc-function2"/>
                 ….
             <vnsAbsFuncProf>

         </vnsAbsFuncProfGrp>

         <vnsAbsFuncProfGrp name = "Function Profiles for
                             Service graph for an Application 2 ">
             …
         </vnsAbsFuncProfGrp>
```

```
</vnsAbsFuncProfContr>
```

機能プロファイルの定義は <vnsAbsFuncProfContr> に含まれています。 それぞれの一義的アプリケーションのプロファイルは<vnsAbsFuncProfGrp> で指定されます。 <vnsAbsFuncProfGrp> 名は、テンプレートが定義されるアプリケーションに対して直観的であるべきです。 たとえば、機能プロファイルが web アプリケーションのロード バランシング機能用の場合、vnsAbsFuncProfGrp は「Web アプリケーション仮想サーバ」という名前であるべきです。

<vnsAbsFuncProfGrp> で指定される機能プロファイルは、必要に応じて 1 つまたは複数の機能を含むことができます。 グラフが同デバイス上の複数の機能の連動を必要とする場合、プロファイルは <vnsAbsFuncProfGrp> 内でそれらの機能のデフォルトを定義できます。 プロファイル内の各機能設定は <vnsAbsFuncProf> に含まれています。

各 vnsAbsFuncProf には、デバイス モデルによって定義される機能への 1 つの関係があります。 関係は機能プロファイルでインスタンス化される機能のタイプを指定します。 機能との関係は vnsAbsFuncProf に含まれるオブジェクトの vnsRsProfToMFunc によって定義されます。 vnsRsProfToMFunc には、機能オブジェクトの完全修飾名で機能を特定する tDn 属性があります。 この例は、デバイス モデル内の機能を指定する tDn を示しています。

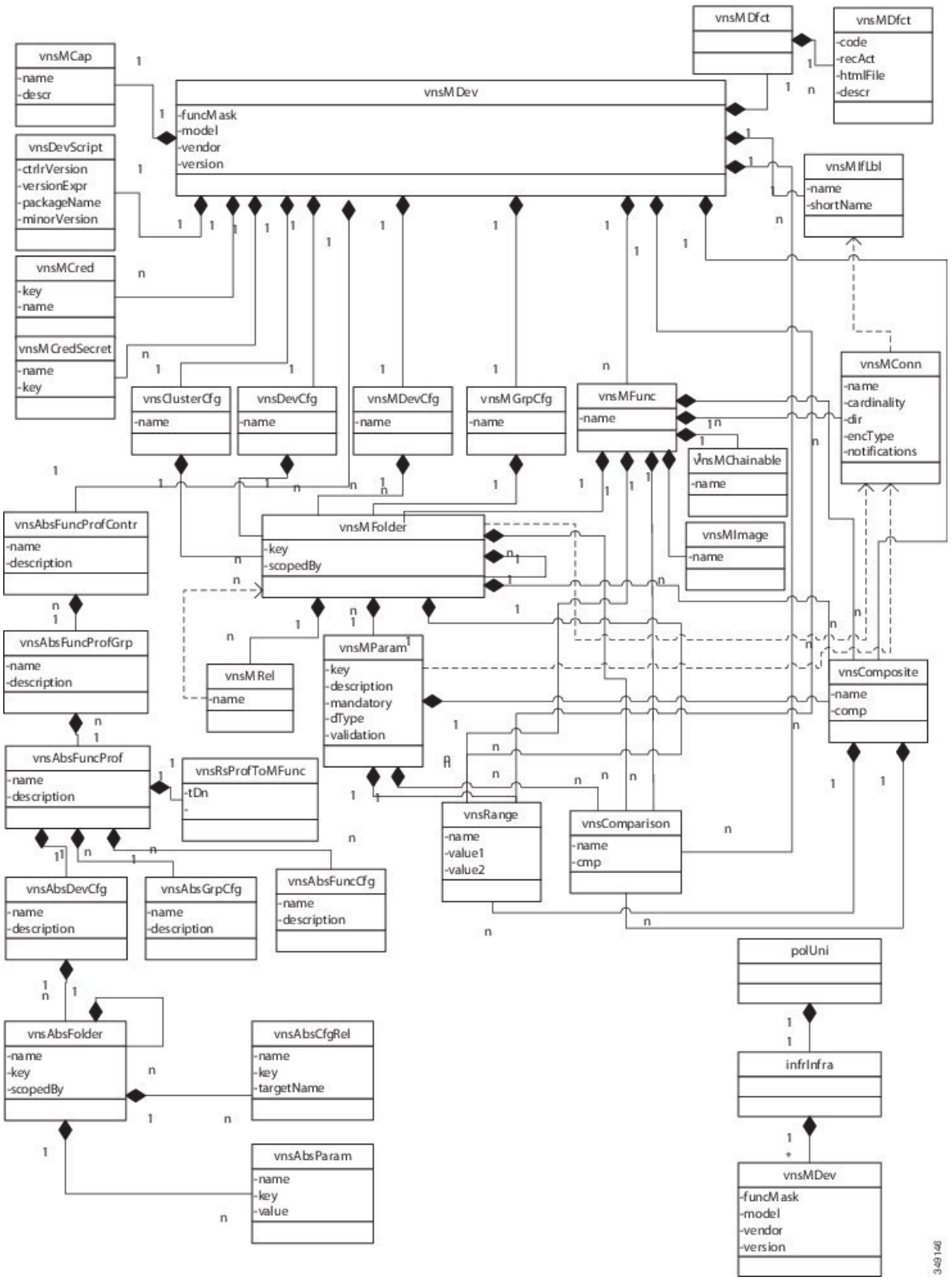
これらの機能のパラメータを設定するメカニズムは、APIC でのサービス グラフの作成と同一です。 機能プロファイルのパラメータ、関係およびフォルダは、vnsMDevCfg、vnsGrpCfg、および vnsFuncCfg 下で定義されるパラメータ、関係およびフォルダのインスタンスである場合もあります。

ノースバウンド API を介したサービス グラフの作成については、『*Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*』を参照してください。

管理対象オブジェクト モデル

次の図に、デバイスを表すオブジェクト モデルを示します。

図 *4* : **管理対象オブジェクト モデル**


vnsMDfct
vnsMDfct
-code
-recAct
-htmlFile
-descr
vnsMCap
-name
-descr
vnsMDev
-funcMask
-model
-vendor
-version
vnsDevScript
-ctrlrVersion
-versionExpr
-packageName
-minorVersion
vnsMIfLbl
-name
-shortName
vnsMCred
-key
-name
vnsMCredSecret
-name
-key
vnsMConn
-name
-cardinality
-dir
-encType
-notifications
vnsClusterCfg
-name
vnsDevCfg
-name
vnsMDevCfg
-name
vnsMGrpCfg
-name
vnsMFunc
-name
vnsMChainable
-name
vnsMImage
-name
vnsAbsFuncProfContr
-name
-description
vnsMFolder
-key
-scopedBy
vnsAbsFuncProfGrp
-name
-description
vnsMRel
-name
vnsMParam
-key
-description
-mandatory
-dfType
-validation
vnsComposite
-name
-comp
vnsAbsFuncProf
-name
-description
vnsRsProfToMFunc
-tDn
vnsRange
-name
-value1
-value2
vnsComparison
-name
-cmp
vnsAbsDevCfg
-name
-description
vnsAbsGrpCfg
-name
-description
vnsAbsFuncCfg
-name
-description
polUni
vnsAbsFolder
-name
-key
-scopedBy
vnsAbsCfgRel
-name
-key
-targetName
infrInfra
vnsAbsParam
-name
-key
-value
vnsMDev
-funcMask
-model
-vendor
-version
349148

次の表では、オブジェクト モデルでのオブジェクトについて説明します。

コンポーネント	説明
vnsMDev	サービスデバイスタイプのメタデータの定義が含まれます。 メタデータは、ベンダー名、デバイス モデル、およびデバイスのバージョンを含むベンダー固有データを含みます。 サービス デバイスは、GoTo および GoThrough デバイスとして分類されます。 デバイスは、パケットがデバイスの MAC アドレスまたは IP アドレスにアドレス指定されている場合、GoTo デバイスです。 パケットがパス内挿入によってデバイスで送信され、パケットがデバイスの MAC アドレスまたは IP アドレスにアドレス指定されていない場合、デバイスは GoThrough デバイスと見なされます。 トランスペアレント モードのファイアウォールは GoThrough デバイスの例です。 デバイス パッケージおよびデバイス固有モデルは、GoTo モードと GoThrough モード両方のデバイスをサポートできます。 デフォルトでは、デバイス仕様は GoTo モードのデバイスを表すと見なされます。 デバイス固有ファイルは、両方のモードをサポートするようにも、GoThrough モードだけをサポートするようにも、次の属性を使って変更できます。 `funcMask: "GoTo,GoThrough"`
vnsMCred	デバイスにユーザを認証するために必要なクレデンシャルを表します。 たとえば、`key`はキー ベースの認証スキームに使用されます。 このモデルは、このようなキー ベースのクレデンシャル認証のメタ情報を詳しく説明します。
vnsDevScript	デバイスのスクリプト ハンドラを表します。 この管理対象オブジェクトには、名前、パッケージ名とバージョンを含むスクリプト ハンドラの関連属性のメタ情報が含まれます。
vnsClusterCfg	クラスタ設定フォルダおよびパラメータが含まれます。 クラスタ設定は、デバイスのクラスタで実行されるグラフとは別に、デバイスのクラスタの機能性に影響を及ぼします。
vnsDevCfg	デバイス固有の設定フォルダおよびパラメータが含まれます。 デバイス設定は、デバイスのクラスタで実行されるグラフとは別に、クラスタ内の特定のデバイスの機能性に影響を及ぼします。
vnsMCredSecret	サービス デバイスにログインするためのパスワードが含まれます。
vnsMDevCfg	基本レベルのデバイス設定を表します。 このオブジェクトはアンカーとして、さまざまなデバイス設定と共有設定（`MGrpCfg`）を区別します。 `MDevCfg` での設定は、複数グラフ上の機能の複数インスタンスで共有することができます。

コンポーネント	説明
vnMGrpCfg	メタグループ設定を表します。 これにはグラフの複数の機能で共有できる設定の一部が含まれます。 グループ設定下での設定はグラフ インスタンス内にあり、他のグラフでは参照することができません。
vnsMFolder	メタ フォルダ情報を表します。 モデルは MFolders および MParams で構成される汎用設定を使用します。 このオブジェクトでは設定を階層として指定できます。
vnsMParam	設定を階層として指定することを可能にします。 このモデルでは、メタデータはキー、タイプ（整数、文字列）、およびパラメータに関連する他の属性で構成されます。
vnsMRel	別のオブジェクトとのメタ関係を表します。 また、別のフォルダまたはパラメータの参照が可能です。
vnsMFunc	デバイスの単一機能のメタデータが含まれます。 機能には、一連のコネクタと機能固有の設定ツリーが含まれます。 この管理対象オブジェクトには、このようなすべてのオペレーション用のメタデータが含まれます。
vnsMConn	ロジカル機能間のコネクタを表します。 メタデータには、任意の接続の濃度、方向、およびカプセル化タイプ（VXLAN または VLAN）などがあります。
vnsMIfLbl	インターフェイスのラベルを表します。 インターフェイスは、抽象的な方法によってデバイスでラベル付けできます。 たとえば、ファイア ウォール デバイスは、信頼できるインターフェイス、信頼できないインターフェイス、管理インターフェイスを実装できます。 コンクリート モデルは、デバイスでサポートされるラベルの数を指定します。
vnsMImage	機能のイメージ ファイルを表します。 この管理対象オブジェクトには、イメージ ファイルとその場所の詳細が含まれます。 グラフィカル ユーザ インターフェイス（GUI）のイメージを実行するのに使われます。
vnsMChainable	すぐに親機能に従えるデバイスの機能名を指定します。 この管理対象オブジェクトには連鎖できる機能名が含まれます。

コンポーネント	説明
vnsAbsFuncProfContr	機能プロファイル グループのコンテナです。 特定のアプリケーションの機能のプロファイル グループ（グラフ）の集合を定義します。 各機能のプロファイル グループは、特定のアプリケーションに対して特定のデフォルト パラメータが初期設定されている 1 つ以上の機能を含むことができます。 機能プロファイルのコンテナは、デバイス パッケージの開発者によるデバイス モデル内で定義することも、テナントで定義し、一連のアプリケーションにグラフのカタログを与えることも可能です。
vnsAbsFuncProfGrp	機能プロファイル グループを表します。 特定のアプリケーションに対してデフォルト パラメータが初期設定されている機能の集合です。 機能プロファイル グループは、デバイス パッケージの開発者によるデバイス パッケージ内で定義することも、特定のアプリケーションのグラフのカタログとしてAPICテナントによって定義することも可能です。
vnsAbsFuncProf	機能プロファイルを表します。 これには vnsAbsDevCfg（vnsMDevCfg のインスタンス）、vnsAbsGrpCfg（vnsMGrpCfg のインスタンス）、および vnsAbsFuncCfg（vnsMFunc のインスタンス）が含まれます。 機能プロファイルはデバイス モデルで定義された特定の機能にリンクされます。 機能プロファイルはデバイス パッケージ内で定義することも、vnsAbsFuncProfGrp 内の機能のカタログとして APIC テナントで 定義することも可能です。

管理対象オブジェクトの例

次の XML ファイルには、管理対象オブジェクトの設定サンプルが含まれます。 同様の XML ファイルを使って、APIC のネットワーク デバイスをインスタンス化できます。

```
<polUni>
    <infraInfra>

        <vnsMDev vendor="Insieme"
                 model="SampleDevice"
                 version="1.0">

            <vnsMIfLbl name="external"/>
            <vnsMIfLbl name="internal" />
            <vnsMIfLbl name="management" />

            <vnsMCred name="username"
                      key="username"/>
            <vnsMCredSecret name="password"
                      key="password"/>

            <vnsDevScript name="Insieme"
                          packageName="DeviceScript.py"
                          versionExpr="1.0"
                          ctrlrVersion="1.0"/>

            <vnsComparison name="isAlpha" cmp="match" value="[a-zA-Z]+" />
```

```
<vnsComposite name="isL4Port" comp="or">
    <vnsRange name="between1And64K" value1="0" value2="65535" />
</vnsComposite>

<vnsComposite name="isProtocol" comp="or">
    <vnsComparison name="ip" cmp="eq" value="IP" />
    <vnsComparison name="tcp" cmp="eq" value="TCP" />
    <vnsComparison name="udp" cmp="eq" value="UDP" />
    <vnsComparison name="http" cmp="eq" value="HTTP" />
</vnsComposite>

<vnsComposite name="yesNo" comp="or">
    <vnsComparison name="yes" cmp="eq" value="YES" />
    <vnsComparison name="No" cmp="eq" value="NO" />
</vnsComposite>

<vnsComposite name="enableDisable" comp="or">
    <vnsComparison name="enabled" cmp="match" value="ENABLED" />
    <vnsComparison name="disabled" cmp="match" value="DISABLED" />
</vnsComposite>

<vnsComposite name="onOff" comp="or">
    <vnsComparison name="on" cmp="match" value="ON" />
    <vnsComparison name="off" cmp="match" value="OFF" />
</vnsComposite>

<vnsComparison name="isIPAddress"
               cmp="match"

value="([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])"/>


<vnsComparison name="isIPMask"
               cmp="match"

value="([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])"/>


<vnsMDfcts>
     <vnsMDfct code="100"
               recAct="Configure a Netmask for the vipaddress"
               descr="VIP requires vipaddress and NetMask"/>
     <vnsMDfct code="200"
               recAct="Configure a relation to VIP Folder"
               descr="Description of defect 100"/>
</vnsMDfcts>

<vnsMDevCfg name="DeviceConfig">
    <vnsMFolder key="Network"
                scopedBy="epg">

        <vnsMFolder key="vip">
            <vnsMParam key="vipaddress"
                       description="VIP IPAddress"
                       dType="str"
                       validation="isIPAddress"/>
            <vnsMParam key="NetMask"
                       dType="str"
                       description="NetMask for the IP Address"
                       validation="isIPMask"/>
        </vnsMFolder>

        <vnsMFolder key="subnetip">
            <vnsMParam key="subnetipaddress"
                       description="Subnet IPAddress of the Device"
                       dType="str"
                       validation="isIPAddress"/>
            <vnsMParam key="NetMask"
                       description="NetMask for the IP Address"
                       dType="str"
                       validation="isIPMask"/>
        </vnsMFolder>
```

```
                    <vnsMFolder key="mgmtip">
                        <vnsMParam key="mgmtipaddress"
                                   description="Mgmtm IPAddress of the Device"
                                   dType="str"
                                   validation="isIPAddress"/>
                        <vnsMParam key="NetMask"
                                   dType="str"
                                   description="NetMask for the IP Address"
                                   validation="isIPMask"/>
                    </vnsMFolder>
                  </vnsMFolder>

                  <vnsMFolder key="Monitor"
                            scopedBy="epg">
                     <vnsMParam key="dup_state"
                                description="dup_state"/>
                     <vnsMParam key="dup_weight"
                                description="dup_state"/>
                     <vnsMParam key="state"
                                description="state"/>
                     <vnsMParam key="weight"
                                description="state"/>
                </vnsMFolder>

                  <vnsMFolder key="Server"
                              scopedBy="epg">

                      <vnsMParam key="name"
                                 description="Server Name"
                                 dType="str"
                                 validation="isAlpha"/>
                      <vnsMParam key="domain"
                                 description="Domain name of the server"/>
                      <vnsMParam key="ipaddress"
                                 description="Server IP address"
                                 dType="str"
                                 validation="isIPAddress"/>
                  </vnsMFolder>

                  <vnsMFolder key="Service"
                              scopedBy="epg" cardinality="n">

                      <vnsMParam key="servicename"
                                 description="Service Name"
                                 dType="str"
                                 validation="isAlpha"/>
                      <vnsMParam key="servicetype"
                                 description="Service Type"
                                 dType="str"
                                 validation="isProtocol"/>
                      <vnsMParam key="servername"
                                 description="Server Name"
                                 dType="str"
                                 validation="isAlpha"/>
                      <vnsMParam key="serveripaddress"
                                 description="Server IP Address"
                                 dType="str"
                                 validation="isIPAddress"/>
                      <vnsMParam key="serviceport"
                                 description="IP port number for service"
                                 validation="isL4Port"/>

                      <vnsMRel key="MonitorConfig"
                               cardinality="n">
                          <vnsRsTarget
tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mDevCfg/mFolder-Monitor"/>
                      </vnsMRel>

                  </vnsMFolder>

                  <vnsMGrpCfg name="SharedConfig">
                  </vnsMGrpCfg>
```

```
                    <vnsMFolder key="Monitor"
                                scopedBy="epg">
                        <vnsMParam key="dup_state"
                                   description="dup_state"/>
                        <vnsMParam key="dup_weight"
                                   description="dup_state"/>
                        <vnsMParam key="state"
                                   description="state"/>
                        <vnsMParam key="weight"
                                   description="state"/>
                    </vnsMFolder>

                </vnsMDevCfg>

                <vnsMGrpCfg name="SharedConfig">
                </vnsMGrpCfg>

                <vnsMFunc name="SLB">
                    <vnsMImage name="slb.png"/>

                    <vnsMConn name="external"
                              dir="input"
                              encType="vlan"
                              notifications="Endpoint">
                        <vnsRsInterface
tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mIfLbl-external" />
                    </vnsMConn>

                    <vnsMConn name="internal"
                              dir="output"
                              encType="vlan"
                              notifications="Endpoint">
                        <vnsRsInterface
tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mIfLbl-internal" />
                    </vnsMConn>

                    <vnsMFolder key="VServer"
                                scopedBy="epg">

                        <vnsMParam key="vservername"
                                   description="Name of VServer"
                                   mandatory="true"
                                   dType="str"
                                   validation="isAlpha"/>
                        <vnsMParam key="port"
                                   description="Port for Virtual server"
                                   validation="isL4Port"/>
                        <vnsMParam key="persistencetype"
                                   description="persistencetype"/>
                        <vnsMParam key="servicename"
                                   description="Service bound to this vServer"/>
                        <vnsMParam key="servicetype"
                                   description="Service bound to this vServer"
                                   dType="str"
                                   validation="isProtocol"/>
                        <vnsMParam key="clttimeout"
                                   description="Client timeout"/>

                        <vnsMFolder key="VServerGlobalConfig"
                                   description="This references VServer global Configuration">


                            <vnsMRel key="ServiceConfig">
                                <vnsRsTarget
tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mDevCfg/mFolder-Service"/>
                            </vnsMRel>

                            <vnsMRel key="ServerConfig">
                                <vnsRsTarget
tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mDevCfg/mFolder-Server"/>
                            </vnsMRel>
```

```
                        <vnsMRel key="VipConfig">
                            <vnsRsTarget
tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mDevCfg/mFolder-Network/mFolder-vip"/>
                        <vnsRsConnector
tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mFunc-SLB/mConn-external"/>
                        </vnsMRel>

                    </vnsMFolder>

                </vnsMFolder>

            </vnsMFunc>

            <vnsMFunc name="SSL">
                <vnsMChainable name="SLB"/>
                <vnsMImage name="ssl.png"/>

                <vnsMConn name="internal"
                          dir="input"
                          encType="vlan">
                    <vnsRsInterface
tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mIfLbl-external" />
                </vnsMConn>

                <vnsMConn name="external"
                          dir="output"
                          encType="vlan">
                    <vnsRsInterface
tDn="uni/infra/mDev-Insieme-SampleDevice-1.0/mIfLbl-internal" />
                </vnsMConn>

                <vnsMFolder key="Certificate"
                            scopedBy="epg">

                    <vnsMParam key="passplain"
                               description="passplain"/>
                    <vnsMParam key="servicename"
                               description="servicename"/>
                    <vnsMParam key="clientcertnotbefore"
                               description="clientcertnotbefore"/>
                    <vnsMParam key="serial"
                               description="serial"/>
                    <vnsMParam key="publickeysize"
                               description="publickeysize"/>
                    <vnsMParam key="subject"
                               description="subject"/>
                    <vnsMParam key="fipskey"
                               description="fipskey"/>
                    <vnsMParam key="data"
                               description="data"/>
                    <vnsMParam key="nodomaincheck"
                               description="nodomaincheck"/>
                    <vnsMParam key="priority"
                               description="priority"/>
                    <vnsMParam key="publickey"
                               description="publickey"/>
                    <vnsMParam key="version"
                               description="version"/>
                    <vnsMParam key="notificationperiod"
                               description="notificationperiod"/>
                    <vnsMParam key="issuer"
                               description="issuer"/>
                    <vnsMParam key="status"
                               description="status"/>
                    <vnsMParam key="clientcertnotafter"
                               description="clientcertnotafter"/>
                    <vnsMParam key="certkey"
                               description="certkey"/>
                    <vnsMParam key="passcrypt"
                               description="passcrypt"/>
                    <vnsMParam key="key"
                               description="key"/>
                    <vnsMParam key="password"
```

```
                                description="password"/>
                        <vnsMParam key="signaturealg"
                                description="signaturealg"/>
                        <vnsMParam key="expirymonitor"
                                description="expirymonitor"/>
                        <vnsMParam key="linkcertkeyname"
                                description="linkcertkeyname"/>
                        <vnsMParam key="inform"
                                description="inform"/>
                        <vnsMParam key="cert"
                                description="cert"/>
                        <vnsMParam key="daystoexpiration"
                                description="daystoexpiration"/>
                    </vnsMFolder>
                </vnsMFunc>
            </vnsMDev>
        </infraInfra>
    </polUni>
```

第 3 章

デバイス スクリプトの開発

デバイスのスクリプトについて

デバイスのスクリプトは、APIC サービス API へのコールをデバイス固有のコールに変換し、Application Policy Infrastructure Controller （APIC）とネットワーク サービス間のアダプタとして動作します。

図 *5*：デバイスのスクリプトのモデル

デバイスのスクリプトは、各デバイス タイプのコールの処理環境である、スクリプトラッパーのコンテキスト内で動作します。 スクリプトラッパーは、CPU、ファイルおよびソケット リソースの消費を制限するネームスペース内で動作します。

デバイス パッケージをアップロードするとスクリプトラッパーが作成され、デバイスのスクリプト ファイルからモジュールをインポートします。 このモジュールは、API で説明されている機能を表示します。

デバイスのスクリプトはステートレスで、冪等（2 度以上実行しても同じ結果を生む）でなければいけません。 /tmp ディレクトリの一時的ステートの生成を除いて、スクリプト内でのファイル I/O 操作は許可されません。 持続的ステートを保存するための /tmp ディレクトリ内に作成されたファイルは使用しないでください。 APIC はクラスタで実行できます。 デバイスのスクリプトは、どの APIC インスタンスカラデモ起動できます。 /tmp ディレクトリに保存されているデータは、2 種類の API コールで使用できるとはかぎりません。 スクリプトは、どのようなファイルにもそれ自体のステートを保存することができません。

APIC では、スクリプトを Python 2.7 用に実行する必要があります。 Python 2.7 で使用できる標準ライブラリの他に、スクリプト環境は Python 要求ライブラリ v1.2.3 を提供します。 デバイス パッケージが他のライブラリを要求する場合は、デバイス パッケージの開発者はそうしたライブラリをデバイスのスクリプトの `zip` ファイルにバンドルすることができます。

（注） スクリプトはスレッド セーフにします。つまり、どのインスタンスでも、異なるデバイスを設定するために複数のスレッドがスクリプトで同じ機能を呼び出すことができます。 スクリプトはスレッドを呼び出すコンテキストで実行され、その実行の一部として新しいスレッドを生成できません。

Cisco Application Centric Infrastructure のリポジトリには、GitHub のサンプル デバイス スクリプト コードがあります。

https://github.com/datacenter/ACI

GitHub では、シスコがサポートしているのはシスコが配信するソリューションのみです。シスコはサードパーティのソリューションをサポートしていません。

デバイスのスクリプト作成の注意事項

Application Policy Infrastructure Controller（APIC）とネットワーク サービス間でアダプタを確立するには、すべてのスクリプト API を実装する必要があります。 API はデバイス固有階層に対応する Python ディクショナリを受け取ります。 特定のデバイスで必要とされる内部形式に Python ディクショナリを変換する必要があります。 同様に、デバイスが値を返した場合、API は APIC が必要とする形式に戻り値を変換する必要があります。 例については、「デバイス仕様の開発」の項の仕様と、このセクションの最後にあるサンプル スクリプトを参照してください。

デバイスのスクリプト API

デバイスのスクリプト API は 4 つのカテゴリーに分類されます。

- デバイス
- クラスタ
- サービス
- エンドポイントおよびネットワーク イベント

APIC では、テナント内の 1 つまたは複数のデバイスのクラスタをユーザが登録する必要があります。 すべてのサービス機能はクラスタに適用されます。 クラスタには 1 つまたは複数のネットワーク サービス デバイスを含めることができます。 単一のデバイスでクラスタを定義することで、デバイスを冗長性のないスタンドアロン モードで実行することができます。クラスタ内にアクティブ/スタンドバイ ピアとして設定した 2 つのデバイスを登録することで、デバイスをアクティブ/スタンバイの HA モードで実行することができます。 同様に、クラスタ内にアクティブ ピアとして設定した複数のデバイスを登録することで、デバイスをアクティブ/アクティブ モード

で実行することができます。クラスタ内に登録されているデバイスは、アクティブ/アクティブまたはアクティブ/スタンバイの組み合わせがあると想定されます。 クラスタ内のデバイスのHA設定はAPIC経由でプッシュすることも、あるいはAPICでデバイスを登録する前にデバイス上で直接アウトオブバンドにすることも可能です。

デバイスでの設定は、3 つのカテゴリに分割されます。サービス機能に固有の設定、デバイスに固有の設定、またはクラスタに固有の設定です。 サービス設定はサービス API 経由でプッシュされます。デバイス設定はデバイス API 経由でプッシュされます。そしてクラスタ設定はクラスタ API 経由でプッシュされます。

デバイス API

以下の API は、APIC でクラスタ内に登録されている各デバイス用に呼び出されます。

```
def deviceValidate( device, version )
def deviceModify( device, interfaces, configuration)
def deviceAudit( device, interfaces, configuration )
def deviceHealth( device, interfaces, configuration )
def deviceCounters( device, interfaces, configuration )
```

これらの API に配信される設定ディクショナリには、APIC でのデバイス固有の設定が含まれます。

（注） APIC は、デバイス API 呼び出し中にはクラスタ レベルまたはサービス機能の設定情報を渡しません。

デバイスの API はどのデバイス固有の設定でも機能しているとみなされ、クラスタのレベル設定を参照したり、クラスタのレベル設定に影響したり、またサービス機能に影響することはありません。

通常、デバイス固有のリンク バンドル（LACP など）のような設定は、deviceModify()、deviceAudit() の呼び出しで実行することができます。

ディクショナリに渡された設定は、デバイス モデルの vnsDevCfg 下で定義されたフォルダおよびパラメータのインスタンスです。

クラスタ API

以下の API は、APIC で登録されている各デバイスのクラスタ用に呼び出されます。

```
def clusterModify( device, interfaces, configuration )
def clusterAudit( device, interfaces, configuration )
```

設定ディクショナリには、APIC でのクラスタ設定が含まれます。

（注） APICは、クラスタAPI呼び出し中にはデバイス設定またはサービス設定の情報を渡しません。

クラスタの API はどのクラスタ レベルの設定でも機能しているとみなされ、デバイス固有または機能固有の設定を参照したり、それに影響することはありません。

通常、NTP サーバ、syslog サーバなどの HA クラスタ レベルで実行される設定は、クラスタ API を介して設定されます。

ディクショナリに渡された設定は、デバイス モデルの vnsClusterCfg 下で定義されたフォルダおよびパラメータのインスタンスです。

サービス API

以下の API は、デバイスで実行されたサービス機能用に呼び出されます。

```
def serviceModify( device, configuration )
def serviceAudit( device, configuration )
def serviceHealth( device, configuration )
def serviceCounters( device, configuration )
```

設定ディクショナリには、vnsMDevCfg、vnsGrpCfg、または vnsMFunc 下で定義されるパラメータおよびフォルダのインスタンスが含まれます。

（注） デバイス モデル内の vnsMDevCfg 下で定義されるフォルダまたはパラメータのインスタンスは、サービス グラフの機能のインスタンスがこれらのパラメータを参照する場合のみ、サービス API の呼び出しで渡されます。 vnsMDevCfg 下で定義されるすべてのパラメータおよびフォルダのインスタンスがサービス機能に渡されるわけではありません。 APIC は、サービス API の呼び出しで参照される特定の機能のインスタンスで使用されるパラメータとフォルダのみ配信します。

エンドポイントとネットワーク イベント API

以下の API は、グラフに関連付けられたエンドポイントのグループ（EPG）用にエンドポイントまたはネットワークが変更された場合に呼び出されます。

```
def attachEndpoint( device, configuration, endpoints )
def detachEndpoint( device, configuration, endpoints )
def attachNetwork( device, configuration, networks)
def detachNetwork( device, configuration, networks )
```

これらの API は、デバイス仕様がエンドポイントまたはネットワーク接続通知をサポートし、機能コネクタの通知が可能な場合のみ呼び出されます。 AttachEndpoint および DetachEndpoint イベントは、EPG 内のエンドポイントが接続または切断されると呼び出されます。 ネットワーク API は、ブリッジ ドメインまたは EPG でサブネット設定が変更されると呼び出されます。 これらの API が提供する情報はサービス機能設定を自動化し、エンドポイントまたはネットワーク設定が変更されると、サービス機能設定も修正されます。例としては、ロードバランサにアタッチされるプールに対して動的にサーバを追加または削除すること、またはファイアウォールに対して定義されたアクセスリスト内のサブネットを動的にアップデートすることなどがあります。デバイス仕様ファイルは、APIC が要求するリターン形式で成功を返す空の機能を定義することができます。機能性を処理するエンドポイントまたはネットワークイベントのサポートは必須ではありません。

デバイス パッケージがこれらの機能をサポートしない場合、このドキュメントで説明するように、リターン ディクショナリ形式で成功のステータスを常に返すようなスタブ機能を定義します。

スクリプト フレームワーク

デバイスのスクリプトでインポートする必要がある 2 つのモジュールは次のとおりです。

- インポート Insieme.Logger
- インポート Insieme.Config

ロギング

Insieme.Logger はロギング ユーティリティを定義します。 デバイスのスクリプトはこのユーティリティを使ってデバッグ情報をログできます。 ロギング ユーティリティは `debug.log` というファイルに設定 API ログを記録します。 このファイルは Application Policy Infrastructure Controller（APIC）から収集されたテクニカル サポート データに含まれます。 デバイスのスクリプトの開発者は、スクリプトの問題のデバッグに役立つよう、できる限り多くの情報を記録する必要があります。

serviceHealth() や serviceCounters() などの定期的 API のログは、periodic.log にリダイレクトされます。 debug.log および periodic.log ファイルには、ファブリック管理者として /data/devicescript/<vendor-model-version>/logs の下の APIC でアクセスできます。

ロギング機能は、Python ロギング機能に類似しています。 ログは、次のカテゴリに分割できます。

- CRIT = 0
- ERROR = 1
- WARN = 2
- INFO = 3
- DEBUG = 4
- DEBUG2 = 5
- DEBUG3 = 6
- DEBUG4 = 7

スクリプトによって、次のように API を呼び出すことができます。

```
Logger.log( level,  Log String)
```

次に、API を呼び出す例を示します。

```
Logger.log( Logger.DEBUG, 'Connection to device failed')
```

定数

Insieme.Config は、ディクショナリの解析に使用できる定数を定義します。

```
Type = Insieme.Fwk.Enum(
    DEV=0,
    GRP=1,
    CONN=2,
    FUNC=3,
    FOLDER=4,
    PARAM=5,
```

```
    RELATION=6,
    ENCAP=7,
    ENCAPASS=8,
    ENCAPREL=9,
    VIF=10,
    CIF=11,
    LIF=12,
)

State = Insieme.Fwk.Enum(
    UNCHANGED=0,
    NEW=1,
    CHANGED=2,
    DELETED=3,
)

Result = Insieme.Fwk.Enum(
    SUCCESS=0,
    TRANSIENT=1,
    PERMANENT=2,
    AUDIT=3,
)
```

設定ディクショナリの形式

クラスタ API、デバイス API およびサービス API で渡される設定ディクショナリは、デバイス仕様ファイルで定義されているとおりの構造に従います。 設定は各レベルがフォルダを指定するディクショナリの階層として渡されます。 ディクショナリの形式は次のとおりです。

```
(type,  key, name) : { 'state': …
                       'transaction': …
                       'connector': …
                       'value': …
                       'target': …
                       'device': …
                     }
```

フィールドは次のとおりです。

フィールド	説明

type	ディクショナリによって表されるオブジェクトのタイプを特定します。 このフィールドは、次のいずれかの値を取ることができます。
	`DEV=0,`
	`GRP=1,`
	`CONN=2,`
	`FUNC=3,`
	`FOLDER=4,`
	`PARAM=5,`
	`RELATION=6,`
	`ENCAP=7,`
	`ENCAPASS=8,`
	`ENCAPREL=9,`
	`VIF=10,`
	`CIF=11,`
	`LIF=12`

key	オブジェクトのデバイス仕様ファイルで定義されているキーまたは名前属性を指定します。
name	ユーザが入力するパラメータまたはフォルダのインスタンス名を指定します。

フィールド	説明

state	オブジェクトの状態を特定します。 このフィールドは、次のいずれかの値を取ることができます。
	`UNCHANGED=0,`
	`NEW=1,`
	`CHANGED=2,`
	`DELETED=3,`

connector	仕様ファイルで定義された関係に従って解決される接続インスタンスの名前を指定します。 このフィールドは、対応する `vnsMFolder` オブジェクトまたは `vnsMRel` オブジェクトにデバイス仕様ファイルで定義される `vnsRsConnector` 関係がある場合にのみ、フォルダまたは関係ディクショナリに読み込まれます。
value	オブジェクトの値を定義します。 フォルダの場合は、このフィールドは別のディクショナリを含むことができます。 関係オブジェクトには値要素が含まれず、代わりにターゲット要素があります。 パラメータ オブジェクトの値は 512 文字を超えてはいけません。
target	関係オブジェクトが解決されるターゲットフォルダを定義します。 この要素は、関係オブジェクトに対してのみ書き込まれます。
transaction	特定の API コールアウトにつながるApplication Policy Infrastructure Controller（APIC）トランザクション ID が含まれます。 トランザクション ID は APIC とデバイス スクリプト間の要求/応答を相関させるために使用されます。 これは主に、デバッグの便宜性のために使用されます。 スクリプトはこの値を無視できます。

フィールド	説明
device	通常はクラスタ API またはサービス API で渡される設定がクラスタに適用され、クラスタ内のすべてのデバイスで有効になります。 ただし、クラスタ内のデバイスのインターフェイス IP を設定する場合など、その設定をクラスタ内の特定のデバイスに適用することが必要になる可能性があります。 各デバイスには一意の IP を割り当てることができます。 その結果、インターフェイス IP 設定をクラスタ内の 1 台の特定のデバイスに適用する必要があります。 これは、APIC のデバイス コンテキストのラベルを使用して行われます。 APIC でパラメータを設定すると、ユーザは（任意で）この設定が適用されるクラスタ内でデバイスを指定するデバイス コンテキストのラベルを関連付けることができます。 パラメータがクラスタ内の特定のデバイス コンテキストに結合されると、APIC は値としてデバイス名でディクショナリの「デバイス」キーをインスタンス化します。 スクリプトはコールアウトで渡されたデバイスディクショナリでデバイス名を検索できます。 スクリプトはデバイス フィールドで指定された特定のデバイスにパラメータ設定を適用することができます。

設定ディクショナリの詳細については、サンプル スクリプト, （61 ページ）を参照してください。 コネクタ、カプセル化およびインターフェイス情報のディクショナリ例については、ファブリック接続, （75 ページ） を参照してください。

API 戻り値

API は次の形式のディクショナリを返します。

```
{ 'state':
  'health': []
  'fault': []
}
```

状態は次の値の 1 つを戻します。

```
SUCCESS=0
TRANSIENT=1
PERMANENT=2
AUDIT=3
```

健全性の詳細については、ヘルス モニタリング, （85 ページ）を参照してください。 エラーの詳細については、エラー コード, （30 ページ）を参照してください。

デバイスとの接続を確立するために少なくとも30秒のタイムアウトをセットするようにデバイスのスクリプトを設定する必要があります。 デバイスのスクリプトが時間間隔内にネットワーク接続を確立できないと、リターン ディクショナリに TRANSIENT（1）状態を返します。 APIC はこのトランシエント状態がクリアされるまでトランザクションを再試行します。

トランシエント エラーは、ユーザがすぐに注意を向けて問題を解決する必要のないエラーを示します。 スクリプトが設定をプッシュするのを妨げる一時的イベントである可能性があります。 APIC はエラーがクリアされるまで 設定を積極的に再プッシュします。 トランシエント エラーが複数（5 回）の再試行後にクリアされなければ、APIC は失敗を持続的としてマークします。

リターン ディクショナリに AUDIT 状態を返すことで、デバイスのスクリプトが監査コールを要求できます。 APIC は AUDIT 状態を返したのがクラスタ API だったのか、デバイス API だったのか、またはサービス API だったのかによって、clusterAudit()、deviceAudit()、または serviceAudit() をトリガーします。 スクリプトは、現在の API コール内で解決できない APIC とデバイスの間の設定不一致を検出するイベントの監査を要求できます。

渡されたパラメータ値または設定に問題があり、問題解決にユーザの介入を必要とする場合、デバイスのスクリプトは PERMANENT エラーを返します。

繰り返し起こるトランシエント エラーは持続的エラーと解釈されることがあります。 APIC はエラーが解決するまで定期的に設定をプッシュし続けます。 再試行はより長い間隔で実行されます。 エラーによっては、クリアするためにユーザの介入が要求される場合もあります（無効なパラメータ値など）。

デバイスのスクリプト API を呼び出す**スクリプトラッパー** プロセスは、API が120 秒以内に返すことを予期します。 スクリプトが 120 秒以上かかる場合、スクリプトラッパー プロセスは終了し、再起動します。 未処理のトランザクションは再起動後に再生されます。

サービス設定

Application Policy Infrastructure Controller（APIC）は、各テナント コンテキストにメタデバイス（`MDev`）のインスタンスを作成します。 `MDev` インスタンスは仮想デバイス、または `vDev` と呼ばれます。 テナントのすべてのサービス設定インスタンスは、`vDev` の下に基づいています。 APIC は、各 `vDev` を指定するための一意的 ID を作成します。 APIC はまた、各グラフ インスタンスに一意的 ID を作成し、`vGrp` として表されます。 グループ設定と機能設定は、特定のグラフ インスタンスを指定するこの `vGrp` インスタンスの下に基づいています。

`serviceAudit()`、`serviceModify()`、`serviceHealth()`、`serviceCounter()` で渡される設定ディクショナリには、常に `vDev` オブジェクトと `vGrp` オブジェクトが含まれています。 マルチ コンテキストのデバイス スクリプトは `vDev` オブジェクトを使ってテナント コンテキストを一意に識別する必要があります。これにより特定のルーティング ドメインまたはコンテキストにマッピングできます。

デバイスでのパラメータ インスタンス名

`vnsMDevCfg` 下で定義されるフォルダとパラメータはすべて、`vDev` の下でインスタンス化されます。 マルチ コンテキストのデバイスで、各 `vDev` ID に対して設定フォルダを作成する必要があり

ます。または、デバイスまたはデバイス スクリプトで、設定ディクショナリに渡される vDev ID を連結し、複数のコンテキスト全体で一意の名前を作成する必要があります。

次に、機能用にグローバル フォルダとパラメータがあるディクショナリの例を示します。

```
{
  (0,'',4304): {
     'state': 1,
     'transaction': 10000,
     'value': {
          (4,'Server','webserver1'): {
             'state': 1,
              'transaction': 10000,
              'value': {
              (5,'ipaddress','ipaddress'): {
                 'state': 1,
                 'transaction': 10000,
                 'value': '192.168.100.2'
               },
               (5,'servername','servername'): {
                 'state': 1,
                 'transaction': 10000,
                 'value': 'webserver1'
               }
          }
      }
}
```

デバイスのスクリプトは、サーバ名のインスタンスが異なるコンテキストで一意に識別されることを確認する必要があります。 名前は同じデバイスで設定された異なるコンテキストで重複することがあるので、デバイスのスクリプトはサーバ名の値に vDev ID を追加し、パラメータとフォルダ名を異なるコンテキストで一意にすることができます。 次に、一意のサーバ名の値の例を示します。

```
4304_webserver1
webserver1_4304
webserver1.4304
```

一意のサーバ名の値を作成するもう 1 つの方法は、4304 というフォルダを作り、その 4303 フォルダの下に webserver1 インスタンスを作成することです。

単一コンテキストのデバイスは設定ディクショナリに渡された vDev 引数を無視できます。 同様に、単一コンテキストまたは複数コンテキストのデバイスは、あるグラフのインスタンスが同じデバイスで実行される他のグラフのインスタンスから一意であるように設定されたパラメータとフォルダの名前を保持するために、グループ ID を追加する必要があります。次の例でそれを示します。

```
(0,'',4304): {
 'state': 1,
 'transaction': 10000,
 'value': {
    (1,'',4368): {
       'state': 1,
       'transaction': 10000,
       'value': {
          (3,'SLB','SLB'): {
             'state': 1,
              'transaction': 10000,
              'value': {
                  (5,'servicename','servicename'): {
                     'state': 1,
                     'transaction': 10000,
                     'value': 'webservice'
                  },
              }
          }
      }
```

```
    }
  }
}
```

デバイスのスクリプトは、このグラフのインスタンスのために作られたサービス名のインスタンスが、同じデバイスで実行される他のグラフのインスタンスと重複しないことを確認する必要があります。 理由は、デバイス仕様におけるグループ設定または機能設定の下で定義されるパラメータおよびフォルダは、グラフ内のグラフ インスタンスまたは機能インスタンスにおいて一意であるためです。 このようなパラメータまたはフォルダの名前は、グラフの異なるインスタンスにおいて、またはグラフ内の機能のインスタンスにおいて、それぞれ一意でない場合があります。デバイスのスクリプトはグループ ID またはデバイスのディクショナリに渡されるグループ ID と機能名を追加して、グラフまたはグラフ内の機能においてフォルダとパラメータの名前を一意にすることができます。次の例でそれを示します。

```
4368.SLB.webservice
```

デバイスがフォルダの作成をサポートしている場合、スクリプトは設定ディクショナリに渡された vgrp ID で指定されるグラフのフォルダを作成し、グループ特定のパラメータを vgrp フォルダの下でインスタンス化することができます。 同様に、機能設定をグループ フォルダ内の機能固有フォルダの下に作成し、複数のグラフのインスタンスにおける各パラメータ/フォルダの一意性を保持することも可能です。

API 呼び出し

クラスタの設定

Application Policy Infrastructure Controller（APIC）にクラスタ内の最初のコンクリート デバイスを登録すると、次のような API 呼び出しのシーケンスがトリガーされます。

1. `deviceValidate`：この API は、APIC に登録されたデバイスのバージョンがデバイス パッケージにサポートされるかどうかを検証します。
2. `deviceAudit`：この APIC コールは、APIC からプッシュされていないデバイスのグローバル設定をすべてクリアします。 デバイスのスクリプトは、APIC でサポートされていなくても、デバイスが作動するのに必要な管理 IP アドレス、ログイン クレデンシャル、および他の設定をクリアしません。 `deviceAudit` コールは設定をクリアする上で選択的である必要があります。APIC からプッシュできる設定のみクリアされます。
   deviceAudit() は、サービス機能のパラメータ/設定、またはクラスタ レベル設定は変更しません。 サービス機能は deviceAudit() コールでは影響を受けません。 deviceAudit() コールの目的は、デバイスのレベル設定を APIC と同期させることです。 スクリプトはデータ パスの中断が最小限になるようデバイスを同期します。 スクリプトはデバイスにある APIC でプッシュされなかった設定を識別し、そのような設定は削除されます。

（注） これはデバイス パッケージによって管理できる設定の場合だけ行われます。

   スクリプトは欠落した設定、またはデバイスと同期していない設定をプッシュします。
3. `clusterAudit`：最初のデバイスがロジカル デバイス（デバイス クラスタ）に追加されると、APIC はこの API を呼び出します。 この API は、APIC によってプッシュされないクラスタか

らの設定をすべてクリアします。 deviceAudit() コールと同様に、この API は、APIC でサポートできる設定のみクリアします。
clusterAudit() は、サービス機能のパラメータ/設定またはデバイス固有のパラメータは変更しません。 サービス機能は clusterAudit() コールでは影響を受けません。 clusterAudit() コールの目的は、クラスタのレベル設定を APIC と同期させることです。 スクリプトはデータ パスの中断が最小限になるようデバイスを同期します。 スクリプトはデバイスにある APIC でプッシュされなかった設定を識別し、そのような設定は削除されます。

（注） これはデバイス パッケージによって管理できる設定の場合だけ行われます。

スクリプトは欠落した設定、またはデバイスと同期していない設定をプッシュします。

4. clusterModify：この API は、ロジカル デバイス（デバイス クラスタ）に登録され、追加されたデバイスで呼び出されます。 そのコールによりクラスタ設定が行われます。
5. serviceAudit：この API は、ユーザが APIC で適用した機能設定で呼び出されます。 スクリプトは、設定で渡されたサービス機能をプッシュします。 デバイスに APIC で設定されていないサービス機能があり、それを API で管理できる場合、スクリプトはその設定を削除します。 serviceAudit() コールの目的は、デバイスにおけるサービス固有の機能が APIC と同期しているのを確認することです。

（注） APIC 内で保持されているデバイスのクラスタ状態は、clusterAudit() が成功を返すと動作するように変更されます。 すべてのサービス レベルの API は、クラスタが作動した後にのみ呼び出されます。

いったんクラスタ内の 1 つのデバイスが動作すると、そのクラスタは作動中とマークされます。

クラスタ内に追加デバイスを登録すると、次のコールがトリガーされます。

- deviceValidate()
- deviceAudit()
- clusterAudit()

（注） clusterAudit() がクラスタ内で作動するデバイスを中断することはありません。 クラスタに展開されるサービス機能以外に、デバイスの追加（deviceAudit() または clusterAudit() など）が他に影響を与えることはなく、クラスタで作動状態にあるサービス機能も影響されません。

デバイスの登録後、APIC は定期的に次の API を呼び出します。

- deviceHealth
- deviceCounter

クラスタ内のデバイスを削除すると、clusterAudit() が呼び出されます。 クラスタから最後のデバイスが削除され、作動を停止するようクラスタ状態が変更されると、APICはサービス固有の設定を削除するよう serviceModify() を呼び出します。

サービス グラフの設定

抽象グラフを、エンドポイントのグループ（EPG）にバインドされるコントラクトに関連付けることで、グラフがインスタンス化されると、APIC は次の API を呼び出します。

- `serviceModify`：この API は、デバイスのサービス機能をインスタンス化します。

サービスが終了すると、APIC は定期的に次の API を呼び出します。

- serviceHealth
- serviceCounter

監査コール

デバイスで未解決のトランザクションがあるときに APIC クラスタが変化すると、APIC は serviceAudit() をトリガーします。 クラスタの変化によって異なる APIC がデバイスとの通信を再開すると、以前の設定トランザクションが完了せず、デバイスと APIC クラスタが同期しなくなることがあります。 新しいマスターによって発行されたserviceAudit() コールを呼び出せば、デバイスの状態は APIC との同期が保持されます。

APIC は1 回の serviceAudit() コールで、任意のデバイスに関連付けられたサービス設定全体を渡します。 スクリプトは、デバイスに設定されたサービスの中断を最小限に抑えた状態で、serviceAudit() を処理するよう要求されます。 serviceAudit() コールで、デバイス固有のグローバル設定やクラスタ設定がクリアされることはありません。

serviceAudit() と同様に、APICは、未解決のデバイス設定またはクラスタ設定トランザクションが進行中かどうかによって、deviceAudit() および clusterAudit() をトリガーします。

デバイスと APIC の設定が変化し、スクリプトが API コール内の違いを解決できないことが検出されると、デバイス スクリプトは監査コールをトリガーすることができます。

`clusterAudit`() コールは、`clusterModify`() コールのリターン状態として「3」（AUDIT）を返すことでトリガーできます。 APIC は、`clusterAudit`() の `vnsClusterCfg` 下のフォルダおよびパラメータで定義されたすべてのクラスタ設定を渡します。 サービス機能の設定は `clusterAudit`() で渡されません。 スクリプトはディクショナリで渡された設定を適用し、APIC によって定義されていない設定を削除する必要があります。 スクリプトは、APIC で管理可能なデバイスで見つかり、`vnsClusterCfg` の下のデバイス仕様ファイルで定義される不要な設定のみ削除します。

`deviceAudit`() コールは、`deviceModify`()、`deviceHealth`() または `deviceCounter`() コールのリターン状態として、「3」（AUDIT）を返すことでトリガーできます。 APICは、`deviceAudit`() コールの `vnsDevCfgg` 下のフォルダおよびパラメータで定義されたデバイス設定全体を渡します。 サービス機能の設定は `deviceAudit`() で渡されません。 スクリプトはディクショナリで渡された設定を適用し、APICによって定義されていないデバイス設定を削除する必要があります。 スクリプトは、APIC で定義され、APIC を介して管理できる設定だけを削除します。

serviceAudit() は、serviceModify()、serviceHealth() または serviceCounter() コールのリターン状態として、「3」を返すことでトリガーできます。 APICは、デバイス クラスタで実行されるすべての vDev （テナント）とグラフのインスタンスにおいてすべてのサービス機能の設定を渡します。 デバイスは、APIC によって設定されず、APIC を介して管理できる設定を削除します。

パラメータの配信

次に、サービス API 用に渡される設定ディクショナリの例を示します。

```
Configuration = {
   (0, mDev-key, mDev-Instance-Name) : {
       'state': state
       'value': {
            (1, '', functionGroup-Instance) : {
               'state': state
               'value': {
                   (3, mDevFunction-Key, mDev-Function-Instance-Name): {
                       'state': state,
                       'value': {
                            (2, mDevFunction-Connector-Key, InstanceName): {
                               'state': state
                               'value': {
                                  'CDev-Instance-Name': 'EncapAssociation-Instance',
                                   ...
                               }
                            }
                            (4, mFolder-Key, mFolder-Instance): {
                                  'state': state
                                  'value': {
                                       (5, mParam-Key, mParam-Instance): {
                                            'state': state
                                             'device': cluster-node-instance
                                              'value': {
                                             }
                                       }
                                  }
                            }
                       }
                   }
               }
            }
            (4, mFolder-Key, mFolder-Instance): {
                  'state': state
                  'value': {
                       (5, mParam-Key, mParam-Instance): {
                            'state': state
                             'value': {
                             }
                       }
                  }
            },
            (7, '', <encap-instance>): {'state': 1, 'tag': TagValue, 'type': TagType },
            (8, '', <encap-association-Instance>): {
                 'state': 1,
                 'encap': <encapInstance>,
                 'vif': <LogicalInterfaceInstanceID>
            },
            (10, '', <LogicalInterfaceInstance>): {
                 'state': 0,
                 'cifs': {'
                    'state': 0,
                    'value': <Interface Value>
                 }
            }
       }
  },
```

```
}         `

Device = {
   'devs': {
      cluster-node-Instance: {
            'host': cluster-node-ipaddress
            'port': cluster-node-port-number,
            'creds': {
               'username': username,
               'password': password
            }
         }
    },
    'name': cluster-name,
    'host': cluster-ipaddress
    'port': cluster-port-number,
    'creds': {
        'username': username,
        'password': password
    }
}
```

（注） すべてのパラメータは文字列です。

デバイスの識別

デバイス パラメータは、デバイスへのアクセスに必要なデバイス設定とクレデンシャルを含むシンプルなディクショナリです。 次の例に示すように、デバイス スクリプトのほとんどの機能は、変更の対象となるデバイスを識別するデバイス パラメータを使用しています。

```
{
  'creds': {
    'password': 'admin',
    'username': 'admin'
  },
  'host': '10.30.13.153',
  'port': 443
}
```

APIC は暗号化されたパーティションにクレデンシャルを保存します。

スクリプトは一時ファイルにクレデンシャルを保存しません。 また、デバッグ ログにクレデンシャルを出力しません。

スクリプト エラーの処理

APICは、約束理論（Promise Theory）に基づく統合モデルを提供し、それによって個々のエージェントが自発的な連携システムに参加します。 APICは目標とする状態をスクリプトにプッシュし、設定の要素に発生したエラーを提起するための API を提供します。

エラーはパラメータ値が変更されたり、デバイス エラーが起こった場合に発生することがあります。 APIC のエラー時には、新たな APIC が `serviceModify` コールを再発行するか、または機能グループを監査するかを決定します。 API はエラー リストを返し、エラーの原因とエラーを解決する可能性のある対処についての詳細情報を提供することができます。

APICはトランザクション履歴を保持しません。 設定ディクショナリ内のオブジェクトの状態は、管理オブジェクト情報ツリーの状態に基づいて決定されます。 オブジェクトの状態は、新しいオ

ブジェクトが管理オブジェクト ツリーに挿入されると「NEW（1）」とマークされます。 その後の APIC からの API コールアウトは、オブジェクト値が、明示的なユーザ設定、またはオブジェクト値を変更するような暗黙の APIC イベントによって変更されるまで、そのオブジェクト状態を「UNCHANGED（0）」に設定します。

オブジェクト値が変わると、次の modify() API コールでのオブジェクトの状態が「CHANGED（2）」に設定されます。 オブジェクトが削除されると、APICはオブジェクト状態を「DELETED（3）」に設定して、削除を示します。

APIが、設定処理中にスクリプトがエラーを検出したことを示す状態「PERMANENT（2）」を戻す場合、APICはスクリプトから「SUCCESS（0）」を受信するまで設定を再試行します。 再試行の間にユーザがどのオブジェクト値も変えていない場合は、APICは再試行中にオブジェクト状態を「UNCHANGED（0）」に設定します。 オブジェクト状態が「NEW（1）」、「CHANGED（2）」、または「DELETED（3）」になるのは、新しいオブジェクトが作成されたか、既存のオブジェクトが変更されたか、または再試行中に削除された場合のみです。

次に、APIC のトランザクション履歴の欠如が原因で起こり得るケースを例として示します。 デバイス パッケージの開発者は、そうした問題を認識し、デバイスのスクリプトで対応する必要があります。

デバイス パッケージに A、B、C の 3 つのパラメータがあると仮定します。A は B に依存し、B は C に依存します。

- A、B、C が APIC で作成されると、APIC は、3 つのすべてのパラメータ A =a(1) -> B=b (1) -> C=c (1) に対して、状態 create (1) のスクリプトをデバイスに呼び出します。

  スクリプトは B にエラーを提起し、C を設定しますが、A は設定せず、状態 PERMANENT を返します。 このコールアウト デバイスの終りには「c」だけがあります。

  オブジェクト B の設定は無効だったため、APIC は B にエラーを提起します。

- B の値が b' に変更されると、これにより設定の問題が解決されます。 この設定の変更の結果、APIC はデバイスのスクリプトに修正を求めます。 それに続く API コールアウトの間のパラメータ状態は次のとおりです。

  A=a(0) -> B=b'(2) -> C=c(0)

  APICはトランザクション履歴を保持しません。 APICは、前回のトランザクション中にA=aがデバイスに適用されなかったことを認識していません。 APIC は設定を再試行し、再試行の間に変更したパラメータが B だけなので、B のパラメータ状態を更新します。

  スクリプトはパラメータ B を b' に更新しようとします。 スクリプトはデバイスにパラメータBが存在せず、作成前に変更されることを識別できる必要があります。 理想的には、デバイスへの修正要求はエラーを返します。

  デバイスが作成前の修正をエラーとして識別できる場合、デバイスからのそうしたエラーに対し、スクリプトは以下の 2 つのオプションのうちの 1 つを使って対処します。

  - オプション 1

    デバイスにおいてパラメータ B を値 b' で作成し、デバイスから失われた可能性のある従属オブジェクトを解決します。 スクリプトはディクショナリで渡される設定に従い、修正されたオブジェクトに依存するオブジェクトがデバイスで作成されたかどうかをチェックする必要があります。 オブジェクトが作成されていないと、スクリプトはオブ

ジェクトを作成します。 この例では、スクリプトはオブジェクト A=a がデバイスに見つからないため、B=b' に依存する A=a を作成します。

- オプション 2

  リターン応答で「AUDIT （3）」状態を返します。 APIC は設定全体を再生します。

（注） APIC が設定全体を渡すので、監査コールの要求はコストのかかる操作となる可能性があります。

デバイスのスクリプトは欠如している設定を識別し、あらゆる欠如設定を適用する必要があります。 このオプションは、ディクショナリ内の修正オブジェクトに設定での従属オブジェクトが存在する場合にのみ使用してください。

作成前にパラメータが修正されたときにデバイスがエラーを返せない場合、デバイスのスクリプトは、修正操作を実行する前に修正されるオブジェクトを読み取る必要があります。 修正前にデバイスから設定を読み取るというこのアプローチはコストがかかり、パフォーマンスに影響する可能性があります。 ソリューションを最適に維持するために、修正するオブジェクトに依存関係のある他のオブジェクトが存在する場合のみ、デバイスからの設定読み取りを実行してください。

サンプル スクリプト

次に、Python でのデバイス スクリプトの例を示します。

```
import pprint
import sys
import Insieme.Logger as Logger

#
# Infra API
#def deviceValidate( device,version ):
    return {
        'state': 0,'version': '1.0'
        }

def deviceModify( device,interfaces,configuration):
    return {
        'state': 0,'faults': [],'health': []
        }

def deviceAudit( device,interfaces,configuration ):
    return {
        'state': 0,'faults': [],'health': []
        }

def deviceHealth( device,interfaces,configuration ):
    return {
        'state': 0,'faults': [],'health': ([], 100)
        }

def deviceCounters( device,interfaces,configuration ):
    return {
        'state': 0,'counters': [
            ( [(11,'','eth0')],{
                'rxpackets':100,
```

```
                'rxerrors':101,
                'rxdrops':102,
                'txpackets':200,
                'txerrors':201,
                'txdrops':202
            }
          )
        ]
    }

def clusterModify( device,interfaces,configuration ):
    return {
        'state': 0,'faults': [],'health': []
        }

def clusterAudit( device,interfaces,configuration ):
    return {
        'state': 0,'faults': [],'health': []
        }

#
# FunctionGroup API
#

def serviceModify( device,configuration ):
    return {
        'state': 0,'faults': [],'health': []
       }

def serviceAudit( device,configuration ):
    return {
        'state': 0,'faults': [],'health': []
        }

def serviceHealth( device,configuration ):
    return {
        'state': 0,'faults': [],'health': []
        }

def serviceCounters( device,configuration ):
    externalIntferface, = [(0, 'Firewall', 4384), (1, '', 4432), (3, 'Firewall-Func',
'FW-1'),
        (2, 'external', 'external1') ]
    internalInterface = [(0, 'Firewall', 4384) (1, '', 4432) (3, 'Firewall-Func', 'FW-1'),
        (2, 'internal','internal1') ]
    Firewall-1-External-Counters = (externalInterface,
        { 'rxpackets': 100,
            'rxerrors': 0,
            'rxdrops': 0
            'txpackets': 100
            'txerrors': 4
            'txdrops': 2} )
    Firewall-1-Internal-Counters = (internalInterface,
        { 'rxpackets': 100,
            'rxerrors': 0,
            'rxdrops': 0
            'txpackets': 100
            'txerrors': 4
            'txdrops': 2} )
    Counters = [ Firewall-1-External-Counters, Firewall-1-Internal-Counters ]
        return {
            'state': 0,
            'counters': Counters
        }

#
# EndPoint/Network API
#

def attachEndpoint( device,
                    configuration,
                    endpoints ):
    return {
```

```
        'state': 0,
        'faults': [],
        'health': [],
        }

def detachEndpoint( device,
                    configuration,
                    endpoints ):
    return {
        'state': 0,
        'faults': [],
        'health': [],
        }

def attachNetwork( device,
                   configuration,
                   networks ):
    return {
        'state': 0,
        'faults': [],
        'health': [],
        }

def detachNetwork( device,
                   configuration,
                   networks ):
    return {
        'state': 0,
        'faults': [],
        'health': [],
        }
```

次に示すのは呼び出し例です。

```
Function: deviceValidate

Arguments:

(
  {
    'creds':
      {
        'password': 'admin','username': 'admin'
      },
      'host': '10.30.13.153','port': 443
  }
  u'1.0'
)

Function: deviceAudit

Arguments:

(
  {
    'host': '10.30.13.153','port': 443,'creds':
      {
        'username': 'admin','password': 'admin'
      }
  },
  {
    (11,'','1_1'): {
      'state': 0,'label': 'int'
    },(11,'','1_2'): {
      'state': 0,'label': 'ext'
    },(11,'','1_3'): {
      'state': 0,'label': 'mgmt'
    }
  },
  {
    (4,'HighAvailabilityCfg','HA'): {
      'state': 2,'value': {
        (5,'peerIP','peerip'): {
```

```
          'state': 2,'value': '10.30.13.154'
        }
      }
    }
  }
)

Function: clusterAudit

Arguments:

(
  {
    'name': 'Cluster1',
    'virtual': False,
    'devs': {
      'SampleDevice1': {
        'host': '10.30.13.153',
        'port': 443,
        'creds': {
          'username': 'admin',
          'password': 'admin'
        }
      }
    },
    'host': '10.30.13.153',
    'port': 443,
    'creds': {
      'username': 'admin',
      'password': 'admin'
    }
  },
  {
    (12, '', 'internal'): {
      'state': 0,
      'label': 'int'
    },
    (12, '', 'external'): {
      'state': 0,
      'label': 'ext'
    }
  },
  {
    (4,'SyslogConfig','syslogconfig'): {
      'state': 2,
      'value': {
        (5,'ipaddress','syslogip'): {
          'state': 2,
          'value': '10.168.62.100'
        }
      }
    },
    (4,'NTPConfig','ntpconfig'): {
      'state': 2,
      'value': {
        (5,'ipaddress','syslogip'): {
          'state': 2,
          'value': '10.168.62.1'
        }
      }
    }
  }
)

Function: clusterModify

Arguments:

(
  {
    'name': 'Cluster1',
    'virtual': False,
    'devs': {
```

```
      'SampleDevice1': {
        'host': '10.30.13.153',
        'port': 443,
        'creds': {
          'username': 'admin',
          'password': 'admin'
        }
      }
    },
    'host': '10.30.13.153',
    'port': 443,
    'creds': {
      'username': 'admin',
      'password': 'admin'
    }
  },
  {
    (12, '', 'internal'): {
      'state': 0,
      'label': 'int'
    },
    (12, '', 'external'): {
      'state': 0,
      'label': 'ext'
    }
  },
  {
    (4,'SyslogConfig','syslogconfig'): {
      'state': 2,
      'value': {
        (5,'ipaddress','syslogip'): {
          'state': 2,
          'value': '10.168.62.100'
        }
      }
    },
    (4,'NTPConfig','ntpconfig'): {
      'state': 2,
      'value': {
        (5,'ipaddress','syslogip'): {
          'state': 2,
          'value': '10.168.62.1'
        }
      }
    }
  }
)
```

Function: serviceModify

Arguments:

```
{
  'creds': {
    'password': 'admin',
    'username': 'admin'
  },
  'devs': {
    'SampleDevice1': {
      'creds': {
        'password': 'admin',
        'username': 'admin'
      },
      'host': '10.30.13.153',
      'port': 443
    }
  },
  'host': '10.30.13.153',
  'name': 'Cluster1',
  'port': 443,
  'virtual': False
}
{
```

```
(0,'',4304): {
  'state': 1,
  'transaction': 10000,
  'value': {
    (1,'',4368): {
      'state': 1,
      'transaction': 10000,
      'value': {
        (3,'SLB','SLB'): {
          'state': 1,
          'transaction': 10000,
          'value': {
            (2,'external','external'): {
              'state': 1,
              'transaction': 10000,
              'value': {
                (9,'','Cluster1_external_40962'): {
                  'state': 2,
                  'target': 'Cluster1_external_40962',
                  'transaction': 10000
                }
              }
            },
            (2,'internal','internal'): {
              'state': 1,
              'transaction': 10000,
              'value': {
                (9,'','Cluster1_internal_57862'): {
                  'state': 2,
                  'target': 'Cluster1_internal_57862',
                  'transaction': 10000
                }
              }
            },
            (4,'VServer','webVServer'): {
              'state': 1,
              'transaction': 10000,
              'value': {
                (4,'VServerGlobalConfig','VServerGlobalConfig'): {
                  'state': 1,
                  'transaction': 10000,
                  'value': {
                    (6,'ServerConfig','serverConfig'): {
                      'state': 1,
                      'target': 'webserver1',
                      'transaction': 10000
                    },
                    (6,'ServiceConfig','serviceConfig'): {
                      'state': 1,
                      'target': 'webservice',
                      'transaction': 10000
                    },
                    (6,'VipConfig','vipConfig'): {
                      'state': 1,
                      'target': 'webnetwork/webvip',
                      'transaction': 10000
                    }
                  }
                },
                (5,'port','port'): {
                  'state': 1,
                  'transaction': 10000,
                  'value': '80'
                },
                (5,'servicename','servicename'): {
                  'state': 1,
                  'transaction': 10000,
                  'value': 'webservice'
                },
                (5,'servicetype','servicetype'): {
                  'state': 1,
                  'transaction': 10000,
                  'value': 'tcp'
```

```
            },
            (5,'vservername','vservername'): {
              'state': 1,
              'transaction': 10000,
              'value': 'webvserver'
            }
          }
        }
      }
    }
  }
},
(4,'Network','webnetwork'): {
  'state': 1,
  'transaction': 10000,
  'value': {
    (4,'vip','webvip'): {
      'state': 1,
      'transaction': 10000,
      'value': {
        (5,'netmask','webnetmask'): {
          'state': 1,
          'transaction': 10000,
          'value': '255.255.255.255'
        },
        (5,'vipaddress','webvipaddress'): {
          'state': 1,
          'transaction': 10000,
          'value': '10.0.0.100'
        }
      }
    }
  }
},
(4,'Server','webserver1'): {
  'state': 1,
  'transaction': 10000,
  'value': {
    (5,'ipaddress','ipaddress'): {
      'state': 1,
      'transaction': 10000,
      'value': '192.168.100.2'
    },
    (5,'servername','servername'): {
      'state': 1,
      'transaction': 10000,
      'value': 'webserver1'
    }
  }
},
(4,'Service','webservice'): {
  'state': 1,
  'transaction': 10000,
  'value': {
    (5,'servicename','servicename'): {
      'state': 1,
      'transaction': 10000,
      'value': 'webservice'
    },
    (5,'serviceport','serviceport'): {
      'state': 1,
      'transaction': 10000,
      'value': '9080'},
    (5,'servicetype','servicetype'): {
      'state': 1,
      'transaction': 10000,
      'value': 'tcp'
    }
  }
},
(7,'','40962'): {
  'state': 1,
  'tag': 235,
```

```
        'transaction': 10000,
        'type': 1
      },
      (7,'','57862'): {
        'state': 1,
        'tag': 201,
        'transaction': 10000,
        'type': 1
      },
      (8,'','Cluster1_external_40962'): {
        'encap': '40962',
        'state': 1,
        'transaction': 10000,
        'vif': 'Cluster1_external'
      },
      (8,'','Cluster1_internal_57862'): {
        'encap': '57862',
        'state': 1,
        'transaction': 10000,
        'vif': 'Cluster1_internal'
      },
      (10,'','Cluster1_external'): {
        'cifs': {
          'SampleDevice1': '1_2'
        },
        'state': 1,
        'transaction': 10000
      },
      (10,'','Cluster1_internal'): {
        'cifs': {
          'SampleDevice1': '1_1'
        },
        'state': 1,'transaction': 10000
      }
    }
  }
}
```

Function: attachEndpoint

Arguments:

```
{
   'creds': {
     'password': 'admin',
     'username': 'admin'
   },
   'devs': {
      'SampleDevice1': {
         'creds': {
             'password': 'admin',
             'username': 'admin'
          },
         'host': '10.30.13.153',
         'port': 443
      }
   },
   'host': '10.30.13.153',
   'name': 'Cluster1',
   'port': 443,
   'virtual': False
}
{
(0,'',4304): {
  'state': 1,
  'transaction': 10000,
  'value': {
    (1,'',4368): {
      'state': 1,
      'transaction': 10000,
      'value': {
      (3,'SLB','SLB'): {
        'state': 1,
```

```
'transaction': 10000,
'value': {
   (2,'external','external'): {
     'state': 1,
     'transaction': 10000,
     'value': {
        (9,'','Cluster1_external_40962'): {
          'state': 2,
          'target': 'Cluster1_external_40962',
          'transaction': 10000
        }
     }
   },
   (2,'internal','internal'): {
     'state': 1,
     'transaction': 10000,
     'value': {
        (9,'','Cluster1_internal_57862'): {
          'state': 2,
          'target': 'Cluster1_internal_57862',
          'transaction': 10000
        }
     }
   },
   (4,'VServer','webVServer'): {
       'state': 1,
       'transaction': 10000,
       'value': {
           (4,'VServerGlobalConfig','VServerGlobalConfig'): {
              'state': 1,
              'transaction': 10000,
              'value': {
                  (6,'ServerConfig','serverConfig'): {
                     'state': 1,
                     'target': 'webserver1',
                     'transaction': 10000
                  },
                  (6,'ServiceConfig','serviceConfig'): {
                     'state': 1,
                     'target': 'webservice',
                     'transaction': 10000
                  },
                  (6,'VipConfig','vipConfig'): {
                     'state': 1,
                     'target': 'webnetwork/webvip',
                     'transaction': 10000
                  }
              }
           },
           (5,'port','port'): {
              'state': 1,
              'transaction': 10000,
              'value': '80'
           },
           (5,'servicename','servicename'): {
              'state': 1,
              'transaction': 10000,
              'value': 'webservice'
           },
           (5,'servicetype','servicetype'): {
              'state': 1,
              'transaction': 10000,
              'value': 'tcp'
           },
           (5,'vservername','vservername'): {
              'state': 1,
              'transaction': 10000,
              'value': 'webvserver'
           }
        }
     }
   }
 }
```

```
   }
},
(4,'Network','webnetwork'): {
   'state': 1,
   'transaction': 10000,
   'value': {
      (4,'vip','webvip'): {
         'state': 1,
         'transaction': 10000,
         'value': {
             (5,'netmask','webnetmask'): {
                'state': 1,
                'transaction': 10000,
                'value': '255.255.255.255'
             },
             (5,'vipaddress','webvipaddress'): {
                'state': 1,
                'transaction': 10000,
                'value': '10.0.0.100'
             }
         }
      }
   }
},
(4,'Server','webserver1'): {
    'state': 1,
    'transaction': 10000,
    'value': {
        (5,'ipaddress','ipaddress'): {
           'state': 1,
           'transaction': 10000,
           'value': '192.168.100.2'
        },
        (5,'servername','servername'): {
           'state': 1,
           'transaction': 10000,
           'value': 'webserver1'
        }
    }
},
(4,'Service','webservice'): {
    'state': 1,
    'transaction': 10000,
    'value': {
       (5,'servicename','servicename'): {
          'state': 1,
          'transaction': 10000,
          'value': 'webservice'
       },
       (5,'serviceport','serviceport'): {
          'state': 1,
          'transaction': 10000,
          'value': '9080'},
       (5,'servicetype','servicetype'): {
          'state': 1,
          'transaction': 10000,
          'value': 'tcp'
       }
    }
},
(7,'','40962'): {
   'state': 1,
   'tag': 235,
   'transaction': 10000,
   'type': 1
},
(7,'','57862'): {
    'state': 1,
    'tag': 201,
    'transaction': 10000,
    'type': 1
},
(8,'','Cluster1_external_40962'): {
```

```
            'encap': '40962',
            'state': 1,
            'transaction': 10000,
            'vif': 'Cluster1_external'
        },
        (8,'','Cluster1_internal_57862'): {
            'encap': '57862',
            'state': 1,
            'transaction': 10000,
            'vif': 'Cluster1_internal'
        },
        (10,'','Cluster1_external'): {
            'cifs': {
                'SampleDevice1': '1_2'
            },
            'state': 1,
            'transaction': 10000
        },
        (10,'','Cluster1_internal'): {
            'cifs': {
                'SampleDevice1': '1_1'
            },
            'state': 1,
            'transaction': 10000
        }
    }
}
}
{
   'addr': '10.0.0.3',
   'conn': 'external'
}
```

API 呼び出しのエンドポイント ディクショナリには、次の属性が含まれます。

- `'addr'`：EPG に接続するエンドポイントの IP アドレス。
- `'conn'`：他の機能ノードを通じて EPG が直接または間接的に接続されたコネクタ。

```
Function: attachEndpoint
Arguments:
{
   'creds': {
     'password': 'admin',
     'username': 'admin'
   },
   'devs': {
      'SampleDevice1': {
         'creds': {
             'password': 'admin',
             'username': 'admin'
          },
         'host': '10.30.13.153',
         'port': 443
      }
   },
   'host': '10.30.13.153',
   'name': 'Cluster1',
   'port': 443,
   'virtual': False
}
{
(0,'',4304): {
  'state': 1,
  'transaction': 10000,
  'value': {
    (1,'',4368): {
      'state': 1,
      'transaction': 10000,
      'value': {
      (3,'SLB','SLB'): {
```

```
'state': 1,
'transaction': 10000,
'value': {
   (2,'external','external'): {
     'state': 1,
     'transaction': 10000,
     'value': {
        (9,'','Cluster1_external_40962'): {
          'state': 2,
          'target': 'Cluster1_external_40962',
          'transaction': 10000
        }
     }
   },
   (2,'internal','internal'): {
     'state': 1,
     'transaction': 10000,
     'value': {
        (9,'','Cluster1_internal_57862'): {
          'state': 2,
          'target': 'Cluster1_internal_57862',
          'transaction': 10000
        }
     }
   },
   (4,'VServer','webVServer'): {
       'state': 1,
       'transaction': 10000,
       'value': {
           (4,'VServerGlobalConfig','VServerGlobalConfig'): {
              'state': 1,
              'transaction': 10000,
              'value': {
                  (6,'ServerConfig','serverConfig'): {
                     'state': 1,
                     'target': 'webserver1',
                     'transaction': 10000
                  },
                  (6,'ServiceConfig','serviceConfig'): {
                     'state': 1,
                     'target': 'webservice',
                     'transaction': 10000
                  },
                  (6,'VipConfig','vipConfig'): {
                     'state': 1,
                     'target': 'webnetwork/webvip',
                     'transaction': 10000
                  }
              }
           },
           (5,'port','port'): {
              'state': 1,
              'transaction': 10000,
              'value': '80'
           },
           (5,'servicename','servicename'): {
              'state': 1,
              'transaction': 10000,
              'value': 'webservice'
           },
           (5,'servicetype','servicetype'): {
              'state': 1,
              'transaction': 10000,
              'value': 'tcp'
           },
           (5,'vservername','vservername'): {
              'state': 1,
              'transaction': 10000,
              'value': 'webvserver'
           }
       }
     }
   }
```

```
      }
   }
},
(4,'Network','webnetwork'): {
   'state': 1,
   'transaction': 10000,
   'value': {
      (4,'vip','webvip'): {
         'state': 1,
         'transaction': 10000,
         'value': {
             (5,'netmask','webnetmask'): {
                'state': 1,
                'transaction': 10000,
                'value': '255.255.255.255'
             },
             (5,'vipaddress','webvipaddress'): {
                'state': 1,
                'transaction': 10000,
                'value': '10.0.0.100'
             }
         }
       }
   }
},
(4,'Server','webserver1'): {
    'state': 1,
    'transaction': 10000,
    'value': {
        (5,'ipaddress','ipaddress'): {
           'state': 1,
           'transaction': 10000,
           'value': '192.168.100.2'
        },
        (5,'servername','servername'): {
           'state': 1,
           'transaction': 10000,
           'value': 'webserver1'
        }
    }
},
(4,'Service','webservice'): {
    'state': 1,
    'transaction': 10000,
    'value': {
       (5,'servicename','servicename'): {
          'state': 1,
          'transaction': 10000,
          'value': 'webservice'
       },
       (5,'serviceport','serviceport'): {
          'state': 1,
          'transaction': 10000,
          'value': '9080'},
       (5,'servicetype','servicetype'): {
          'state': 1,
          'transaction': 10000,
          'value': 'tcp'
       }
    }
},
(7,'','40962'): {
   'state': 1,
   'tag': 235,
   'transaction': 10000,
   'type': 1
},
(7,'','57862'): {
    'state': 1,
    'tag': 201,
    'transaction': 10000,
    'type': 1
},
```

```
    (8,'','Cluster1_external_40962'): {
         'encap': '40962',
         'state': 1,
         'transaction': 10000,
         'vif': 'Cluster1_external'
    },
    (8,'','Cluster1_internal_57862'): {
         'encap': '57862',
         'state': 1,
         'transaction': 10000,
         'vif': 'Cluster1_internal'
    },
    (10,'','Cluster1_external'): {
        'cifs': {
            'SampleDevice1': '1_2'
         },
        'state': 1,
        'transaction': 10000
    },
    (10,'','Cluster1_internal'): {
        'cifs': {
            'SampleDevice1': '1_1'
         },
        'state': 1,
        'transaction': 10000
    }
  }
}
}
{
   'addr': '10.0.0.0/24',
   'conn': 'external'
}
```

API 呼び出しのエンドポイント ディクショナリには、次の属性が含まれます。

- `'addr'`：EPG に接続するエンドポイントの IP アドレス。
- `'conn'`：他の機能ノードを通じて EPG が直接または間接的に接続されたコネクタ。

第 4 章

ファブリック接続

ファブリック接続の概要

デバイスの登録

Application Policy Infrastructure Controller（APIC）によってサービス ノードを管理するには、管理者がサービスデバイスを明示的に登録する必要があります。 登録手順の実行時に、次の情報を入力する必要があります。

- トポロジ情報：ファブリックのリーフ ノードへのデバイスのインターフェイスの接続方法です。
- ラベルのインターフェイス：デバイス要件に基づきます。 APIC が使用するラベルは特定の機能のためにインターフェイスをコネクタにバインドし、それによってサービスデバイスが提供されます。
- IP アドレスとポート情報：デバイスに接続するために必要な情報です。
- ユーザ名とパスワード：デバイスを設定するために使用するクレデンシャルです。

次に示すファイアウォール デバイス仕様のサンプルは、インターフェイスの 3 種類のラベルを定義します。

- `Inside`：より信頼されるネットワーク インターフェイスを特定します（セキュア）。
- `Outside`：より信頼されないネットワーク インターフェイスを特定します。
- `Management`：管理接続に使うインターフェイスを特定します。

ラベルは vnsMLabel タグを使用して、デバイス仕様で定義されています。 ファイアウォール デバイス上のすべてのインターフェイスは、デバイス仕様で定義されたタイプのうちの 1 つに分類されます。

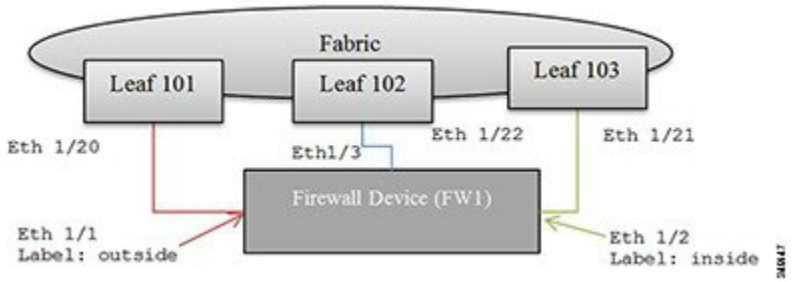
（注） APIC は、インターフェイスがデバイス上に実際に存在するかどうかチェックしません。

次の例は、APIC にデバイスを登録するためのノースバウンド XML ポストを示しています。 以下のようにリクエストをポストするか、APIC GUI を使用して、デバイスを登録できます。

```
<polUni>
    <fvTenant
        dn="uni/tn-Tenant1"
        name="Tenant1">
        <vnsLDevVip name="Firewall-1">

            <vnsLIf name="external">
                <vnsRsMetaIf tDn="uni/infra/mDev-CISCO-ASA-1.0.1.16/mIfLbl-external"/>
                <vnsRsCIfAtt tDn="uni/tn-Tenant1/lDevVip-Firewall/cDev-ASA/cIf-Eth1_1"/>
            </vnsLIf>
            <vnsLIf name="internal">
                <vnsRsMetaIf tDn="uni/infra/mDev-CISCO-ASA-1.0.1.16/mIfLbl-internal"/>
                <vnsRsCIfAtt tDn="uni/tn-Tenant1/lDevVip-Firewall/cDev-ASA/cIf-Eth1_2"/>
            </vnsLIf>
            <vnsCDev name="FW1">

                <vnsCIf name="Eth1_1">
                       <vnsRsCIfPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/20]"/>
                </vnsCIf>
                <vnsCIf name="Eth1_2">
                       <vnsRsCIfPathAtt tDn="topology/pod-1/paths-102/pathep-[eth1/21]"/>
                </vnsCIf>
                <vnsCIf name="Eth1_3">
                       <vnsRsCIfPathAtt tDn="topology/pod-1/paths-103/pathep-[eth1/22]"/>
                </vnsCIf>

                <vnsCMgmt name="devMgmt"
                          host="192.168.78.62"
                          port="80"
                          />

               <vnsCCred name="username"
                          value="admin"
                          />
               <vnsCCredSecret name="password"
                          value="insieme"
                          />
            </vnsCDev>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

次の図に、デバイス登録のトポロジを示します。

図 *6*：デバイス登録のトポロジ



デバイスを登録する 3 つのステップは、次のとおりです。

1 インターフェイスの登録：

- Eth 1/1：`outside` としてラベル付けされます。 リーフ ノード 101、Eth 1/20 に接続されます。
- Eth 1/2：`inside` としてラベル付けされます。 リーフ ノード 102、Eth 1/21に接続されます。
- Eth 1/3：`management` としてラベル付けされます。 リーフ ノード 103、Eth 1/22に接続されます。

2 デバイスに到達するための管理 IP アドレス（192.168.78.62）とポート（80）を指定します。

3 デバイスと通信するためのユーザ名とパスワードのクレデンシャルを指定します。

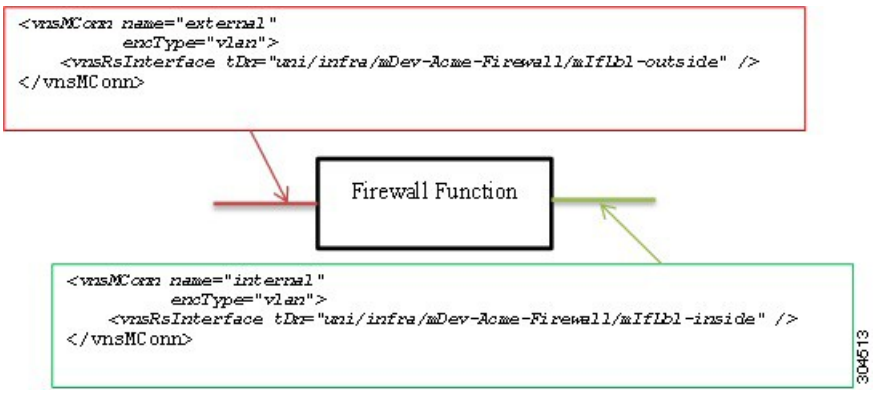
コネクタ

`vnsMFunc` 用コネクタは、2 つ以上の機能ノード間、またはグラフ内の機能ノードとファブリック間の接続を定義します。 コネクタは次の属性を持ちます。

- `name`：特定のコネクタを識別します。
- `encType`：パケットが VLAN ヘッダでタグ付けされているか、または VXLAN カプセル化されているかを定義します。
- `vnsRsInterface`：コネクタがファブリックへの接続を提供すると、このインターフェイスがコネクタをデバイスのインターフェイス タイプに関連付けます。

次の図では、2 台のコネクタがファイアウォール機能に関連付けられています。 最初のコネクタは外部またはアウトサイド ネットワークへの接続を表し、2 番目のコネクタは内部またはインサイド ネットワークへの接続を表します。 どちらも VLAN ヘッダでタグ付けされています。

図 7: ファイアウォール機能に関連付けられたコネクタ



サービス グラフ

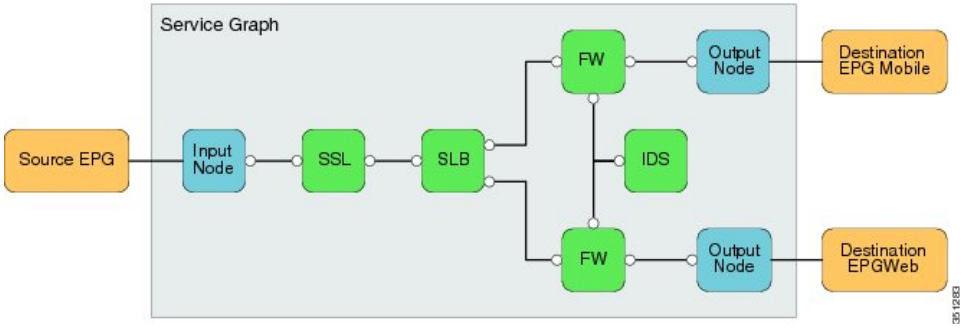
サービス グラフは、一連のターミナル間の順序付けられた機能セットです。 サービス グラフは GUI または CLI を使って手動で作成でき、また Application Policy Infrastructure Controller（APIC）のノースバウンド サービス統合 API を使ってプログラミング形式でも作成できます。 グラフ内の 1 つの機能は 1 つまたは複数のパラメータを必要とし、1 つまたは複数のコネクタを持っている場合があります。

サービス グラフは、次の要素を使ってネットワークを表します。

- 機能ノード（緑）：機能は、トランスフォーム（SSL 終了、VPN ゲートウェイ）、フィルタ（ファイアウォール）、または端末（侵入検知システム）などのトラフィックに適用されます。
- 終端ノード（青）：サービス グラフの入出力
- コネクタ（白）：ノードからの入出力
- 接続：トラフィックがネットワークを介して送られる方法

次の図に、サービス グラフを示します。

図 *8*：サービス グラフ



（注） この一般的なサービス グラフには 2 つの出力ノードがありますが、現在、ファブリックがサポートするのは 1 つのサービス グラフで入力ノード1 つ、出力ノード1 つのみです。

`serviceModify` 機能は、ネットワークと機能設定のインスタンス化に使われます。

グラフ レンダリング

デバイスで機能をインスタンス化するときに、Application Policy Infrastructure Controller（APIC）は次の操作を実行します。

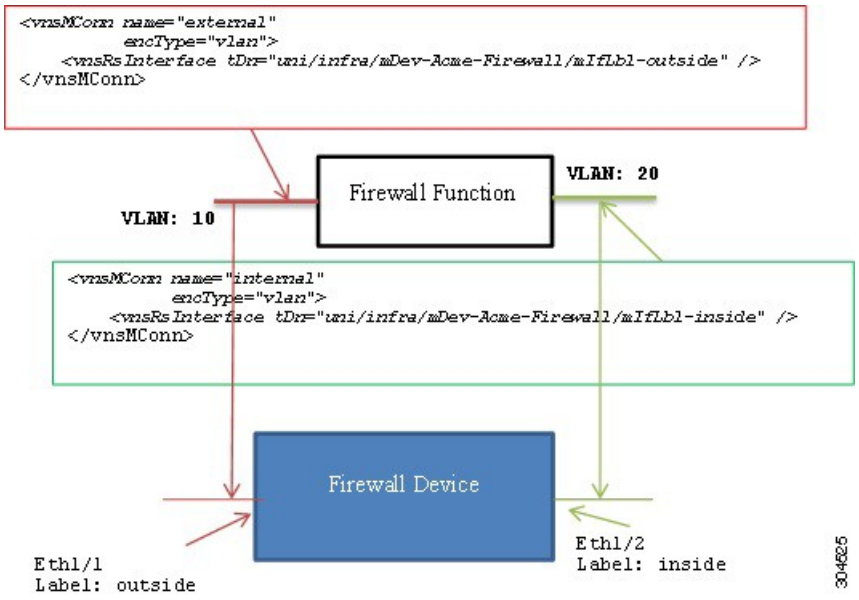
- コネクタの VLAN/VXLAN ID を割り当てます。 APIC は、以前のノードに VLAN/VXLAN ID が割り当てられているかどうかをチェックします。 以前のノード値を使用するか、またはコネクタの新しいタグを割り当てます。 `encType` は VLAN/VXLAN ID が割り当てられているかどうかを示します。
- インターフェイス関係とデバイスのインターフェイスのラベル情報を使って、インターフェイスをコネクタに関連付けます。 前の図ではファイアウォール デバイスにファイアウォール機能をレンダリングしていたので、次のようなバインディングとなります。
  - `external` としてラベル付けられるコネクタ：デバイス インターフェイス Eth1/1
  - `internal` としてラベル付けられるコネクタ：デバイス インターフェイス Eth1/2

関連付けられたインターフェイスへのコネクタに割り当てられた VLAN または VXLAN タグを有効化するか、またはバインドします。

次の図では、APIC がVLAN 10 を外部コネクタに、VLAN 20 を内部コネクタに割り当てています。デバイスのスクリプトはVLAN 10 を Eth1/1 インターフェイスに設定し、ファイアウォール機能に

バインドする必要があります。 同様に、デバイスのスクリプトはVLAN 20をEth1/2インターフェイスに設定し、ファイアウォール機能にバインドする必要があります。

図 *9*：*VLAN* がバインドされたファイアウォール



デバイスのスクリプト インターフェイス

VXLAN/VLAN カプセル化、インターフェイス、およびインターフェイスと VLAN/VXLAN 間の関連付けは、デバイス設定のパラメータとしてデバイスのスクリプトに渡されます。 前の図では、Application Policy Infrastructure Controller（APIC）がデバイス設定に次の情報を提供しています。

- カプセル化：VLAN 10、VLAN 20
- インターフェイス：Eth1/1、Eth1/2
- ロジカル インターフェイスへのカプセル化の関連付け：
  - 「Firewall-1_outside_1553」：（VLAN 10、Eth1/1）
  - 「Firewall-1_inside_7697」：（VLAN 20、Eth1/2）

カプセル化タグ、インターフェイス、および関連付けは、タグを使用するすべてのグラフにおけるすべての機能がデバイスから削除された場合のみ破棄されます。 デバイス設定のカプセル化情報を提供することで、APICは、タグを参照するすべての機能が削除されるまでカプセル化タグがデバイスから削除されないことを保証します。 次に、ディクショナリの例を示します。

```
Configuration =
{
 (0, '', <LdevInstance>): {
   ...
```

```
   (7, '', <encap-instance>): {'state': 1, 'tag': TagValue, 'type': TagType},
   (7, '', <encap-instance>): {'state': 1, 'tag': TagValue, 'type': 0},
   (8, '', <encap-association-Instance>): {
         'state': 1,
         'encap': <encapInstance>,
         'vif': <LogicalInterfaceInstanceID>}
   },
   (8, '', <encap-association-Instance>): {
         'state': 1,
         'encap': <encapInstance>,
         'vif': <LogicalInterfaceInstanceID>},
   },
   (10, '', <LogicalInterfaceInstance>): {
         'state': 0,
         'cifs': {
            'cDevInstance': <Interface Value>
         }
   },
   (10, '', <LogicalInterfaceInstance>): {
         'state': 0,
         'cifs': {
            'cDevInstance': <Interface Value>
         }
   },
 }
}

Legend:
The dictionary format is as follows:

(type, key, name): {
   'state': StateValue,
   'device': CDevName,
   'connector': connectorValue,
   'value': Parameter Value,
}


type:
7 - Encap Instance [Encap Type=0 (VLAN), Encap Type=1 (VXLAN))]
                    Encap Tag = VLAN ID or VNID (VXLAN case)
8 - VEncapAss (Device Interface and Encap (VXLAN/VLAN) association)
10 - VIF (logical interface) - Identifies interface on the device.

CDevName: Identifies a specific device within a cluster node. This attribute is not applicable
 to encap, VIF, or vEncapAss.

connectorValue: Identifies the connector to which this parameter should be bound. This
attribute is not applicable to encap, VIF, or vEncapAss.

value: Value of the parameter.

StateValue:
0 - No change
1 - Create
2 - Modify
3 - Destroy
```

機能内のコネクタは、デバイス設定で指定されたカプセル化関連付けパラメータと関係しています。 カプセル化関連付けパラメータは、特定の VLAN/VXLAN タグおよびインターフェイスにコネクタをバインドします。 上記の例では、機能用のディクショナリには次のコネクタ情報が含まれます。

```
Configuration =
{
    (0, '', 'Firewall-1'): {
      'state': 2,
      'value': {
         (7, '', '1553'): {'state': 1, 'tag': 10, 'type': 0},
         (7, '', '7697'): {'state': 1, 'tag': 20, 'type': 0},
         (8, '', 'Firewall-1_outside_1553'): {'state': 1,
```

```
                                          'encap': '1553',
                                          'vif': 'Firewall-1_outside'},
        (8, '', 'Firewall-1_inside_7697'): {'state': 1,
                                           'encap': '1553',
                                           'vif': 'Firewall-1_inside'},

        (10, '', 'Firewall-1_outside'): {'state': 1,
                                         'cifs': {'FW2': 'Eth1/1' }
                                        },
        (10, '', 'Firewall-1_inside'):  {'state': 1,
                                         'cifs': {'FW': 'Eth1/2'}}
                                        }
        (1, '', '4552'): {
           'state': 1,
           'value': {
             (3, 'Firewall', 'F1'): {
                 'state': 1,
                 'value': {
                     (2, 'external', 'conn1'): { 'state': 1,
                         'value': {
                            ('9', '', 'outside_1553'): {
                                 'state': 1,
                                 'value': 'Firewall-1-outside_1553'
                            },
                         }
                     },
                     (2, 'internal', 'conn2'): { 'state': 1
                         'value': {
                            ('9', '', 'inside_7697'): {
                                 'state': 1,
                                 'value': 'Firewall-1-inside_7697'
                            },
                         }
                     },
                     (4, 'Firewall-Config', 'FW-Config 1'): {
                          'state': 1,
                          'value' : {
                              (5, 'Param-1', ''): { 'state': 1, 'value': value },
                          . . .
                          }
                     },
                 },
             },
           },
        },
      },
}
```

上のディクショナリに基づいて、次のとおりにデバイス スクリプトを設定する必要があります。

- インターフェイス Eth1/1 上で VLAN 10を有効化。
  - encap VLAN 10 で Eth1/1.10 サブインターフェイスを作成します。
  - VLAN 10 に Eth1/1 を追加します。
- インターフェイス Eth1/2 上で VLAN 20 を有効化します。
  - encap VLAN 20 で Eth1/2.20 サブインターフェイスを作成します。
  - VLAN 20 に Eth1/2 を追加します。

（注） コネクタ値は、クラスタ内の各デバイスがさまざまなインターフェイスを使えるようにするディクショナリです。

第 5 章

サービス挿入のサポート

サービス挿入のサポートについて

ヘルス モニタリング

Application Policy Infrastructure Controller（APIC）は、次の機能を使用して、0 （非稼働）から 100（完全作動）まで、デバイスとサービスの健全性ステータスを照会することができます。

- **deviceHealth**：サービス デバイスの健全性を返します。
- **serviceHealth**：サービス機能、エンドポイント、エンドポイントのグループの健全性を返します。

APIC は、定期的に `deviceHealth()` API を呼び出します。 デバイスのスクリプトはデバイスの健全性を返すことができます。 健全性はCPU 使用率やメモリ使用状況に加え、電源装置やHA ステータスなどの重要なリソースへの照会に基づいてスクリプトが計算した正規化値です。 健全性の値は 0 から 100 のスコアで表されます。 0 はデバイスが稼働していないことを、100 はデバイスが完全に作動していることを示します。

次に、`deviceHealth()` API からの戻り値の例を示します。

```
def deviceHealth( device,interfaces,configuration ):
    …
    return {
       'state': 0,'faults': [],'health': ([],80)
    }
```

（注） ヘルス値は、メモリ使用率、CPU 使用率、および接続数に基づく正規化値です。

デバイスのスクリプトは、サービス ヘルス API を使って、サービス ノードが提供するすべての機能の健全性を報告することができます。 APIC は、サービス ノード設定でサービス ヘルス API を定期的に呼び出します。

サービス ヘルス API の定義は次のとおりです。

```
serviceHealth (device, configuration)
```

`serviceHealth` パラメータは次のように定義されています。

- `device`：デバイス IP およびクレデンシャルを提供するディクショナリ。 APIC はこの情報を使ってサービス ノードに接続します。
- `configuration`：サービス ノード設定です。 APIC は `serviceHealth` ポーリング時に、すべてのグラフにわたってデバイス設定全体をプッシュします。

サービス ヘルス API はデバイスを照会し、サービスの健全性に関わる情報を集積できます。 たとえば、スクリプトは CPU 使用率、メモリ使用率、またはサービスに関連付けられた接続数を収集することができます。 スクリプトはデバイスから集めたデータを使い、サービスの健全性を表す 0 から 100 までの正規化値を計算します。 0 は不良状態を示し、100 はサービス良好状態にあることを示します。

APIC では、スクリプトが `(path, Service Health)` のタプルのリストとしてサービス ヘルスを返すことを予期します。

`path`：デバイス内の特定のサービス機能を識別するタプルのリストです。

```
Path = [ (type, key, name) (type, key, name) ...]
```

`state` は次の値のうちの 1 つを取ることができます。

- `OK`：成功
- `TRANSIENT`：一時的エラー。 スクリプト エンジンがデバイスのスクリプト API を再度呼び出し、設定を再試行します。
- `PERMANENT`：設定エラー。 これは無効な設定パラメータ、またはデバイスにサポートされていない機能の使用が原因である可能性があります。 スクリプト エンジンはこの状態の設定を再試行しません。
- `AUDIT`：スクリプトは監査をトリガーするよう要求できます。
- `True`：成功。
- `False`：永続的エラー。 ユーザの介入を必要とする設定の問題があります。

シスコはブール演算の `True` または `False` 値ではなく、`OK`、`TRANSINET`、`PERMANENT`、または `AUDIT` を `state` 値として使うことを推奨します。

次のように、エラーは（オブジェクト、エラー）タプルのリストとして返され、システムで更新されます。

- スクリプトが `OK`、`True`、または `PERMANENT` の `state` 値を返す場合：エラーは戻り値の一連のエラーに置き換えられます。 報告されなかった以前のエラーは暗黙的に削除されます。 た

とえば、obj1 でエラーが発生したが、2 度目の試行では obj2 のみがエラーを返す場合、obj1 エラーは削除され、APIC は obj2 のエラーのみ報告します。 この戻り状態でのみエラーが置き換えられます。

- スクリプトが TRANSIENT、False、または AUDIT の state 値を報告する場合：エラーが増加します。 たとえば、スクリプトが obj1 でのエラーを報告したが、2 回目は obj2 でのみエラーが返ると、APIC は obj1 および obj2 両方のエラーを報告します。

デバイスの接続が切断されると、スクリプトはデバイスのエラーとともに TRANSIENT 状態を返します。そうでなければ、一時的なデバイスリソース問題によって設定が適用できない場合、オブジェクトの一時的エラーを返すことがあります。

無効なパラメータまたは設定の問題によって設定が適用できないときは、スクリプトはオブジェクトのエラーとともに PERMANENT 状態を返します。 ユーザはエラーを解消するために設定を変更する必要があります。

設定が正常に適用できる場合、スクリプトは OK または True 状態を返します。 成功時にエラーを書き込まないでください。 APIC はこのコール前に報告されたすべてのエラーを暗黙的にクリアします。

設定解析が失敗した場合、スクリプトはデバイスのエラーとともに PERMANENT 状態を返します。

次の例は、デバイスが SLB 機能の複数のインスタンスで設定される場合の戻り値を示しています。

```
device =
       {'creds': {'password': 'admin', 'username': 'admin'},
        'devs': {'cdev1': {'creds': {'password': 'admin',
                                     'username': 'admin'},
                           'host': '172.21.158.182',
                           'port': 80},
                 'cdev2': {'creds': {'password': 'admin',
                                     'username': 'admin'},
                                     'host': '172.21.158.224',
                                     'port': 80}},
        'host': '1.1.1.3',
        'name': 'cluster1',
        'port': 80}

configuration =
     {(0, '', 4447): {'state': 1, 'transaction': 10000,
                      'value': {(1, '', 4208): {'state': 1,
                                'transaction': 10000,
                                'value': {(3, 'SLB', 'Node1'): {'state': 1,
                                                                'transaction': 10000,
                                                                'value': { ... }
                                                               }
                                         }
                                }
                      }
     }
```

機能ごとに：

- 物理デバイスを照会します。
- 接続、CPU使用率、エラーなどのデバイスに関連する基準に基づいて各デバイスのスコアを判定します。

- クラスタのスコアを判定します。 アクティブ/アクティブ クラスタに対しては最小限のノードを使ってスコアを判定し、各デバイスからのスコアを正規化します。 アクティブ/スタンバイのクラスタに対してはスコアにアクティブ ノードを使い、またハイ アベイラビリティ状態も使用します。
- 次のフォーマットを使って、クラスタと各デバイスのヘルス スコアを返します。

```
func1 = [(0, '', 4447), (1, '', 4208), (3, 'SLB', 'Node1')]
   return { 'state': OK,
            'health': [ (func1, 100) ]
              'devs': {
                    'cdev1': {
                        'state': True,
                        'health': [ (func1, 100) ]
                    },
                    'cdev2': {
                        'state': True,
                        'health': [ (func1, 100) ]
                    },
                }
          }
```

エラー

APICにはアラーム、通知、ログのための総括的なインフラストラクチャがあり、デバイスのスクリプト内で使用できます。 デバイス パッケージの開発者は、MDfct オブジェクトを使用して一連のエラーを定義できます。 MDfct オブジェクトは MDfcts オブジェクトに含まれています。 APICによってデバイス パッケージの開発者は、MDev 内に含まれる MDfcts の 1 つのインスタンスを定義できます。 MDfcts オブジェクトには、1 つまたは複数の MDfct オブジェクトを含めることができます。 MDfcts オブジェクトと MDfct オブジェクトの階層は次のとおりです。

```
<polUni>
   <infraInfra>
      <vnsMDev …>
           <vnsMDfcts>
               <vnsMDfct …>
                   <vnsRsDfctToCat…/>
               </vnsMDfct>
           </vnsMDfcts>
     …
      </vnsMDev>
   </infraInfra>
</polUni>
```

各MDfct オブジェクトはデバイスのスクリプトが報告できるエラーのクラスを記述し、そのエラーに関する追加情報をユーザに提供します。 MDfct オブジェクトには次の属性があります。

属性	必須	説明
code	Yes	code は欠陥のクラスを一意に特定します。 デバイスのスクリプトはエラー文字列とともに code 値を返す必要があります。

属性	必須	説明
description	Yes	このフィールドはエラーについて説明します。 APIC GUI は description フィールドを使って、ユーザにヘルプを提供します。 デバイス パッケージの開発者は、エラーの正確な説明を入力する必要があります。 description フィールドのサイズは512文字以内に制限されています。
recAct	Yes	このフィールドは、エラーを解決するためユーザに推奨される修正処置を指定します。 このフィールドは、最大512文字に制限されています。
htmlFile	No	デバイス パッケージの開発者は、エラーの追加ヘルプにリンクを追加できます。

APIC はエラーを 4 カテゴリまたは重大度レベルに分類します。

- warning(1)
- minor(2)
- major(3)
- critical(4)

デバイス パッケージの開発者は、MDfct オブジェクトによって説明されるエラーを重大度のいずれかのレベルに関連付ける必要があります。 次の例に示すように、重大度との関係はvnsRsDfctToCat オブジェクトを使用して指定されます。

```
<vnsRsDfctToCat tDn="dfctCats/dfctCat-warning"/>
<vnsRsDfctToCat tDn="dfctCats/dfctCat-minor"/>
<vnsRsDfctToCat tDn="dfctCats/dfctCat-major"/>
<vnsRsDfctToCat tDn="dfctCats/dfctCat-critical"/>
```

次に、デバイス パッケージのエラー コードの定義の例を示します。

```
    <vnsMDfcts>
        <vnsMDfct code="10"
                  descr="This is a description of the fault 10"
                  recAct="Recommended action for resolving fault 10"
                  htmlFile="http://somewhere/file.html">
                 <vnsRsDfctToCat tDn="dfctCats/dfctCat-minor"/>
        </vnsMDfct>
        <vnsMDfct code="20"
                  descr="This is a description of the fault 20"
                  recAct="Recommended action for resolving fault 20"
                  htmlFile="http://somewhere/file.html">
                 <vnsRsDfctToCat tDn="dfctCats/dfctCat-major"/>
```

```
        </vnsMDfct>
    </vnsMDfcts>
```

デバイスのスクリプトは、APIのリターンディクショナリでエラーを返す場合があります。APICは、スクリプトが「エラー」を返すことを許可します。リターンディクショナリには、Pythonのタプルのリストが含まれます。エラーのリストの各タプルには、次の要素が含まれていなければいけません。

```
(object-path, code, fault string)
```

- `object-path`：オブジェクト パスは、障害が発生した設定ディクショナリ内のオブジェクトを一意に特定します。スクリプトは、設定で渡される 1 つまたは複数のオブジェクトでエラーを提起できますが、問題の修正にはユーザの介入が必要です。

  特定のテナントのコンテキストでデバイスのエラーを提起するために、スクリプトはディクショナリで渡された `vDev` としてオブジェクト パスを返すことができます。例：

  ```
  (0, '', 4133)
  ```

- `code`：スクリプトは、デバイス パッケージの `MDfct` オブジェクトに定義されているエラーコードを返す必要があります。

- `fault string`：デバイスのスクリプトは、エラーについてより具体的な情報を提供するためのエラー文字列をオプションとして追加できます。このエラー文字列はヌルでも、512 文字までの英数字文字列でもかまいません。

APICはデバイスのスクリプトが状態を `OK` として返す場合のみ、パスによって指定されるオブジェクトでエラーを更新します。リターン ディクショナリのエラーは、他のすべての状態の APIC によって無視されます。スクリプトは、API 実行中に提起されたすべてのエラーを返す必要があります。APICは、以前APIに渡されたデバイス、グラフ、機能、フォルダ、関係、またはパラメータに存在したが、リターン ディクショナリで再度報告されることがなかったエラーをすべて消去します。APIC はリターン ディクショナリで状態が `OK` の場合に、API に渡されるオブジェクトのエラーを、APIが返す一連のエラーに置き換えます。APIで渡されるオブジェクトでAPIがエラーを返し続けるかぎり、スクリプトが提起したエラーは存在し続けます。スクリプトが API で渡されるオブジェクトでエラーを返さない場合、APICは状態が `OK` という前提で、それを消去します。

デバイス スクリプトから返されるエラーは、APIC ノースバウンド API を介して照会できます。APIC GUI はまた、デバイスのスクリプトから返されるエラーを報告します。APIC はAPI によって返されたエラー コードと文字列の重大度を増大させます。

次の例で、エラー コードを定義します。

```
    <vnsMDfcts>
        <vnsMDfct code="10"
                  descr="Invalid VIP address "
                  recAct="Please enter valid unicast VIP address"
                  htmlFile="http://insieme.net/SLBCfgExample.html">
                 <vnsRsDfctToCat tDn="dfctCats/dfctCat-major"/>
        </vnsMDfct>
    </vnsMDfcts>
```

```
def serviceModify(device, configuration):
    path = [
        (0,'',1234), (1,'',2345), (3, 'Function', 'SLB'),
        (4, 'folder', 'network'), (5,'param','vip')
    ]
    code = 10
    message = '225.0.0.1'
```

```
    faults = [ (path,code,message) ]

    return {
      'state': Config.SUCCESS,
      'faults': faults,
      'health': []
    }
```

APIC はエラーを表示し、次のテキストを付記します。

```
     (APIC defect category defined in MDfct object in device package,
      Defect-code defined in MDfct object in device package,
      Defect-Description defined in MDfct object in device package,
      Fault string passed in the return dictionary by the device script)
```

`serviceModify` のエラー例では、APIC がVIP オブジェクトで次のエラー文字列を報告します。

```
Major Fault: 10, "Invalid VIP address ": '225.0.0.1'
```

カウンタ

Application Policy Infrastructure Controller（APIC）は `deviceCounter` と `serviceCounters` 機能を使ってパケット カウンタを照会することができます。その場合は、それぞれサービス機能と関連するインターフェイスとコネクタのパケットとエラー、ドロップの送受信カウンタとともにディクショナリが返されます。

`deviceCounters` API は特定のデバイスからインターフェイス統計情報を返し、次のように定義されます。

```
def deviceCounters( device,interfaces,configuration ):
```

次の例は `deviceCounters` コールです。

```
def deviceCounters( device, interfaces, configuration ):
        return {
              'state': 0,
              'counters': [ ([cif], counters), ...]

counters: {
   'rxpackets': <rxpackets>,
   'rxerrors': <rxerrors>,
   'rxdrops': <rxdrops>,
   'txpackets': <txpackets>
   'txerrors': <txerrors>
   'txdrops': <txdrops>
}
```

`cif` はインターフェイスを識別する（タイプ、キー、値）のタプルです。

例：

```
  eth0Count = {
   'rxpackets': 100,
   'rxerrors': 0,
   'rxdrops': 0
   'txpackets': 10
   'txerrors': 4
   'txdrops': 2
  }


  return {
        'state': 0,
        'counters': [ ([(11, '', 'eth0')], eth0Count) ]
  }
```

serviceCounters API はサービス機能に関連付けられたコネクタの統計情報を返し、次のように定義されます。

```
serviceCounters (device, configuration)
```

serviceCounters パラメータは次のように定義されます。

- device：デバイス IP およびクレデンシャルを提供するディクショナリです。 APIC はこの情報を使ってサービス ノードに接続します。
- configuration：サービス ノード設定です。

次に、APIC がサービス機能のパケット カウンタを照会する方法を示します。

```
def serviceCounters(device, configuration):
   externalIntferface, = [(0, 'Firewall', 4384), (1, '', 4432), (3, 'Firewall-Func', 'FW-1'),
 (2, 'external', 'external1') ]
   internalInterface = [(0, 'Firewall', 4384) (1, '', 4432) (3, 'Firewall-Func', 'FW-1'),
(2, 'internal','internal1') ]

   Firewall-1-External-Counters = (externalInterface,
                                   {  'rxpackets': 100,
                                      'rxerrors': 0,
                                      'rxdrops': 0
                                      'txpackets': 100
                                      'txerrors': 4
                                      'txdrops': 2} )

   Firewall-1-Internal-Counters =  (internalInterface,
                                    {  'rxpackets': 100,
                                       'rxerrors': 0,
                                       'rxdrops': 0
                                       'txpackets': 100
                                       'txerrors': 4
                                       'txdrops': 2} )


   Counters = [ Firewall-1-External-Counters,
                Firewall-1-Internal-Counters ]
   return {
     'state': 0,
     'counters': Counters
   }
```