Cisco APIC レイヤ 4 ～ レイヤ 7 サービス導入ガイド リリース 1.2(2x)

初版：2016 年 02 月 22 日

最終更新：2016 年 04 月 07 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 つ以上に責任を持つ、専門知識を備えたデータセンター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
- レイヤ 4 ～ レイヤ 7 サービスのインストールと管理
- スイッチおよびネットワークの管理

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。装置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 URL の Cisco.com から入手可能な次のドキュメントが含まれます。http://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html

Web ベースのマニュアル

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

ダウンロード可能なドキュメント

- ナレッジ ベースの記事（KB 記事）は、次の URL から入手できます。 http://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html
- 『*Cisco Application Centric Infrastructure Controller Release Notes*』
- 『*Cisco Application Centric Infrastructure Fundamentals Guide*』

- 『*Cisco APIC Getting Started Guide*』
- 『*Cisco ACI Basic Configuration Guide*』
- 『*Cisco ACI Virtualization Guide*』
- 『*Cisco APIC REST API User Guide*』
- 『*Cisco APIC Object Model Command Line Interface User Guide*』
- 『*Cisco APIC NX-OS Style Command-Line Interface Configuration Guide*』
- 『*Cisco APIC Faults, Events, and System Messages Management Guide*』
- 『*Cisco ACI System Messages Reference Guide*』
- 『*Cisco APIC レイヤ 4 ～ レイヤ 7 サービス導入ガイド*』
- 『*Cisco APIC Layer 4 to Layer 7 Device Package Development Guide*』
- 『*Cisco APIC Layer 4 to Layer 7 Device Package Test Guide*』
- 『*Cisco ACI Firmware Management Guide*』
- 『*Cisco ACI Troubleshooting Guide*』
- 『*Cisco APIC NX-OS Style CLI Command Reference*』
- 『*Cisco ACI Switch Command Reference, NX-OS Release 11.0*』
- 『*Verified Scalability Guide for Cisco ACI*』
- 『*Cisco ACI MIB Quick Reference*』
- 『*Cisco Nexus CLI to Cisco APIC Mapping Guide*』
- 『*Application Centric Infrastructure Fabric Hardware Installation Guide*』
- 『*Cisco NX-OS Release Notes for Cisco Nexus 9000 Series ACI-Mode Switches*』
- 『*Nexus 9000 Series ACI Mode Licensing Guide*』
- 『*Cisco Nexus 9332PQ ACI-Mode Switch Hardware Installation Guide*』
- 『*Cisco Nexus 9336PQ ACI-Mode Switch Hardware Installation Guide*』
- 『*Cisco Nexus 9372PX and 9372PX-E ACI-Mode Switch Hardware Installation Guide*』
- 『*Cisco Nexus 9372TX ACI-Mode Switch Hardware Installation Guide*』
- 『*Cisco Nexus 9396PX ACI-Mode Switch Hardware Installation Guide*』
- 『*Cisco Nexus 9396TX ACI-Mode Switch Hardware Installation Guide*』
- 『*Cisco Nexus 93128TX ACI-Mode Switch Hardware Installation Guide*』
- 『*Cisco Nexus 9504 NX-OS-Mode Switch Hardware Installation Guide*』
- 『*Cisco Nexus 9508 ACI-Mode Switch Hardware Installation Guide*』
- 『*Cisco Nexus 9516 ACI-Mode Switch Hardware Installation Guide*』

シスコ アプリケーション セントリック インフラストラクチャ（ACI）シミュレータのマニュアル

次のシスコ ACI シミュレータのマニュアルは、次の URL から入手できます。http://www.cisco.com/c/en/us/support/cloud-systems-management/application-centric-infrastructure-simulator/tsd-products-support-series-home.html

- 『*Cisco ACI Simulator Release Notes*』
- 『*Cisco ACI Simulator Installation Guide*』
- 『*Cisco ACI Simulator Getting Started Guide*』

Cisco Nexus 9000 シリーズ スイッチのマニュアル

Cisco Nexus 9000 シリーズ スイッチのマニュアルは、次の URL で入手できます。http://www.cisco.com/c/en/us/support/switches/nexus-9000-series-switches/tsd-products-support-series-home.html

Cisco Application Virtual Switch のマニュアル

Cisco Application Virtual Switch（AVS）のマニュアルは、次の URL で入手できます。http://www.cisco.com/c/en/us/support/switches/application-virtual-switch/tsd-products-support-series-home.html

シスコ アプリケーション セントリック インフラストラクチャ（ACI）と OpenStack の統合に関するマニュアル

Cisco ACI と OpenStack の統合に関するマニュアルは、次の URL から入手できます。http://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html

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

『*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

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レイヤ 4 ～ 7 サービスの導入について

従来の方法を使用する場合、サービスをネットワークに挿入すると、手間がかかって複雑なVLAN（レイヤ2）または仮想ルーティングおよび転送（VRF）インスタンス（レイヤ3）スティッチングを、ネットワーク要素およびサービス アプライアンスの間で実行する必要があります。この従来のモデルでは、アプリケーションに対する新規サービスを配備するのに数日から数週間かかります。サービスには柔軟性が少なく、操作エラーはより頻繁に発生し、トラブルシューティングはより困難です。アプリケーションが使用されなくなる場合、ファイアウォール ルールなどのサービス デバイス設定の削除は困難になります。ロードに基づいたサービスのスケール アウト/スケール ダウンを実行することもできません。

VLAN および仮想ルーティングおよび転送（VRF）スティッチングは従来のサービス挿入モデルによってサポートされますが、Application Policy Infrastructure Controller（APIC）はポリシー制御の中心点として機能する一方でサービス挿入を自動化できます。APICポリシーは、ネットワーク ファブリックとサービス アプライアンスの両方を管理します。APIC は、トラフィックがサービスを通って流れるように、ネットワークを自動的に設定できます。APICは、アプリケーション要件に従ってサービスを自動的に設定することもでき、それにより組織はサービス挿入を自動化し、従来のサービス挿入の複雑な技術の管理に伴う課題を排除できます。

APIC を使用して次のタスクを実行し、レイヤ 4 ～ 7 サービスを展開します。

1 デバイス パッケージのインポート

プロバイダーの管理者のみがデバイス パッケージをインポートできます。

2 テナントの設定
3 デバイスおよび論理インターフェイスの登録

また、このタスクでは、具象デバイスと具象インターフェイスを登録し、具象デバイス パラメータを設定します。

4 デバイス（論理デバイス）パラメータの設定
5 レイヤ 3 ネットワークの設定
6 ブリッジ ドメインの設定
7 アプリケーション プロファイルの設定
8 物理ドメインまたは VMM ドメインの設定

VMM ドメインの場合：

a VMM ドメイン クレデンシャルの設定
b vCenter/vShield コントローラ プロファイルの設定

9 VLAN プールの設定

a カプセル化ブロック範囲の設定

10 コントラクトの設定
11 管理エンドポイント グループ（EPG）の設定
12 サービス グラフ テンプレートの設定
13 アプリケーション プロファイルからのデフォルトのサービス グラフ テンプレートのパラメータの選択
14 必要に応じたサービス グラフ テンプレートのパラメータの設定
15 コントラクトへのサービス グラフ テンプレートのアタッチ
16 追加のコンフィギュレーション パラメータの設定

（注） 仮想アプライアンスは、VLAN を使用して VMware ESX サーバとリーフ ノード間にトランスポートとして導入できますが、ハイパーバイザとして導入する場合は VMware ESX のみが使用できます。

GUI を使用したレイヤ 4 ～ レイヤ 7 サービスの設定

GUI を使用して、Application Policy Infrastructure Controller（APIC）に対してレイヤ 4 ～ レイヤ 7 サービスを設定することができます。

サービスおよびサービス グラフ テンプレートを設定する手順については、GUI の使用方法, （153 ページ）を参照してください。

サービス グラフ テンプレートについて

Cisco Application Centric Infrastructure（ACI）では、特定のタイプのファイアウォールとその後に続く特定のモデルおよびバージョンのロードバランサなどの一連のメタデバイス定義できます。これは、サービスグラフテンプレート呼ばれ、また、抽象グラフとも呼ばれます。抽象サービスグラフテンプレートがコントラクトによって参照されると、サービスグラフテンプレートはファブリック内に存在するファイアウォールやロードバランサなどの具象デバイスにマッピングすることでインスタンス化されます。マッピングは「コンテキスト」の概念で発生します。「デバイス コンテキスト」は、ACI がどのファイアウォールおよびどのロード バランサをサービス グラフ テンプレートにマップできるかの識別を可能にするマッピング設定です。もう 1 つの重要な概念は、具象デバイスのクラスタを表す「論理デバイス」です。サービスグラフテンプレートのレンダリングは、コントラクトによって定義されるパスに挿入可能な適切な論理デバイスの識別に基づいています。

ACIは、アプリケーションの重要部分としてサービスを見なします。必要とされるすべてのサービスが、Cisco Application Policy Infrastructure Controller（APIC）から ACI ファブリックでインスタンス化されるサービス グラフとして扱われます。ユーザは、アプリケーションに対してサービスを定義し、サービスグラフテンプレートはアプリケーションが必要とする一連のネットワークまたはサービス機能を識別します。APIC でグラフがいったん設定されると、APIC はサービス グラフテンプレートで指定されるサービス機能要件に基づいてサービスを自動的に設定します。APIC はまた、サービスグラフテンプレートで指定されるサービス機能のニーズに応じてネットワークを自動的に設定しますが、これによってサービス デバイスでの変更は要求されません。

第 2 章

デバイス パッケージのインポート

デバイス パッケージについて

Application Policy Infrastructure Controller（APIC）は、サービス デバイスの設定およびモニタリングにデバイス パッケージを必要とします。デバイス パッケージは、サービス デバイスの単一のクラスを管理し、デバイスとその機能に関する情報を APIC に提供します。デバイス パッケージは次の項目を含む `zip` ファイルです。

デバイス仕様	次を定義する XML ファイル： • デバイス プロパティ： ◦ [Model]：デバイスのモデル。 ◦ [Vendor]：デバイスのベンダー。 ◦ [Version]：デバイスのソフトウェア バージョン。 • ロードバランシング、コンテンツ切り替え、およびSSL終端などの、デバイスによって提供される機能。 • 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。 • デバイス設定パラメータ。 • 各機能の設定パラメータ。

デバイス スクリプト	APIC からデバイスとやりとりする Python スクリプト。APIC イベントは、デバイス スクリプトで定義した関数呼び出しにマッピングされます。デバイス パッケージには、複数のデバイス スクリプトを含めることができます。
機能プロファイル	ベンダーによって指定されたデフォルト値を持つ機能パラメータ。これらのデフォルト値を使用するように機能を設定できます。
デバイスレベル設定パラメータ	デバイスに必要なパラメータを指定するコンフィギュレーションファイル。この設定は、デバイスを使用している 1 つ以上のグラフで共有できます。

デバイス パッケージを作成できます。または、デバイス ベンダーか Cisco によって提供されるものを使用できます。

REST を使用したデバイス パッケージのインストール

デバイス パッケージは HTTP または HTTPS POST を使用してインストールできます。

デバイス パッケージをインストールします。

- HTTP がApplication Policy Infrastructure Controller（APIC）で有効になっている場合の POST の URL は次のとおりです。

  ```
  http://10.10.10.10/ppi/node/mo/.xml
  ```

- HTTPS が APIC で有効になっている場合の POST の URL は次のとおりです。

  ```
  https://10.10.10.10/ppi/node/mo/.xml
  ```

メッセージには有効なセッション Cookie が必要です。

Cookie の入手方法については、『*Cisco APIC REST API User Guide*』を参照してください。

POST の本文にはアップロードされるデバイス パッケージを含める必要があります。POST で許可されるのは 1 つのパッケージだけです。

GUI を使用したデバイス パッケージのインポート

GUI を使用してデバイス パッケージをインポートします。

デバイス パッケージのインポート手順については、GUI の使用方法, （153 ページ）を参照してください。

第 3 章

論理デバイスの定義

デバイス クラスタについて

デバイス クラスタ（別名論理デバイス）は、単一のデバイスとして機能する 1 つ以上の具象デバイスです。デバイスクラスタには、そのデバイスクラスタのインターフェイス情報を説明するクラスタ（論理）インターフェイスがあります。サービス グラフ テンプレートのレンダリング時に、機能ノード コネクタはクラスタ（論理）インターフェイスに関連付けられます。Application Policy Infrastructure Controller（APIC）は、サービス グラフのインスタンス化およびレンダリング時に機能ノード コネクタにネットワーク リソース（VLAN または Virtual Extensible Local Area Network（VXLAN））を割り当て、クラスタ（論理）インターフェイスにネットワーク リソースをプログラミングします。

サービス グラフ テンプレートは、管理者が定義するデバイス選択ポリシー（論理デバイス コンテキストと呼ばれます）に基づく特定のデバイスを使用します。

管理者は、アクティブ/スタンバイ モードで最大 2 つの具象デバイスをセットアップできます。

デバイス クラスタをセットアップするには、次のタスクを実行する必要があります。

1 ファブリックに具象デバイスを接続します。

2 デバイス クラスタに管理 IP アドレスを割り当てます。

3 デバイス クラスタを APIC に登録します。APIC は、デバイス パッケージからデバイス仕様を使用してデバイスを検証します。

管理対象デバイス クラスタについて

デバイス クラスタは管理対象デバイス クラスタとして設定できます。管理対象モードでは、Application Policy Infrastructure Controller（APIC）が APIC 管理者によって APIC に提供された設定を使用し、グラフのインスタンス化時にデバイスをプログラミングします。管理対象デバイス クラスタでは、APIC にそのデバイス クラスタでデバイスを管理するためのデバイス パッケージが必要です。

デフォルトでは、デバイス クラスタは管理対象デバイス クラスタとして設定されます。

デバイス クラスタが管理対象として設定されている場合は次の設定が必要です。

- デバイス パッケージ
- 論理デバイス（vnsLDevViP）とデバイス（CDev）の接続情報（管理 IP アドレス、クレデンシャル、およびインバンド接続情報）
- サポートされる機能タイプ（go-through、go-to）に関する情報
- コンテキスト認識に関する情報（シングル コンテキストかマルチコンテキスト）

APIC はデバイス クラスタおよびデバイスのトポロジ情報（LIF、CIF）を把握する必要があります。この情報は、APIC がリーフ上で適切なポートをプログラミングできるようにするために必要であり、また、APIC はこの情報をトラブルシューティング ウィザードのために使用することもできます。さらに、APICは、カプセル化の割り当てに使用する DomP との関係も把握する必要があります。

非管理対象デバイス クラスタについて

デバイス クラスタは非管理対象デバイス クラスタとして設定できます。非管理対象デバイス クラスタでは、Application Policy Infrastructure Controller（APIC）で、サービス グラフにネットワーク リソースのみを割り当て、グラフのインスタンス化時にファブリックのみをプログラミングすることができます。これは、デバイス クラスタ内のデバイスをプログラミングする既存のオーケストレータまたは dev-op がすでに環境にある場合に便利です。また、サービス アプライアンスのデバイスパッケージが使用できない場合もあります。非管理モードでは、デバイスパッケージを必要とすることなく、APIC がサービス デバイスと連携できます。

APIC はデバイス クラスタおよびデバイスのトポロジ情報（LIF、CIF）を把握する必要があります。この情報は、APIC がリーフ上で適切なポートをプログラミングできるようにするために必要であり、また、APIC はこの情報をトラブルシューティング ウィザードのために使用することもできます。さらに、APICは、カプセル化の割り当てに使用する DomP との関係も把握する必要があります。

具象デバイスについて

具象デバイスには、具象インターフェイスがあります。具象デバイスが論理デバイスに追加されると、具象インターフェイスが論理インターフェイスにマッピングされます。サービス グラフテンプレートのインスタンス化時に、VLAN および VXLAN は、論理インターフェイスとの関連付けに基づいた具象インターフェイス上でプログラミングされます。

GUI を使用したデバイスの作成

物理デバイスにも、仮想マシンにも接続できません。接続先のタイプによって、フィールドが若干異なります。物理デバイスに接続する場合は、物理インターフェイスを指定します。仮想マシンに接続する場合は、VMM ドメイン、仮想マシン、および仮想インターフェイスを指定します。さらに、不明モデルを選択することで、接続を手動で設定することもできます。

はじめる前に

- テナントを作成しておく必要があります。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、*[tenant_name]* > [L4-L7 Services] > [L4-L7 Devices] の順に選択します。

ステップ 4 [Work] ペインで、[Actions] > [Create L4-L7 Devices] の順に選択します。

ステップ 5 [Create L4-L7 Devices] ダイアログボックスで、[General] セクションの次のフィールドに入力します。

名前	説明
[Managed] チェックボックス	管理対象デバイスを作成する場合はこのチェックボックスをオンにします。非管理対象デバイスを作成する場合はこのチェックボックスをオフにします。
[Name] フィールド	デバイスの名前を入力します。
[Service Type] ドロップダウン リスト	サービス タイプを選択します。
[Device Type] ボタン	デバイス タイプを選択します。
[Physical Domain] ドロップダウン リストまたは [VMM Domain] ドロップダウン リスト	物理ドメインまたは VMM ドメインを選択します。

名前	説明
[Mode] オプション ボタン	デバイスのモードを選択します。
[Device Package] ドロップダウン リスト	（管理対象デバイスの場合のみ）デバイスのモードを選択します。
[Model] ドロップダウン リスト	（管理対象デバイスの場合のみ）使用するベンダー提供のパッケージを選択します。

ステップ 6 （管理対象デバイスのみ）[Connectivity] セクションで、次のフィールドに入力します。

名前	説明
[APIC to Device Management Connectivity] オプション ボタン	接続のタイプを選択します。ファブリックの外側のデバイスに接続するときは [Out-of-Band] を選択し、ファブリックを通じてデバイスに接続するときは [In-Band] を選択します。

ステップ 7 （管理対象デバイスのみ）[Credentials] セクションで、次のフィールドに入力します。

名前	説明
[User Name] フィールド	ユーザ名を入力します。
[Password] フィールド	パスワードを入力します。
[Confirm Password] フィールド	もう一度パスワードを入力します。

ステップ 8 [Device 1] セクションで、次のフィールドに入力します。

名前	説明
[Management IP Address] フィールド	（管理対象デバイスの場合のみ）接続するデバイスの管理 IP アドレスを入力します。
[Management Port] フィールドとドロップダウン リスト	（管理対象デバイスの場合のみ）管理ポートを入力するか、またはドロップダウン リストから値を選択します。
[VM] ドロップダウン リスト	（仮想デバイス タイプの場合のみ）仮想マシンを選択します。
[Chassis] ドロップダウン リスト	（管理対象デバイスの場合のみ）シャーシを選択します。

ステップ 9 [Device Interfaces] テーブルで、[+] ボタンをクリックしてインターフェイスを追加し、次のフィールドに入力します。

名前	説明
[Name] ドロップダウン リスト	インターフェイス名を選択します。
[VNIC] ドロップダウン リスト	（仮想デバイス タイプの場合のみ）vNIC を選択します。
[Path] ドロップダウン リスト	インターフェイスの接続先のポート、ポート チャネル、またはバーチャル ポート チャネルを選択します。

ステップ 10 [Update] をクリックします。

ステップ 11 （HA クラスタの場合のみ）各デバイスのフィールドに入力します。

ステップ 12 [Cluster] セクションのフィールドに入力します。

ステップ 13 [Next] をクリックします。
使用しているパッケージで実行可能な機能とパラメータのリストが [Device Configuration] ページに表示されます。基本パラメータが表示された [Basic] タブと、デバイス パッケージで使用可能なすべてのパラメータが表示された [All Parameters] タブが表示されます。基本パラメータは [All Parameters] に含まれています。

ステップ 14 [Features] セクションで、使用する一連の機能を選択します。
使用する特定のパッケージと選択した特定の機能に応じて、一連のパラメータは変化します。

ステップ 15 選択した機能のパラメータに対して、次のように値を指定します。

a) 変更するフィールドをダブルクリックします。
b) 表示されたフィールドに必要な情報を入力します。
c) [Update] をクリックします。

ステップ 16 [Finish] をクリックします。

NX-OS スタイルの CLI を使用したデバイスの作成

物理デバイスにも、仮想マシンにも接続できません。物理デバイスに接続する場合は、物理インターフェイスを指定します。仮想マシンに接続する場合は、VMM ドメイン、仮想マシン、および仮想インターフェイスを指定します。

はじめる前に

- テナントを作成しておく必要があります。

ステップ 1 コンフィギュレーション モードを開始します。

例：
```
apic1# configure
```

ステップ 2 テナントのコンフィギュレーション モードを開始します。
```
tenant tenant_name
```

例：
```
apic1(config)# tenant t1
```

ステップ 3 レイヤ 4 ～ レイヤ 7 デバイス クラスタを追加します。
```
l4l7 cluster name cluster_name type cluster_type vlan-domain domain_name
  [function function_type] [service service_type]
```

パラメータ	説明
name	デバイス クラスタの名前。
type	デバイス クラスタのタイプ。値は次のとおりです。 • virtual • physical
vlan-domain	VLAN の割り当てに使用するドメイン。このドメインは、仮想デバイスの場合はVMMドメイン、物理デバイスの場合は物理ドメインである必要があります。
function	（任意）機能タイプ。値は次のとおりです。 • go-to • go-through
service	（任意）サービス タイプ。ADC 固有またはファイアウォール固有のアイコンおよび GUI を表示するために GUI で使用します。値は次のとおりです。 • ADC • FW • OTHERS

例：
物理デバイスの場合は、次のように入力します。

```
apic1(config-tenant)# l4l7 cluster name D1 type physical vlan-domain phys
  function go-through service ADC
```

仮想デバイスの場合は、次のように入力します。

```
apic1(config-tenant)# l4l7 cluster name ADCCluster1 type virtual vlan-domain mininet
```

ステップ 4 1 つ以上のクラスタ デバイスをデバイス クラスタに追加します。

```
cluster-device device_name [vcenter vcenter_name] [vm vm_name]
```

パラメータ	説明
vcenter	（仮想デバイスの場合のみ）仮想デバイスの仮想マシンをホストする VCenter の名前。
vm	（仮想デバイスの場合のみ）仮想デバイスの仮想マシンの名前。

例：
物理デバイスの場合は、次のように入力します。

```
apic1(config-cluster)# cluster-device C1
apic1(config-cluster)# cluster-device C2
```

仮想デバイスの場合は、次のように入力します。

```
apic1(config-cluster)# cluster-device C1 vcenter vcenter1 vm VM1
apic1(config-cluster)# cluster-device C2 vcenter vcenter1 vm VM2
```

ステップ 5 1 つ以上のクラスタ インターフェイスをデバイス クラスタに追加します。

```
cluster-interface interface_name [vlan static_encap]
```

パラメータ	説明
vlan	（仮想デバイスの場合のみ）クラスタ インターフェイスのスタティックなカプセル化。VLAN の値は、1 ～ 4094 とする必要があります。

例：
物理デバイスの場合は、次のように入力します。

```
apic1(config-cluster)# cluster-interface consumer vlan 1001
```

仮想デバイスの場合は、次のように入力します。

```
apic1(config-cluster)# cluster-interface consumer
```

ステップ 6 1 つ以上のメンバーをクラスタ インターフェイスに追加します。

```
member device device_name device-interface interface_name
```

パラメータ	説明
device	**cluster-device** コマンドを使用して、このデバイスにすでに追加されている必要があるクラスタ デバイスの名前。
device-interface	クラスタ デバイス上のインターフェイスの名前。

例：
```
apic1(config-cluster-interface)# member device C1 device-interface 1.1
```

ステップ 7 メンバーにインターフェイスを追加します。

```
interface {ethernet ethernet_port | port-channel port_channel_name [fex fex_ID] |
  vpc vpc_name [fex fex_ID]} leaf leaf_ID
```

インターフェイスではなく vNIC を追加する場合は、このステップをスキップします。

パラメータ	説明
ethernet	（イーサネットまたはFEXイーサネットインターフェイスの場合のみ）クラスタ デバイスが Cisco Application Centric Infrastructure（ACI）ファブリックに接続されているリーフ上のイーサネット ポート。FEXイーサネット メンバーを追加する場合は、FEX IDと FEX ポートの両方を次の形式で指定します。 *FEX_ID*/*FEX_port* 次に例を示します。 101/1/23 FEX ID は、クラスタ デバイスがファブリック エクステンダにどこで接続するかを指定します。
port-channel	（ポートチャネルまたはFEXポートチャネルインターフェイスの場合のみ）クラスタ デバイスが ACIファブリックに接続するポート チャネル名。
vpc	（バーチャル ポート チャネルまたは FEX バーチャル ポート チャネル インターフェイスの場合のみ）クラスタ デバイスが ACIファブリックに接続するバーチャル ポート チャネル名。
fex	（ポート チャネル、FEX ポート チャネル、バーチャル ポート チャネル、または FEX バーチャル ポートの場合のみ）ポート チャネルまたはバーチャル ポート チャネルの形成に使用するスペース区切りリスト形式の FEX ID。
leaf	クラスタ デバイスがどこで接続するかのスペース区切りリスト内のリーフ ID。

例：
イーサネット インターフェイスの場合は、次のように入力します。

```
apic1(config-member)# interface ethernet 1/23 leaf 101
apic1(config-member)# exit
```

FEX イーサネット インターフェイスの場合は、次のように入力します。

```
apic1(config-member)# interface ethernet 101/1/23 leaf 101
apic1(config-member)# exit
```

ポート チャネル インターフェイスの場合は、次のように入力します。

```
apic1(config-member)# interface port-channel pc1 leaf 101
apic1(config-member)# exit
```

FEX ポート チャネル インターフェイスの場合は、次のように入力します。

```
apic1(config-member)# interface port-channel pc1 leaf 101 fex 101
apic1(config-member)# exit
```

バーチャル ポート チャネル インターフェイスの場合は、次のように入力します。

```
apic1(config-member)# interface vpc vpc1 leaf 101 102
apic1(config-member)# exit
```

FEX バーチャル ポート チャネル インターフェイスの場合は、次のように入力します。

```
apic1(config-member)# interface vpc vpc1 leaf 101 102 fex 101 102
apic1(config-member)# exit
```

ステップ 8 メンバーに vNIC を追加します。

```
vnic "vnic_name"
```

vNIC の代わりにインターフェイスを追加する場合は、前のステップを参照してください。

パラメータ	説明
vnic	クラスタ デバイスの仮想マシンの VNIC アダプタの名前。名前を二重引用符で囲みます。

例：

```
apic1(config-member)# vnic "Network adapter 2"
apic1(config-member)# exit
```

ステップ 9 デバイスの作成が完了したら、コンフィギュレーション モードを終了します。

例：

```
apic1(config-cluster-interface)# exit
apic1(config-cluster)# exit
apic1(config-tenant)# exit
apic1(config)# exit
```

REST API を使用したインポートされたデバイスの使用

次の REST API ではインポートされたデバイスを使用します。

```
<polUni>
  <fvTenant dn="uni/tn-tenant1" name="tenant1">
    <vnsLDevIf ldev="uni/tn-mgmt/lDevVip-ADCCluster1"/>
    <vnsLDevCtx ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any">
      <vnsRsLDevCtxToLDev tDn="uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]"/>
      <vnsLIfCtx connNameOrLbl="inside">
        <vnsRsLIfCtxToLIf
tDn="uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-inside"/>
        <fvSubnet ip="10.10.10.10/24"/>
        <vnsRsLIfCtxToBD tDn="uni/tn-tenant1/BD-tenant1BD1"/>
      </vnsLIfCtx>
      <vnsLIfCtx connNameOrLbl="outside">
        <vnsRsLIfCtxToLIf
tDn="uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-outside"/>
        <fvSubnet ip="70.70.70.70/24"/>
        <vnsRsLIfCtxToBD tDn="uni/tn-tenant1/BD-tenant1BD4"/>
      </vnsLIfCtx>
    </vnsLDevCtx>
  </fvTenant>
</polUni>
```

NX-OS スタイルの CLI を使用した別のテナントからのデバイスの作成

共有サービスのシナリオでは、別のテナントからデバイスをインポートできます。

ステップ 1 コンフィギュレーション モードを開始します。

例：

```
apic1# configure
```

ステップ 2 テナントのコンフィギュレーション モードを開始します。

```
tenant tenant_name
```

例：

```
apic1(config)# tenant t1
```

ステップ 3 デバイスをインポートします。

```
l4l7 cluster import-from tenant_name device-cluster device_name
```

パラメータ	説明
import-from	デバイスのインポート元のテナントの名前。
device-cluster	指定したテナントからインポートするデバイス クラスタの名前。

例：

```
apic1(config-tenant)# l4l7 cluster import-from common device-cluster d1
apic1(config-import-from)# end
```

オブジェクト モデル CLI を使用したインポートされたデバイスの使用

次のコマンドでは、オブジェクト モデル CLI を使用してインポートされたデバイスを使用します。

```
# logical-device-context
cd '/aci/tenants/tenant1/l4-l7-services/device-cluster-selection-policies'
mocreate 'any' 'any' 'any'
cd 'any-any-any'
moset device-cluster
'tenants/tenant1/l4-l7-services/imported-device-clusters/[uni/tn-mgmt/lDevVip-ADCCluster1]'
moconfig commit


# vns-lifctx
cd
'/aci/tenants/tenant1/l4-l7-services/device-cluster-selection-policies/any-any-any/logical-interface-contexts'
mocreate 'inside'

cd 'inside'

moset logical-interface
'uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-inside'
moset bridge-domain 'tenants/tenant1/networking/bridge-domains/tenant1BD1'

moconfig commit


# fv-subnet
cd
'/aci/tenants/tenant1/l4-l7-services/device-cluster-selection-policies/any-any-any/logical-interface-contexts/inside/subnets'
mocreate '10.10.10.10/24'

moconfig commit


# vns-lifctx
cd
'/aci/tenants/tenant1/l4-l7-services/device-cluster-selection-policies/any-any-any/logical-interface-contexts'
mocreate 'outside'

cd 'outside'

moset logical-interface
'uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-outside'
moset bridge-domain 'tenants/tenant1/networking/bridge-domains/tenant1BD4'

moconfig commit


# fv-subnet
cd
'/aci/tenants/tenant1/l4-l7-services/device-cluster-selection-policies/any-any-any/logical-interface-contexts/outside/subnets'
mocreate '70.70.70.70/24'
```

```
moconfig commit


# logical-device-consumer
cd '/aci/tenants/tenant1/l4-l7-services/imported-device-clusters'
mocreate 'uni/tn-mgmt/lDevVip-ADCCluster1'
cd '[uni--tn-mgmt--lDevVip-ADCCluster1]'
moset faultcode '0'
moconfig commit
```

GUI を使用したデバイスのインポートの確認

GUI を使用して、デバイスが正常にインポートされたことを確認することができます。

ステップ 1 メニュー バーで、[TENANTS] タブをクリックします。[Tenant] ウィンドウが表示されます。
ステップ 2 サブメニュー バーで、デバイスをインポートするテナントの名前をクリックします。
ステップ 3 [Navigation] ペインで、テナントのブランチを展開します。
ステップ 4 [L4-L7 Services] ブランチを展開します。
ステップ 5 [Imported Devices] ブランチを展開します。
ステップ 6 適切なデバイスを選択します。デバイス情報が [Work] ペインに表示されます。

第 4 章

デバイスへの接続の設定

デバイスのインバンド管理について

Cisco Application Centric Infrastructure（ACI）は、Cisco Application Centric Infrastructure（ACI）ファブリックを通過する各テナントのインバンド内のデバイスを管理するメカニズムを提供します。この設定オプションでは、インフラ テナントと管理テナント内でのルーティングを可能にするためにデバイスで使用される管理 IP アドレスを必要とすることなく、デバイスの管理接続が実現します。

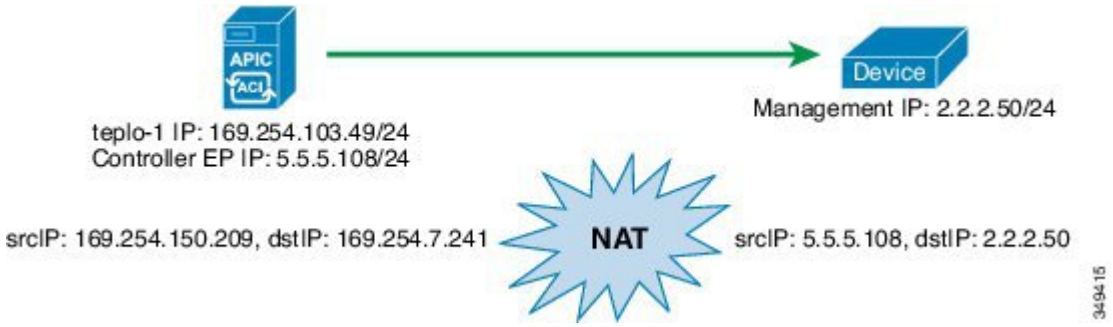
（注） この機能は、ACI とファブリック ノードのインバンド管理からは切り離されています。デバイスのインバンドを管理するには、ファブリックのインバンド管理は必要ありません。

ACI とデバイス間のインバンド管理通信は、ACI に一意の IP アドレスを設定することによって可能になります。この IP アドレスはコントローラ エンドポイントと呼ばれています。これらの IP アドレスは実際には ACI に設定されませんが、その代わりに、ネットワーク アドレス変換（NAT）と共に使用してデバイスとの管理通信を確立します。ACI が使用する NAT アドレスは ACI によって自動的に選択され、169.254.0.0/16 のアドレス範囲内に収まります。

また、各デバイス管理 IP アドレスは変換後の IP アドレスとして ACI に提示されます。この変換後のアドレスを、マップされたホスト アドレスと呼びます。

次の図に、ACI とデバイス間のアドレス変換を示します。

図 *1* : *ACI* とデバイス間のネットワーク アドレス変換



GUI を使用したデバイスのインバンド管理の設定

GUI を使用してデバイスにインバンド管理を設定することができます。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、*[tenant_name]* > [L4-L7 Services] > [L4-L7 Devices] の順に選択します。

ステップ 4 [Work] ペインで、[Actions] > [Create L4-L7 Devices] の順に選択します。

ステップ 5 [Create L4-L7 Devices] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [APIC to Device Management Connectivity] オプション ボタンに [In-Band] を選択します。

b) [EPG] ドロップダウン リストで、[Create Management EPG] を選択します。

ステップ 6 [Create Management EPG] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [Application Profile] ドロップダウン リストで、EPG を配置する既存のアプリケーション プロファイルを選択します。オプションで、新しいアプリケーションプロファイルを作成するには [Create Application Profile] を選択します。
新しいアプリケーション プロファイルを作成する場合は、[EPG] セクションと [Contracts] セクションは空白のままにします。

b) [Name] フィールドに、管理 EPG の名前を入力します。

c) [Bridge Domain] ドロップダウン リストで、ドメインを選択します。

d) [Domains] で、ドメイン プロファイルを追加します。

e) [Reserved IP addresses for APICs] セクションで、[+] をクリックして新しい IP アドレス プールを作成します。

ステップ 7 [Create IP Address Pool] ダイアログボックスで、すべてのフィールドに入力し、[OK] をクリックします。

IP アドレス プールは、コントローラのエンドポイント アドレスを定義します。プール内の IP アドレスは、デバイスが Application Policy Infrastructure Controller（APIC）の IP アドレスと見なす IP アドレスです。

コントローラのエンドポイントに定義したアドレス範囲が、デバイスに定義した管理 IP アドレスと同じサブネットに含まれていない場合は、デバイスにネクストホップ ゲートウェイを提供する管理EPGブリッジ ドメインの下にサブネットを定義して、コントローラのエンドポイントに到達するようにする必要があります。

ステップ 8 [Create Management EPG] ダイアログボックスで、[Submit] をクリックします。
これで、管理 EPG のドメイン名が設定されました。

ステップ 9 [Create L4-L7 Devices] ダイアログボックスで、デバイスのセットアップを実行します。インターフェイスの設定に管理インターフェイスを必ず含めてください。

GUI を使用したデバイスのインバンド管理のトラブルシューティング

既存のエンドポイント グループ（EPG）をデバイスの管理 EPG として選択した場合は、管理 IP アドレス プールとコントローラ管理ポリシーを手動で追加する必要があります。GUI を使用してこれらを追加することができます。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、*[tenant_name]* > [Application Profiles] > *[application_profile_name]* > [Application EPGs] > *[EPG_name]* > [L4/L7 IP Address Pool] の順に選択します。

ステップ 4 [Work] ペインで、[Actions] > [Create Address Pool] の順に選択します。

ステップ 5 [Create IP Address Pool] ダイアログボックスで、必要に応じてフィールドに入力します。
これで、管理 IP プールが追加されます。

ステップ 6 [Navigation] ペインで、*[tenant_name]* > [L4-L7 Services] > [Inband Management Configuration for L4-L7 devices] の順に選択します。

ステップ 7 [Work] ペインの [Controller Management Policies] セクションで、[+] をクリックし、次のようにフィールドに入力します。

a) [Private Networks] ドロップダウン リストで、プライベート ネットワークを選択します。

b) [Address Pool] ドロップダウンで、作成したばかりのプールを選択します。

ステップ 8 [Update] をクリックします。
これで、コントローラ管理ポリシーが追加されます。

第 5 章

グラフをレンダリングするレイヤ 4 ～ レイヤ 7 デバイスの選択

デバイス（論理デバイス）のコンテキストについて

デバイスは、コントラクト名、グラフ名、またはグラフ内の機能ノード名に基づいて選択できます。デバイスを作成した後は、デバイスに選択条件ポリシーを提供するデバイス コンテキストを作成できます。

デバイス コンテキストは、サービス グラフ テンプレートにデバイスを選択するポリシーを指定します。これにより、管理者は複数のデバイスを持つことができ、それらを異なるサービス グラフ テンプレートに対して使用することができます。たとえば、管理者は、高いパフォーマンス ADC アプライアンスがあるデバイスと、パフォーマンスが低い ADC アプライアンスがある別のデバイスを持つことができます。高いパフォーマンスの ADC デバイス用と低いパフォーマンスの ADC デバイス用の 2 つの異なるデバイス コンテキストを使用して、管理者は高いパフォーマンスが必要となるアプリケーションには高いパフォーマンスの ADC デバイスを選択し、低いパフォーマンスが必要なアプリケーションには低いパフォーマンスの ADC デバイスを選択することができます。

GUI を使用した論理デバイス コンテキストの作成

[Apply L4-L7 Service Graph Template To EPGs] ウィザードを使用せずにサービス グラフ テンプレートを適用した場合は、論理デバイス コンテキスト（論理デバイス選択ポリシーとも呼ぶ）を設定する必要がある場合があります。論理デバイス コンテキストはグラフのレンダリングに使用する

ファイアウォールやロード バランサについて Cisco Application Centric Infrastructure（ACIコンテキスト）に指示します

[Apply L4-L7 Service Graph Template To EPGs] ウィザードを使用してサービス グラフ テンプレートを適用した場合は、論理デバイス コンテキストが自動的に作成されているため、手動で設定する必要はありません。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、[Tenant]*[tenant_name]* > [L4-L7 Services] > [Devices Selection Policies] の順に選択します。

ステップ 4 [Work] ペインで、[Actions] > [Create Logical Device Context] の順に選択します。

ステップ 5 [Create Logical Device Context] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [Service Type] ドロップダウン リストで、論理デバイス コンテキストのコントラクトを選択します。デバイスを使用する条件の一部としてコントラクト名を使用しない場合は、[any] を選択します。

b) [Graph Name] ドロップダウン リストで、論理デバイス コンテキストのグラフを選択します。デバイスを使用する条件の一部としてグラフ名を使用しない場合は、[any] を選択します。

c) [Node Name] ドロップダウン リストで、論理デバイス コンテキストのノードを選択します。デバイスを使用する条件の一部としてグノード名を使用しない場合は、[any] を選択します。

ステップ 6 [Cluster Interface Contexts] セクションの [+] をクリックしてクラスタ インターフェイス コンテキストを追加します。

- [Connector Name]：サービス グラフ テンプレートのコネクタの名前。
- [Logical Interface]：論理インターフェイス コンテキストで指定するコネクタに使用する論理インターフェイス。
- [Bridge Domain]：論理インターフェイス コンテキストで指定されているブリッジ ドメイン。
- [Subnets]：サービス グラフ テンプレートがインスタンス化される場合に論理インターフェイスで設定するサブネット。

ステップ 7 [Submit] をクリックします。

REST API を使用したデバイス選択ポリシーの設定

REST API を使用してデバイス選択ポリシーを設定することができます。

REST API を使用したデバイス コンテキストの作成

次の REST API はデバイス コンテキストを作成します。

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevCtx ctrctNameOrLbl="webCtrct" graphNameOrLbl="G1" nodeNameOrLbl="Node1">
            <vnsRsLDevCtxToLDev tDn="uni/tn-acme/lDevVip-ADCCluster1"/>

             <!-- The connector name C4, C5, etc.. should match the
                  Function connector name used in the service graph template -->

             <vnsLIfCtx connNameOrLbl="C4">
                <vnsRsLIfCtxToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/LIf-ext"/>
            </vnsLIfCtx>
            <vnsLIfCtx connNameOrLbl="C5">
                <vnsRsLIfCtxToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/LIf-int"/>
            </vnsLIfCtx>
        </vnsLDevCtx>
    </fvTenant>
</polUni>
```

REST API を使用したデバイスでの論理インターフェイスの追加

次の REST API はデバイス内に論理インターフェイスを追加します。

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1">

            <!-- The LIF name defined here (such as e.g., ext, or int) should match the
                 vnsRsLIfCtxToLIf `tDn' defined in LifCtx  -->

            <vnsLIf name="ext">

                <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-outside"/>
                <vnsRsCIfAtt tDn="uni/tn-acme/lDevVip-ADCCluster1/cDev-ADC1/cIf-ext"/>
            </vnsLIf>
            <vnsLIf name="int">
                <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-inside"/>
                <vnsRsCIfAtt tDn="uni/tn-acme/lDevVip-ADCCluster1/cDev-ADC1/cIf-int"/>
            </vnsLIf>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

オブジェクト モデルの CLI を使用したデバイス選択ポリシーの設定

ここで示すコマンドで、オブジェクト モデルの CLI を使用してデバイス選択ポリシーを設定することができます。

（注） NX OS スタイルの CLI を使用すると、デバイス選択ポリシーは自動的に設定されますが、同等の NX-OS スタイルの CLI コマンドはありません。

```
# logical-device-context

cd '/aci/tenants/acme/l4-l7-services/device-cluster-selection-policies'
mocreate 'any' 'any' 'any'
cd 'any-any-any'
moset device-cluster 'tenants/acme/l4-l7-services/device-clusters/InsiemeCluster'
moconfig commit


# vns-lifctx

cd
'/aci/tenants/acme/l4-l7-services/device-cluster-selection-policies/webCtrt-G1-Node1/logical-interface-contexts'
mocreate 'int'
cd 'int'
moset logical-interface
'tenants/acme/l4-l7-services/device-clusters/InsiemeCluster/logical-interfaces/int'
moset bridge-domain ‘tenants/acme/networking/bridge-domains/MySrvrBD'
moconfig commit


# vns-lifctx

cd
'/aci/tenants/acme/l4-l7-services/device-cluster-selection-policies/webCtrt-G1-Node1/logical-interface-contexts'
mocreate ‘ext'
cd 'ext'
moset logical-interface
‘tenants/acme/l4-l7-services/device-clusters/InsiemeCluster/logical-interfaces/ext'
moset bridge-domain ‘tenants/acme/networking/bridge-domains/MyClntBD'
moconfig commit
```

第 6 章

サービス グラフの設定

サービス グラフについて

サービス グラフは、端末セット間の順序付けられた一連の機能ノードで、アプリケーションが必要とする一連のネットワーク サービス機能を識別します。グラフ内のサービス機能は、アプリケーションの要件に基づいたサービス デバイスに自動的にプロビジョニングされます。

GUI、CLI、または Application Policy Infrastructure Controller（APIC）を使用してサービス グラフを定義できます。APIC を通じたサービス デバイスの設定では、サービス デバイスの変更は必要ありません。

機能ノードについて

機能ノードは、単一のサービス機能を表します。機能ノードには、サービス機能のネットワーク要件を表す機能ノード コネクタがあります。

サービス グラフ内の機能ノードは、1つ以上のパラメータが必要になる場合があります。パラメータは、エンドポイント グループ（EPG）、アプリケーション プロファイル、またはテナント コン

テキストで指定できます。パラメータは、サービス グラフ定義時に割り当てることができます。パラメータ値は変更がさらに加えられるのを防ぐためにロックできます。

機能ノード コネクタについて

機能ノード コネクタは、サービス グラフに機能ノードを接続し、グラフのコネクタ サブネットに基づいて適切なブリッジ ドメインと接続と関連付けられます。各コネクタは、VLAN または Virtual Extensible LAN（VXLAN）に関連付けられます。コネクタの両側がエンドポイント グループ（EPG）として扱われ、ホワイトリストがスイッチにダウンロードされ、2つの機能ノード間の通信がイネーブルになります。

サービス グラフ接続について

サービス グラフ接続は、1 つの機能ノードを別の機能ノードに接続します。

端末ノードについて

端末ノードはサービス グラフとコントラクトを接続します。コントラクトに端末ノードを接続することにより、2台のアプリケーションエンドポイント グループ（EPG）間のトラフィックにサービス グラフを挿入できます。接続されると、コントラクトのコンシューマ EPG とプロバイダー EPG 間のトラフィックはサービス グラフにリダイレクトされます。

サービスグラフテンプレートのコンフィギュレーション パラメータについて

サービス グラフ テンプレートは、デバイス パッケージによって指定される、コンフィギュレーション パラメータを持つことができます。コンフィギュレーション パラメータは、EPG、アプリケーション プロファイルまたはテナント コンテキストでも指定できます。サービス グラフ テンプレート内の機能ノードでは、1 つ以上のコンフィギュレーション パラメータが必要になる場合があります。パラメータ値は変更がさらに加えられるのを防ぐためにロックできます。

サービス グラフテンプレートを設定し、コンフィギュレーション パラメータの値を指定すると、Application Policy Infrastructure Controller（APIC）はデバイス パッケージ内のデバイス スクリプトにパラメータを渡します。デバイス スクリプトは、パラメータ データをデバイスにダウンロードされる設定に変換します。

GUI を使用したサービス グラフ テンプレートの設定

GUI を使用して、サービス グラフ テンプレートを設定することができます。

サービス グラフ テンプレートを設定する手順については、GUI の使用方法, （153 ページ）を参照してください。

REST API を使用したサービス グラフ テンプレートの作成

次の REST API を使用してサービス グラフ テンプレートを作成することができます。

```
<polUni>
  <fvTenant name="acme">
    <vnsAbsGraph name="G1">
      <vnsAbsTermNodeCon name="Input1">
        <vnsAbsTermConn name="C1">
                </vnsAbsTermConn>
      </vnsAbsTermNodeCon>
      <vnsAbsNode name="Node" funcType="GoTo">
        <vnsRsDefaultScopeToTerm
          tDn="uni/tn-acme/AbsGraph-G1/AbsTermNodeProv-Output1/outtmnl"/>
        <vnsAbsFuncConn name="inside">
          <vnsRsMConnAtt
            tDn="uni/infra/mDev-Insieme-Generic-1.0/mFunc-SubnetFunc/mConn-external"/>
        </vnsAbsFuncConn>
        <vnsAbsFuncConn name="outside">
          <vnsRsMConnAtt
            tDn="uni/infra/mDev-Insieme-Generic-1.0/mFunc-SubnetFunc/mConn-internal"/>
        </vnsAbsFuncConn>
        <vnsAbsDevCfg>
          <vnsAbsFolder key="oneFolder" name="f1">
            <vnsAbsParam key="oneParam" name="p1" value="v1"/>
          </vnsAbsFolder>
        </vnsAbsDevCfg>
        <vnsAbsFuncCfg>
          <vnsAbsFolder key="folder" name="folder1" devCtxLbl="C1">
            <vnsAbsParam key="param" name="param" value="value"/>
          </vnsAbsFolder>
          <vnsAbsFolder key="folder" name="folder2" devCtxLbl="C2">
            <vnsAbsParam key="param" name="param" value="value"/>
          </vnsAbsFolder>
        </vnsAbsFuncCfg>
        <vnsRsNodeToMFunc tDn="uni/infra/mDev-Insieme-Generic-1.0/mFunc-SubnetFunc"/>
      </vnsAbsNode>
      <vnsAbsTermNodeProv name="Output1">
        <vnsAbsTermConn name="C6">
                </vnsAbsTermConn>
      </vnsAbsTermNodeProv>
      <vnsAbsConnection name="CON1">
        <vnsRsAbsConnectionConns
          tDn="uni/tn-acme/AbsGraph-G1/AbsTermNodeCon-Input1/AbsTConn"/>
       <vnsRsAbsConnectionConns tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node/AbsFConn-inside"/>

      </vnsAbsConnection>
      <vnsAbsConnection name="CON3">
       <vnsRsAbsConnectionConns tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node/AbsFConn-outside"/>

        <vnsRsAbsConnectionConns
          tDn="uni/tn-acme/AbsGraph-G1/AbsTermNodeProv-Output1/AbsTConn"/>
      </vnsAbsConnection>
    </vnsAbsGraph>
  </fvTenant>
</polUni>
```

NX-OS スタイルの CLI を使用したサービス グラフの設定

NX-OS スタイルの CLI を使用して、サービス グラフを設定することができます。

ステップ 1 コンフィギュレーション モードを開始します。

例：
```
apic1# configure
```

ステップ 2 テナントのコンフィギュレーション モードを開始します。
```
tenant tenant_name
```

例：
```
apic1(config)# tenant t1
```

ステップ 3 サービス グラフを追加します。
```
l4l7 graph graph_name [contract contract_name]
```

パラメータ	説明
graph	サービス グラフの名前。
contract	このサービス グラフ インスタンスに関連付けられたコントラクトの名前。サービス グラフ インスタンスを作成する場合にのみ、コントラクトを指定します。インスタンス化せずに（サービス グラフ テンプレートと同様に）簡単にサービス グラフを設定できます。

例：
```
apic1(config-tenant)# l4l7 graph G2 contract C2
```

ステップ 4 サービス グラフにノード（サービス）を追加します。
```
service node_name [device-cluster-tenant tenant_name] [device-cluster device_name] [mode
deployment_mode]
```

パラメータ	説明
service	追加するサービス ノードの名前。
device-cluster-tenant	デバイス クラスタのインポート元のテナント。グラフを設定するテナントと同じテナントにデバイス クラスタがない場合にのみ、このパラメータを指定します。
device-cluster	このサービス ノードに使用するデバイス クラスタの名前。

パラメータ	説明
mode	導入モード。値は次のとおりです。 • ADC_ONE_ARM：ワンアーム モードを指定します。 • ADC_TWO_ARM：ツーアーム モードを指定します。 • FW_ROUTED：ルーテッド（GoTo）モードを指定します。 • FW_TRANS：トランスペアレント（GoThrough）モードを指定します。 • OTHERS：他の導入モードを指定します。 モードを指定しないと、導入モードは使用されません。

例：
次に、ノード N1 をテナント t1 からデバイス クラスタ D4 に追加する例を示します。

```
apic1(config-graph)# service N1 device-cluster-tenant t1 device-cluster D4
```

次に、ノード N1 をテナント t1 からデバイス クラスタ D4 に追加し、ルーテッド導入モードを使用する例を示します。

```
apic1(config-graph)# service N1 device-cluster-tenant t1 device-cluster D4 mode FW_ROUTED
```

ステップ 5 コンシューマ コネクタを追加します。

```
connector connector_type [cluster-interface interface_type]
```

パラメータ	説明
connector	サービス グラフ内のコネクタのタイプ。値は次のとおりです。 • provider • consumer
cluster-interface	デバイス クラスタ インターフェイスのタイプ。値は次のとおりです。 • provider • consumer テナント Common 内のサービス グラフ テンプレートの場合は、このパラメータを指定しないでください。

例：
```
apic1(config-service)# connector consumer cluster-interface consumer
```

ステップ 6 ブリッジ ドメイン情報と、そのブリッジ ドメインが存在するテナントを指定し、コネクタにブリッジ ドメインを設定します。

```
bridge-domain tenant tenant_name name bridge_domain_name
```

パラメータ	説明
tenant	ブリッジ ドメインを所有するテナント。同じテナントまたはテナント Common からのみ、ブリッジを指定できます。たとえば、テナント t1 の場合、テナント t2 からのブリッジ ドメインは指定できません。
name	ブリッジ ドメインの名前。

例：
```
apic1(config-connector)# bridge-domain tenant t1 name bd2
```

ステップ 7 （任意） コネクタの Direct Server Return（DSR）仮想 IP アドレス（VIP）を設定します。

```
dsr-vip ip_address
```

DSR VIP を指定した場合、Application Policy Infrastructure Controller（APIC）は VIP を取得しません。

パラメータ	説明
dsr-vip	コネクタの DSR の仮想 IP アドレス。

例：
```
apic1(config-connector)# dsr-vip 192.168.10.100
```

ステップ 8 コンシューマとプロバイダーに対する接続を設定して、サービス グラフ コンフィギュレーション モードを終了します。

```
connection connection_name {terminal terminal_type service node_name connector connector_type} |
  {intra_service service1 node_name connector1 connector_type service2 node_name connector2
connector_type}
exit
```

パラメータ	説明
connection	接続の名前。

パラメータ	説明
terminal	サービス ノードを端末に接続します。端末のタイプを指定します。値は次のとおりです。 • provider • consumer
service service1 service2	追加するサービス ノードの名前。service は terminal でのみ使用します。service1 と service2 は、intra_service でのみ使用します。
connector connector1 connector2	コネクタのタイプ。値は次のとおりです。 • provider • consumer connector は terminal でのみ使用します。connector1 と connector2 は intra_service でのみ使用します。
intra_service	別のノードにサービス ノードを接続します。

例：
次に、単一ノード グラフの接続を設定する例を示します。

```
apic1(config-graph)# connection CON1 terminal consumer service N1 connector consumer
apic1(config-graph)# connection CON2 terminal provider service N2 connector provider
apic1(config-graph)# exit
```

次に、2 ノード グラフの接続を設定する例を示します。

```
apic1(config-graph)# connection CON1 terminal consumer service N1 connector consumer
apic1(config-graph)# connection CON2 intra_service service1 N1 connector1 provider service2 N2
connector2 consumer
apic1(config-graph)# connection CON3  terminal provider service N2 connector provider
apic1(config-graph)# exit
```

ステップ 9 コンフィギュレーション モードを終了します。

例：

```
apic1(config-tenant)# exit
apic1(config)# exit
```

第 7 章

ルート ピアリングの設定

ルート ピアリングについて

ルート ピアリングは、トランジットの使用例としてより一般的なCisco Application Centric Infrastructure（ACI）ファブリックの特殊ケースで、ルート ピアリングによってACI ファブリックが Open Shortest Path First（OSPF）プロトコルまたは Border Gateway Protocol（BGP）プロトコルのトランジット ドメインとして機能できるようになります。ルート ピアリングの一般的な使用例はルート ヘルス インジェクションであり、サーバのロード バランシング仮想 IP が OSPF または内部 BGP（iBGP）を使用して、ACI ファブリック外にあるクライアントにアドバタイズされます。デバイスが接続されている ACI リーフ ノードとピアリングしたり、ルートを交換したりできるように、ルート ピアリングを使用して OSPF ピアリングや BGP ピアリングをサーバ デバイス上に設定することができます。

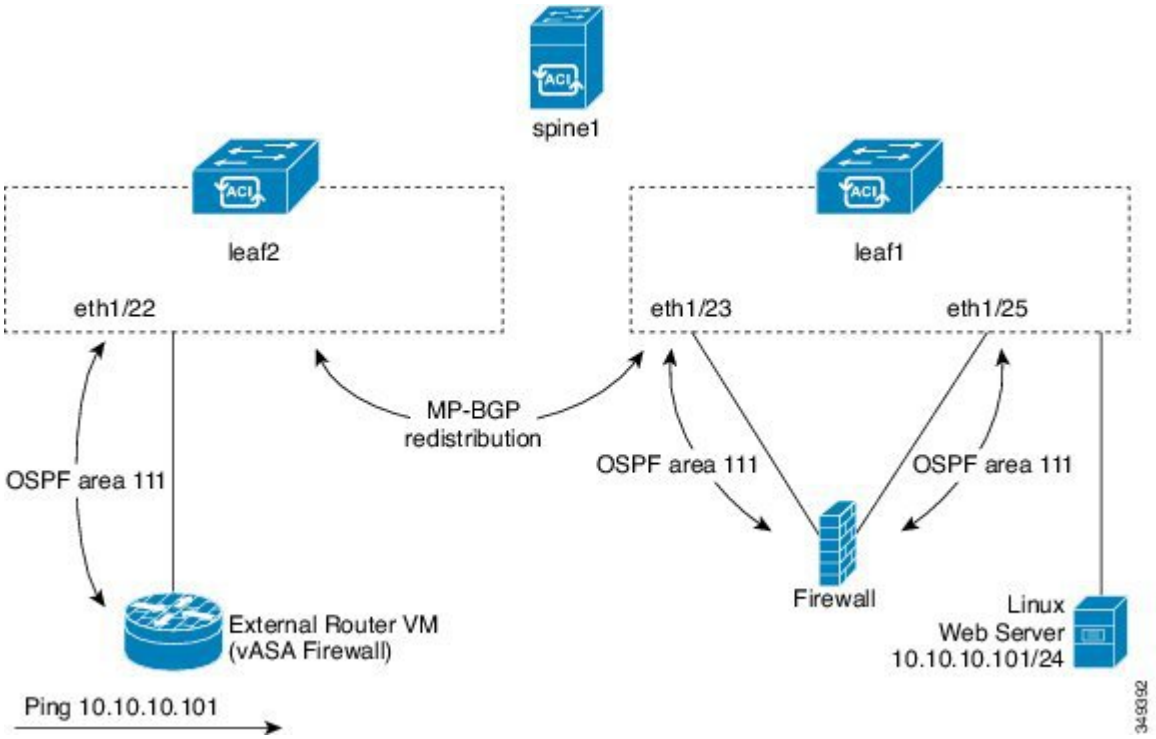
次のプロトコルは、ルート ピアリングをサポートしています。

- OSPF
- OSPFv3

- iBGPv4
- iBGPv6
- 静的ルート

次の図に、ルート ピアリングの一般的な導入方法を示します。

図 *2*：一般的なルート ピアリング トポロジ



図に示すように、ルート ピアリングを設定してサービス グラフを導入することによって、Web サーバのパブリック IP アドレスがファイアウォールを介して外部ルータにアドバタイズされます。ファイアウォールの各レッグに OSPF ルーティング ポリシーを導入する必要があります。通常、これを行うには、`l3extOut` ポリシーを導入します。これにより、Web サーバの到達可能性情報がファイアウォールを介してボーダー リーフと外部 ルータに OSPF でアドバタイズされるようになります。

ファブリック内のリーフ間のルート配布は Multi-Protocol Border Gateway Protocol（MP-BGP）により内部的に実行されます。

ルート ピアリング トポロジのより詳しい例については、ルート ピアリングのエンドツーエンド フロー, （45 ページ） を参照してください。

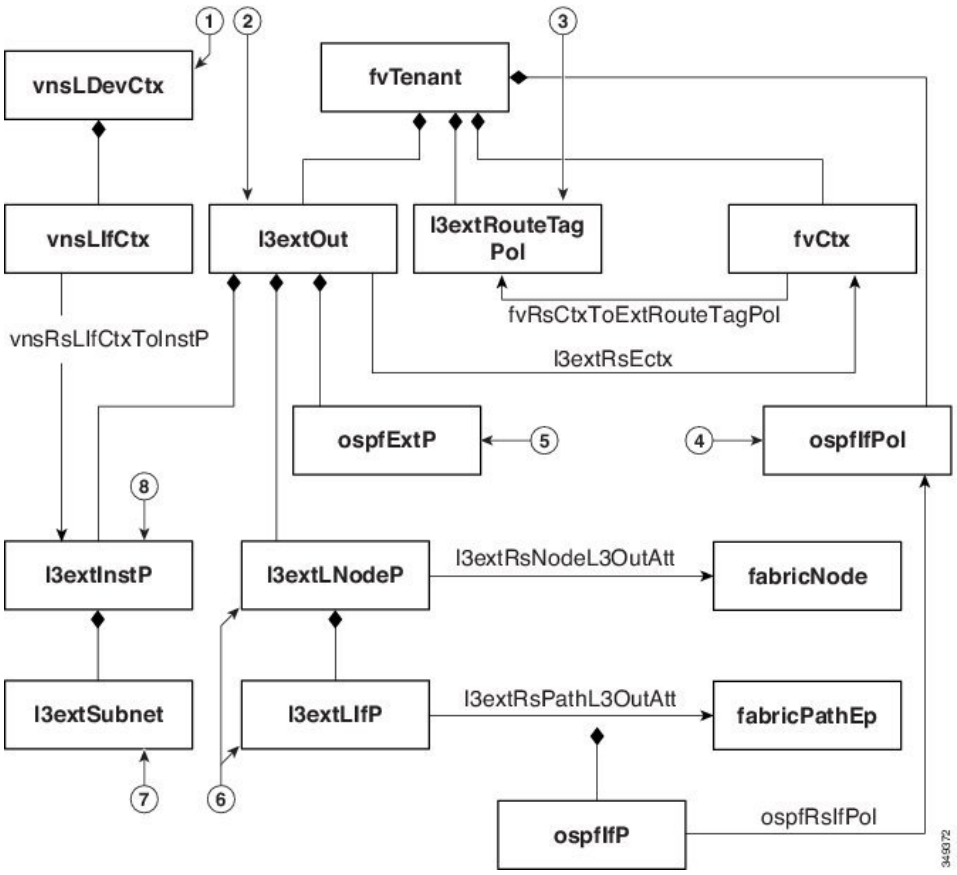
`l3extOut` ポリシーの詳細については、『*Cisco Application Centric Infrastructure Fundamentals Guide*』を参照してください。

（注） ポイントツーポイントの非ブロードキャスト モードは、Adaptive Security Appliance（ASA）ではサポートされていません。ポイントツーポイントの非ブロードキャスト モード設定が存在する場合は、Application Policy Infrastructure Controller（APIC）からその設定を削除する必要があります。

Open Shortest Path First ポリシー

ルート ピアリングを設定するには、最初に 1 つ以上の `l3extOut` ポリシーを作成し、サービス デバイスを接続するファブリック リーフ ノードに導入します。これらの `l3extOut` ポリシーで、ファブリック リーフで有効にする必要がある Open Shortest Path First（OSPF）のパラメータを指定します。これらのポリシーは外部通信に使用される `l3extOut` ポリシーとよく似ています。次の図に、ルート ピアリング オブジェクトの関係を示します。

図 *3* : *OSPF* ルート ピアリング オブジェクトの関係



1 `vnsLDevCtx`：デバイス選択ポリシー。

2. l3extOut：1 つのエリアのすべての OSPF ポリシーが含まれます。
3. l3extRouteTagPol：ルート ピアリングで必要な各コンテキストには OSPF ループを回避するための一意のルート タグが必要です。1 つのレッグから取得される OSPF ルートは、ルート タグが異なっていない限り、他のレッグでは取得されません。
4. ospfIfPol：インターフェイスごとの OSPF ポリシー。
5. ospfExtP：エリア ポリシーごとの OSPF。
6. l3extLNodeP/l3extLIfP：この l3extOut を導入するノードまたはポート。
7. l3extSubnet：ファブリックに対してエクスポートまたはインポートするサブネット。
8. l3extInstP：プレフィックス ベースの EPG。

次に、l3extOut の 2 つの 例（OspfExternal と OspfInternal）を示します。これらのポリシーは、図 2：一般的なルート ピアリング トポロジ, （36 ページ） のファイアウォール デバイスの外部レッグと内部レッグに導入されます。l3extOut ポリシーは、ファブリック リーフがトラフィックを分類する方法と、サービス デバイスに対してルートをインポートまたはエクスポートする方法も制御する 1 つ以上のプレフィックス ベースの EPG（l3extInstP）を指定します。l3extOut ポリシーには、そのポリシーの下で指定される OSPF のエリアごとのポリシー（ospfExtP）と 1 つ以上の OSPF インターフェイス ポリシー（ospfIfPol）が含まれています。

次に、値「100」で設定される area-Id を持つ OSPF エリアの例を示します。

```
<ospfExtP areaId="100" areaType="regular" areaCtrl="redistribute"/>
```

エリア タイプは「regular」に設定し、エリア制御属性は「redistribute」に設定します。

OSPF インターフェイス ポリシーで、1 つ以上の OSPF インターフェイス タイマーを指定します。

```
<ospfIfPol name="ospfIfPol" ctrl="mtu-ignore" nwT="bcast" xmitDelay="1" helloIntvl="10"
  deadIntvl="40" status="created,modified"/>
```

デフォルト タイマーが正常であれば、このポリシーを指定する必要はありません。このポリシーでは、特定のタイマーをデフォルト値から変更し、次の関係を使用することによって、1 つ以上のインターフェイスに関連付けることができます。

```
<l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]" ifInstT="ext-svi"
  encap="vlan-3844" addr="30.30.30.100/28" mtu="1500"/>
```

l3extRsPathL3OutAtt の関係の属性は次のとおりです。

- ifInstT：論理インターフェイス タイプ。通常は「ext-svi」。
- encap：このインターフェイスを作成するときは VLAN カプセル化を指定する必要があります。カプセル化はサービス デバイスにプッシュされます。
- addr：この l3extOut を導入するときにファブリック リーフで作成された SVI インターフェイスの IP アドレス。

次のポリシーで、l3extOut ポリシーをどこに導入するかを制御します。

```
<l3extNodeP name="bLeaf-101">
  <l3extRsNodeL30utAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.11"/>
  <l3extLIfP name="portlf">
    <l3extRsPathL30utAtt tDn="topology/pod-1/paths-101/pathep-teth1/251"
      ifInstT="ext-svi" encap="vlan-3844" addr="30.30.30.100/28" mtu="1500"/>
```

```
    <ospfIfP authKey="tecom" authType="md5" authKeyId='1'>
      <ospfRslfPol tnOspfIfPolName="ospfIfPol"/>
    </ospflfP>
  </l3extLIfP>
</l3extLNodeP>
```

l3extOut ポリシーは、サービス デバイスが接続されているリーフ ポートと同じものに導入する必要があります。

scope=import-security 属性は次を実行します。

- データ プレーン内のトラフィックのフローを制御する
- このルートをアドバタイズする外部デバイスへのディレクティブとして機能する

（注） ルート ピアリングを正しく動作させるには、l3extRsPathL3OutAtt の関係が、デバイスを表す vnsCDev の下の RsCIfPathAtt の関係と同じファブリックの宛先を指している必要があります。

OspfExternal ポリシー

```
<polUni>
  <fvTenant name="common">
    <fvCtx name="commonctx">
      <fvRsCtxToExtRouteTagPol tnL3extRouteTagPolName="myTagPol"/>
    </fvCtx>
    <l3extRouteTagPol tag="212" name="myTagPol"/>
    <l3extOut name="OspfExternal" status="created,modified">
      <l3extLNodeP name="bLeaf-101">
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.8/28"/>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
            ifInstT="ext-svi" encap="vlan-3843" addr="40.40.40.100/28" mtu="1500"/>
          <ospfIfP authKey="tecom" authType="md5" authKeyId='1'>
            <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
          </ospfIfP>
        </l3extLIfP>
      </l3extLNodeP>
      <ospfExtP areaId="100" areaType="regular" areaCtrl="redistribute"/>
      <l3extInstP name="ExtInstP">
        <l3extSubnet ip="40.40.40.100/28" scope="import-security"/>
        <l3extSubnet ip="10.10.10.0/24" scope="import-security"/>
      </l3extInstP>
      <l3extRsEctx tnFvCtxName="commonctx"/>
    </l3extOut>
   <ospfIfPol name="ospfIfPol" ctrl="mtu-ignore" nwT="bcast" xmitDelay="1" helloIntvl="10"

      deadIntvl="40" status="created,modified"/>
  </fvTenant>
</polUni>
```

OspfInternal ポリシー

```
<polUni>
  <fvTenant name="tenant1">
    <l3extRouteTagPol tag="213" name="myTagPol"/>
    <fvCtx name="tenant1ctx1">
      <fvRsCtxToExtRouteTagPol tnL3extRouteTagPolName="myTagPol"/>
    </fvCtx>
    <l3extOut name="OspfInternal" status="created,modified">
      <l3extLNodeP name="bLeaf-101">
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.11"/>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]"
```

```
            ifInstT="ext-svi" encap="vlan-3844" addr="30.30.30.100/28" mtu="1500"/>
          <ospfIfP authKey="tecom" authType="md5" authKeyId='1'>
            <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
          </ospfIfP>
        </l3extLIfP>
      </l3extLNodeP>
      <ospfExtP areaId="100" areaType="regular" areaCtrl="redistribute"/>
      <l3extInstP name="IntInstP">
        <l3extSubnet ip="30.30.30.100/28" scope="import-security"/>
        <l3extSubnet ip="20.20.20.0/24" scope="import-security"/>
      </l3extInstP>
      <l3extRsEctx tnFvCtxName="tenant1ctx1"/>
    </l3extOut>
   <ospfIfPol name="ospfIfPol" ctrl="mtu-ignore" nwT="bcast" xmitDelay="1" helloIntvl="10"

      deadIntvl="40" status="created,modified"/>
  </fvTenant>
</polUni>
```

`OspfExternalInstP` ポリシーは、プレフィクスの 40.40.40.100/28 と 10.10.10.0/24 をプレフィクス ベースのエンドポイントのアソシエーションに使用する必要があることを指定します。また、このポリシーは、プレフィックスの 20.20.20.0/24 をサービス デバイスにエクスポートするようにファブリックに指示します。

```
      <l3extInstP name="OspfExternalInstP">
        <l3extSubnet ip="40.40.40.100/28" scope="import-security"/>
        <l3extSubnet ip="10.10.10.0/24" scope="import-security"/>
        <l3extSubnet ip="20.20.20.0/24" scope="export"/>
      </l3extInstP>
```

`bleaf-101` ポリシーは、この `l3extOut` ポリシーを導入する場所を制御します。

```
      <l3extLNodeP name="bLeaf-101">
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.8/28"/>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
            ifInstT="ext-svi" encap="vlan-3843" addr="40.40.40.100/28" mtu="1500"/>
          <!-- <ospfIfP authKey="tecom" authType="md5" authKeyId='1'> -->
          <ospfIfP>
            <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
          </ospfIfP>
        </l3extLIfP>
      </l3extLNodeP>
```

仮想サービス

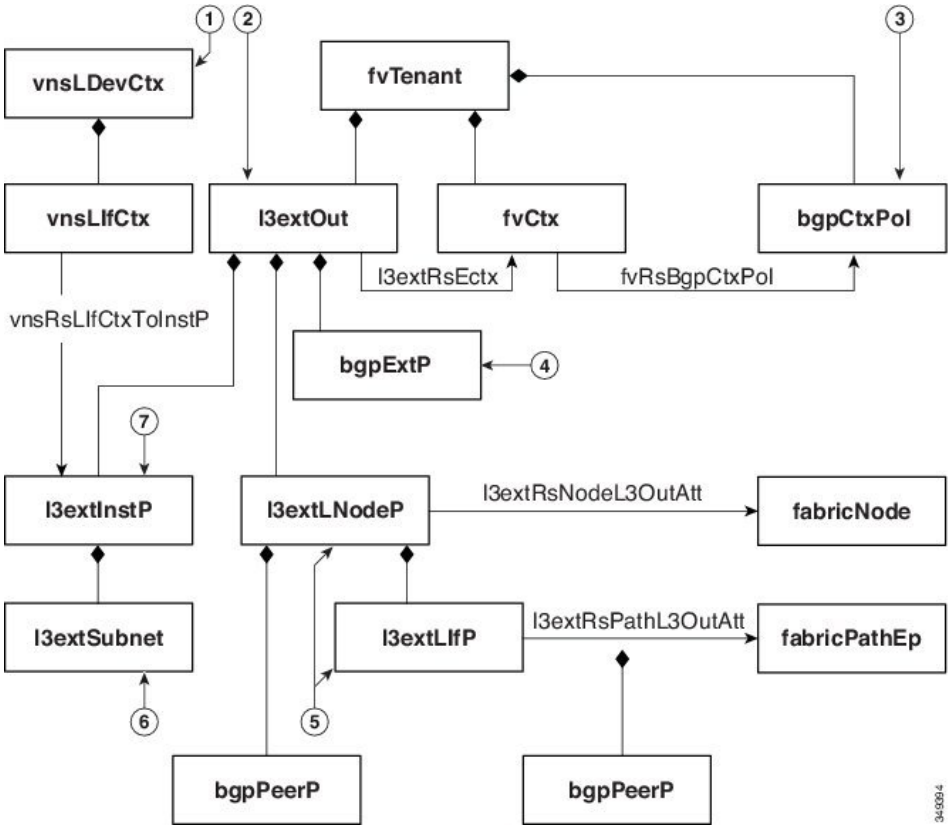
仮想サービスはルート ピアリングとともに導入できますが、`vnsCIf` オブジェクトでの `l3extRsPathL3OutAtt` 検証は実行されません。このデータ パスは、`l3extOut` オブジェクトが仮想サービス データが接続されている正しいリーフに導入されている場合にのみ動作します。

Border Gateway Protocol ポリシー

内部 Border Gateway Protocol（iBGP）を使用してデバイスの外部インターフェイスにルート ピアリングを設定し、内部インターフェイスに静的ルートを設定できます。追加設定なしにデバイスの内部インターフェイスと外部インターフェイスの両方に iBGP を設定することはできません。これは、インターフェイスが異なる自律システムに存在する必要があり、相互自律システム再配布ポリシーをプッシュ ダウンしないためです。

次の図に、ルート ピアリング オブジェクトの関係を示します。

図 *4*：*iBGP* ルート ピアリング オブジェクトの関係



1. `vnsLDevCtx`：デバイス選択ポリシー。
2. `l3extOut`：単一の自律システム用のすべての BGP ポリシーが含まれます。
3. `bgpCtxPol`：コンテキスト単位の BGP タイマー。
4. `bgpExtP`：ASN ポリシー単位の BGP。
5. `l3extLIfP/l3extLNodeP`：これらのエンドポイント グループ（EPG）を導入するノードまたはポートを制御します。
6. `l3extSubnet`：ファブリックからのエクスポートするサブネットとファブリックにインポートするサブネット。
7. `l3extInstP`：プレフィックス ベースの EPG。

次のポリシーは、外部インターフェイスに iBGPv4/v6 を設定します。

```
<polUni>
  <fvTenant name="common">
    <fvCtx name="commonctx">
      <fvRsBgpCtxPol tnBgpCtxPolName="timer-3-9"/>
      <fvRsCtxToExtRouteTagPol tnL3extRouteTagPolName="myTagPol"/>
    </fvCtx>
```

```
    <l3extRouteTagPol tag="212" name="myTagPol"/>
    <bgpCtxPol grCtrl="helper" holdIntvl="9" kaIntvl="3" name="timer-3-9" staleIntvl="30"/>

    <l3extOut name="BgpExternal" status="created,modified">
      <l3extLNodeP name="bLeaf-101">
        <!-- <bgpPeerP addr="40.40.40.102/32 "ctrl="send-com"/> -->
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.8/28>"
          <l3extLoopBackIfP addr="50.50.50.100/32"/>
        </l3extRsNodeL3OutAtt>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
            ifInstT="ext-svi" encap="vlan-3843" addr="40.40.40.100/28 "mtu="1500">
            <bgpPeerP addr="40.40.40.102/32 "ctrl="send-com"/>
          </l3extRsPathL3OutAtt>
        </l3extLIfP>
      </l3extLNodeP>
      <bgpExtP/>
      <l3extInstP name="ExtInstP">
        <l3extSubnet ip="40.40.40.100/28 "scope="import-security"/>
        <l3extSubnet ip="10.10.10.0/24 "scope="import-security"/>
        <l3extSubnet ip="20.20.20.0/24 "scope="export-rtctrl"/>
      </l3extInstP>
      <l3extRsEctx tnFvCtxName="commonctx"/>
    </l3extOut>
  </fvTenant>
</polUni>
```

iBGP ピアは、物理インターフェイス レベルまたはループバック レベルで設定できます。次に、物理インターフェイス レベルで設定された iBGP ピアの例を示します。

```
<l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
            ifInstT="ext-svi" encap="vlan-3843" addr="40.40.40.100/28 "mtu="1500">
            <bgpPeerP addr="40.40.40.102/32 "ctrl="send-com"/>
          </l3extRsPathL3OutAtt>
</l3extLIfP>
```

この場合、ファブリック上で実行する iBGP プロセスはスイッチ仮想インターフェイス（SVI）IP アドレス 40.40.40.100/28 を使用して、ネイバーとピアリングします。ネイバーは、IP アドレス 40.40.40.102/32 のサービス デバイスです。

次に、iBGP ピアの定義が論理ノード レベル（`l3extLNodeP` の下）に移動され、ループバック インターフェイスが作成されている例を示します。

```
      <l3extLNodeP name="bLeaf-101">
        <bgpPeerP addr="40.40.40.102/32 "ctrl="send-com"/>
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.8/28>"
          <l3extLoopBackIfP addr="50.50.50.100/32"/>
        </l3extRsNodeL3OutAtt>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
            ifInstT="ext-svi" encap="vlan-3843" addr="40.40.40.100/28 "mtu="1500">
          </l3extRsPathL3OutAtt>
        </l3extLIfP>
      </l3extLNodeP>
```

この例では、iBGP プロセスはループバック アドレスを使用してネイバーとピアリングします。ループバックが設定されていない場合は、ファブリックは rtrId で指定された IP アドレスを使用してネイバーとピアリングします。

この場合、デバイスには SVI に到達するルートが必要です。通常、これは、IP アドレス 50.50.50.0 が IP アドレス 40.40.40.100 から到達できる場合は、次の ASA の例に示すようにグラフ パラメータを使用して設定します。

```
<vnsAbsFolder name="ExtRouteCfg" key="StaticRoute">
              <vnsAbsFolder name="route1" key="route">
                <vnsAbsParam name="network" key="network" value="50.50.50.0"/>
                <vnsAbsParam name="netmask" key="netmask" value="255.255.255.0"/>
```

```
              <vnsAbsParam name="gateway" key="gateway" value="40.40.40.100"/>
            </vnsAbsFolder>
            <vnsAbsFolder name="route2" key="ipv6_route">
              <vnsAbsParam name="prefix" key="prefix" value="2005::/64"/>
              <vnsAbsParam name="gateway" key="gateway" value="2004::2828:2866"/>
            </vnsAbsFolder>
</vnsAbsFolder>
```

次に、デバイスの内部インターフェイス用にファブリック上で静的ルートを設定する例を示します。

```
<polUni>
  <fvTenant name="tenant11">
    <l3extOut name="StaticInternal" status="created,modified">
      <l3extLNodeP name="bLeaf-201">
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.11>"
          <ipRouteP ip="20.20.20.0/24>"
            <ipNexthopP nhAddr="30.30.30.102/32"/>
          </ipRouteP>
        </l3extRsNodeL3OutAtt>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]"
            ifInstT="ext-svi" encap="vlan-3844" addr="30.30.30.100/28 "mtu="1500"/>
        </l3extLIfP>
      </l3extLNodeP>
      <l3extInstP name="IntInstP">
        <l3extSubnet ip="20.20.20.0/24 "scope="import-security"/>
      </l3extInstP>
      <l3extRsEctx tnFvCtxName="tenant1ctx1"/>
    </l3extOut>
  </fvTenant>
</polUni>
```

クラスタ用の L3extOut ポリシーの選択

特定の `l3extOut` ポリシーを、選択ポリシー `vnsLIfCtx` を使用して論理デバイスのインターフェイスに関連付けることができます。次に、これを実現する例を示します。

```
<vnsLDevCtx ctrctNameOrLbl="webCtrct1" graphNameOrLbl="WebGraph" nodeNameOrLbl="FW">
  <vnsRsLDevCtxToLDev tDn="uni/tn-tenant1/lDevVip-Firewall"/>
  <vnsRsLDevCtxToRtrCfg tnVnsRtrCfgName="FwRtrCfg"/>
  <vnsLIfCtx connNameOrLbl="internal">
    <vnsRsLIfCtxToInstP tDn="uni/tn-tenant1/out-OspfInternal/instP-IntInstP"
      status="created,modified"/>
    <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-internal"/>
  </vnsLIfCtx>
  <vnsLIfCtx connNameOrLbl="external">
    <vnsRsLIfCtxToInstP tDn="uni/tn-common/out-OspfExternal/instP-ExtInstP"
      status="created,modified"/>
    <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-external"/>
  </vnsLIfCtx>
</vnsLDevCtx>
```

`vnsRsLIfCtxToInstP` の関係を使用して、サービス デバイスのこのレッグと関連付ける特定のプレフィックス ベースの EPG（`l3extInstP`）を選択します。この関係に、`redistribute` プロトコル再配布プロパティを指定できます。`redistribute` プロパティのデフォルト値は「ospf,bgp」です。`redistribute` をデフォルト値のままにすると、各レッグで構成されているルーティング プロトコルがApplication Policy Infrastructure Controller（APIC）によって自動検出され、適切な再配布設定にプッシュされます。自動設定は、常に Interior Gateway Protocol（OSPF）から外部ゲートウェイ プロトコル（BGP）に再配布します。

静的または接続済みといった特定の再配布設定を使用する場合は、それらの設定をこの関係に追加します。たとえば、redistribute="ospf,bgp,static" は、自動検出設定と redistribute-static をサービス デバイスにプッシュします。

このプロパティをデフォルト値を含まない特定の値（たとえば、redistribute="ospf,static,connected"）に設定すると、それらの設定がそのままサービス デバイスにプッシュされます。これは、APIC によって選択されたデフォルト値を上書きする場合に役に立ちます。

（注） この関係は l3extOut 自体でなく、EPG（l3extInstP）を指します。これは、l3extOut ポリシーにはこのような EPG が複数存在する可能性があり、別のデバイス選択ポリシーがそれらの EPG を指していることがあるためです。これにより、さまざまなサービス グラフによってインポートまたはエクスポートされるプレフィックスを細かく制御できます。

vnsRsLDevCtxToRtrCfg の関係を使用して、特定の vnsRtrCfg ポリシーをこのデバイス セレクタに選択します。vnsRtrCfg ポリシーで、Open Shortest Path First（OSPF）や内部ボーダー ゲートウェイ プロトコル（IBGP）などのルーティング プロトコルで使用するルータ ID を指定するようにする必要があり、これらのポリシーはユーザが指定する必要があります。このルータ ID はデバイスに送信されます。

次のコードで、vnsRtrCfg ポリシーの例を示します。

```
<vnsRtrCfg name="FwRtrCfg" rtrId="180.0.0.10”/>
```

関連付けられた具象デバイスには vnsRsCIfPathAtt オブジェクトが必要です。このオブジェクトでは、デバイスを同じファブリック リーフに導入します（下記参照）。

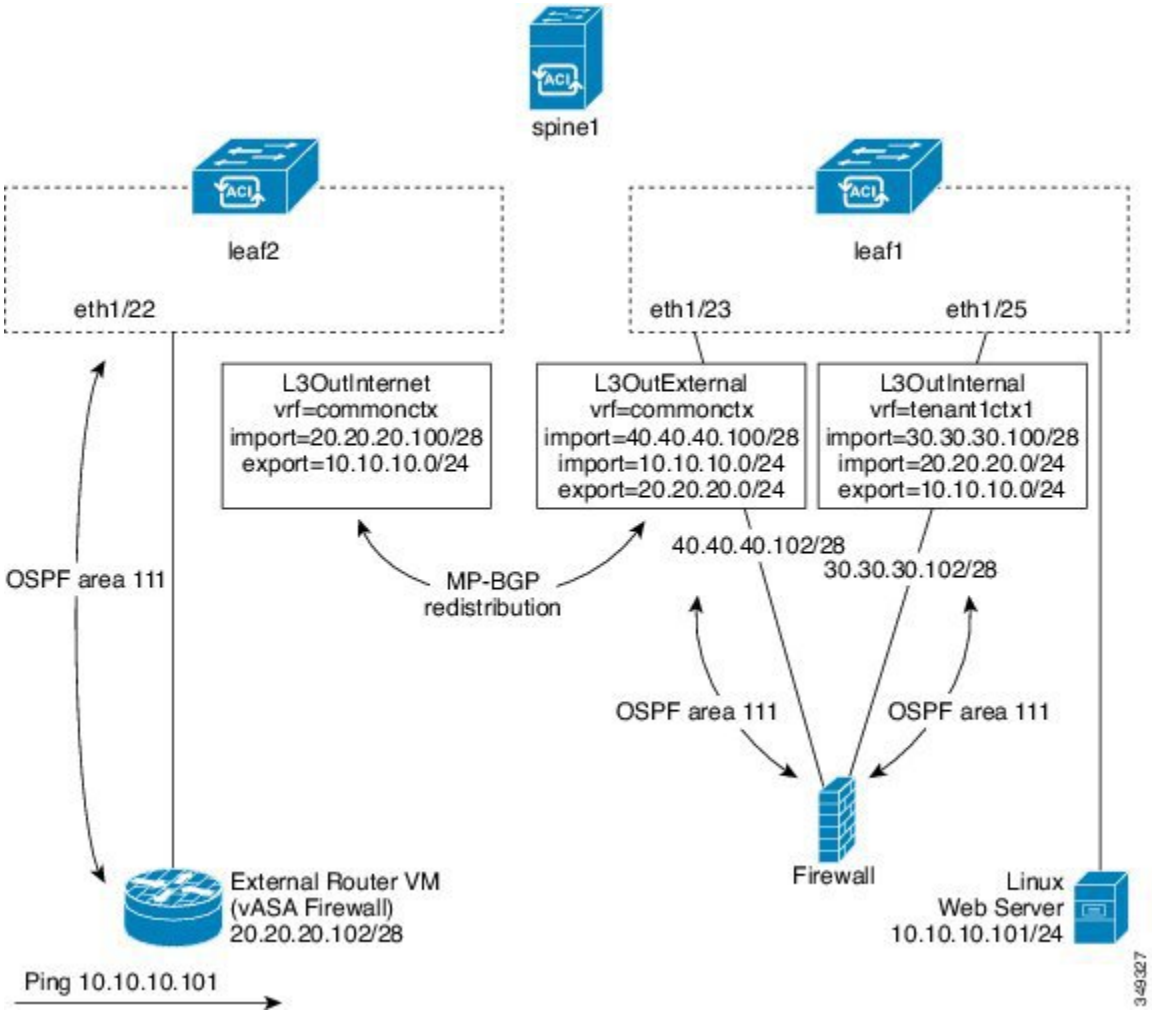
```
<vnsCDev name="ASA">
  <vnsCIf name="Gig0/0">
    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"/>
  </vnsCIf>
  <vnsCIf name="Gig0/1">
    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]"/>
  </vnsCIf>
  <vnsCMgmt name="devMgmt" host="{{asaIp}}" port="443"/>
  <vnsCCred name="username" value="admin"/>
  <vnsCCredSecret name="password" value="insieme"/>
</vnsCDev>
```

（注） ルート ピアリングを設定した場合は、vnsLIfCtx セレクタにブリッジ ドメインを設定する必要がありません。ブリッジ ドメインの関係（vnsRsLIfCtxToBD）と l3extInstP の関係（vnsRsLIfCtxToInstP）の両方を設定しようとすると、エラーになります。

ルート ピアリングのエンドツーエンド フロー

次の図に、ルート ピアリングがエンドツーエンドでどのように動作するかを示します。

図 *5* : ルート ピアリングのエンドツーエンド フロー



この図には、ルート ピアリングを使用して Linux Web サーバの IP アドレスが外部ルータにアドバタイズされる単一スパイン トポロジである 2 リーフの例が示されています。Linux Web サーバは IP アドレス 10.10.10.101 にあり、`leaf1` に接続する ESX サーバ上でホストされています。通常のブリッジ ドメイン ベースのエンドポイント グループ（EPG）が導入されており、Web サーバから発信されるトラフィックを表しています。

2 アームのルーティング可能なファイアウォールから構成され、両方のアームを `leaf1` に接続したサービス グラフを導入します。ファイアウォール デバイスには Virtual Routing and Forwarding（VRF）分割があります。つまり、ファイアウォールの各アームが異なる VRF のリーフ（コンテキスト）に接続されています。VRF 分割は、トラフィックがリーフによって短絡されるのではなく、サービス デバイスを通じて確実にルーティングされるようにするために必要です。外部トラ

フィックは、leaf2 上に導入された l3extOut（L3OutInternet）で表されています。このシナリオでは、leaf2 はファブリック ボーダー リーフと見なすことができます。L3OutInternet と Web サーバ EPG 間にコントラクトを導入できます。このコントラクトは、ファイアウォールデバイスを含むサービス グラフに関連付けられます。

Web サーバ ルートを外部にパブリッシュするには、2 つの l3extOut（L3OutExternal と L3OutInternal）をサービス デバイスを接続するリーフ ポートに導入します。その結果、Open Shortest Path First（OSPF）ピアリング セッションが両方のコンテキスト（commonctx と tenant1ctx1）のリーフとファイアウォール間で確立されます。これらの l3extOut の export 属性がボーダー リーフへのルーティング情報のアドバタイズ方法を制御します。ルートは Multiprotocol Border Gateway Protocol（MP-BGP）の再配布を使用して、ファブリック リーフ間で内部的に交換されます。

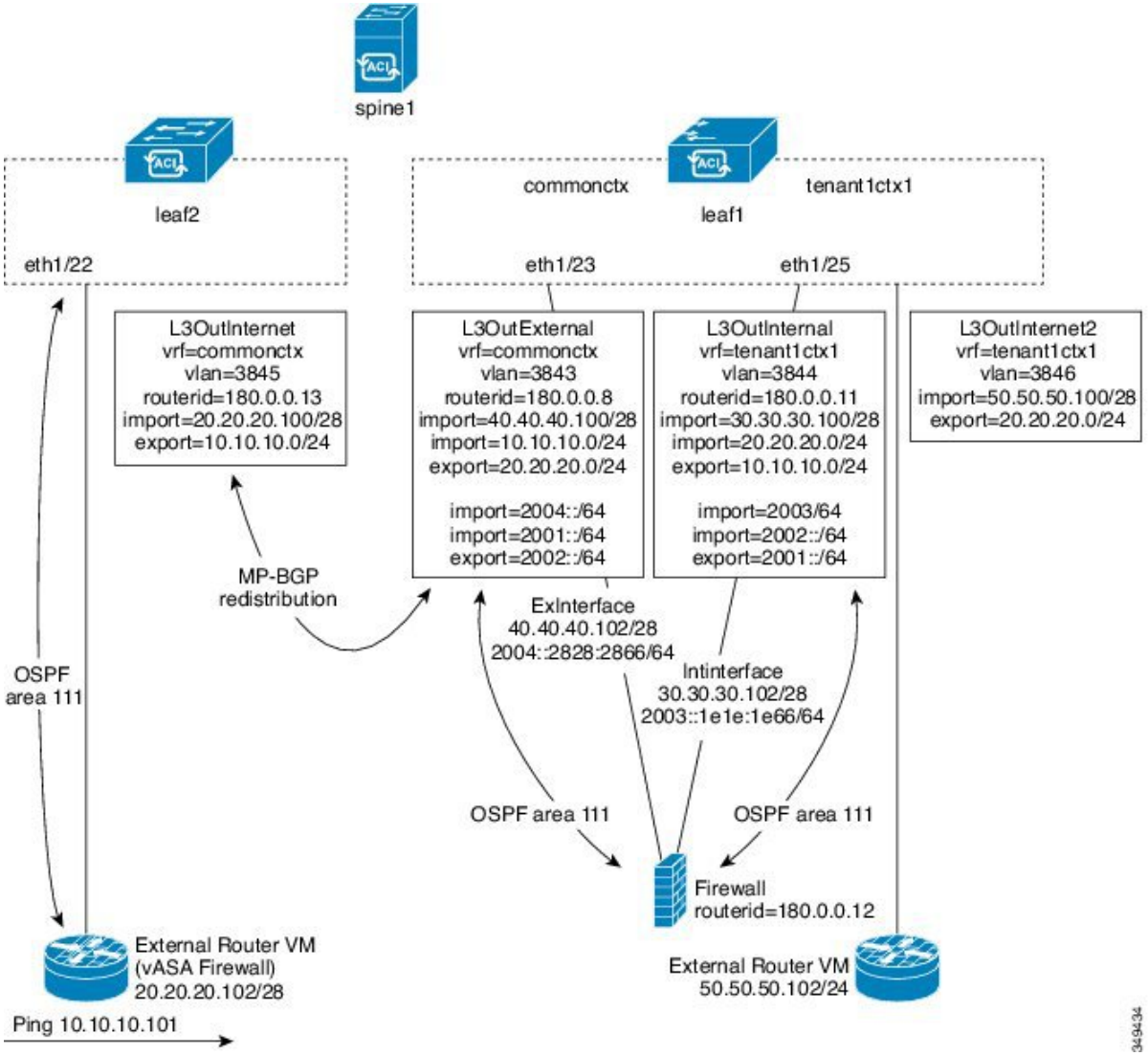
最終的に、別の OSPF セッションを使用して Web サーバ ルートが外部ルータ（IP アドレス 20.20.20.102）にアドバタイズされます。これにより、静的ルートを手動で設定することなく、外部ルータから Web サーバを ping できるようになります。

トランジットルーティングドメインとして機能するCisco Application Centric Infrastructure ファブリック

Cisco Application Centric Infrastructure（ACI）ファブリックをトランジット ルーティング ドメインとして導入すると、ACI の受渡しポイント（POD）が他の POD 間のトランジット ルーティング ドメインとして機能している場合に便利です。次の図に、2 つのボーダー リーフへの 2 つの 外部 l3extOut（L3OutInternet と L3OutInternet2）の導入を示します。これらの 3extOut 間には関連付

けられているコントラクトがあり、そのコントラクトはファイアウォールサービスデバイスを含む単一ノードのサービス グラフに適用されています。

図 *6*：トランジット ルーティング ドメインとして機能する *ACI* ファブリック



2 つの追加 `l3extOut` は、ファイアウォール デバイスの外部レッグと内部レッグに導入され、それらの間に Open Shortest Path First（OSPF）ピアリング セッションを確立します。インポート セキュリティ制御（`import-security` 属性）を適切に設定することで、ボーダー リーフへの ACI ファブリックの通過を許可するルートを制御できます。

GUI を使用したルート ピアリングの設定

ルート ピアリングを設定するには、次のタスクを実行する必要があります。

1. デバイスと Cisco Application Centric Infrastructure（ACI）ファブリック間のカプセル化 VLAN に使用するスタティック VLAN プールを作成します。

   GUI を使用したスタティック VLAN プールの作成, （48 ページ）を参照してください。

2. デバイスの場所（リーフ ノード/パス）と VLAN プールを結びつける外部ルーテッド ドメインを作成します。

   GUI を使用した外部ルーテッド ドメインの作成, （49 ページ）を参照してください。

3. ルート ピアリングで ACI のルーティング設定を指定するために使用する外部ルーテッド ネットワークを作成します。

   GUI を使用した外部ルーテッド ネットワークの作成, （50 ページ）を参照してください。

4. デバイスで使用するルータ ID を指定する新しいルータ設定を作成します。

   GUI を使用したルータ設定の作成, （52 ページ）を参照してください。

5. サービス グラフのアソシエーションを作成します。これには、外部ルーテッド ネットワーク ポリシーおよびルータ設定とデバイス選択ポリシーの関連付けが含まれます。

   GUI を使用したサービス グラフ アソシエーションの作成, （53 ページ）を参照してください。

GUI を使用したスタティック VLAN プールの作成

外部ルーテッドネットワーク設定を作成する前に、デバイスとファブリック間のカプセル化VLANに使用するスタティック VLAN プールを作成する必要があります。

ステップ 1 メニュー バーで、[Fabric] > [Access Policies] の順に選択します。

ステップ 2 [Navigation] ペインで、[Pools] > [VLAN] の順に選択します。

ステップ 3 [Work] ペインで、[Actions] > [Create VLAN Pool] の順に選択します。

ステップ 4 [Create VLAN Pool] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [Allocation Mode] オプション ボタンでは [Static Allocation] を選択します。

b) [Encap Blocks] セクションでは、[+] をクリックします。

ステップ 5 [Create Ranges] ダイアログボックスで、一意の VLAN 範囲を入力し、[OK] をクリックします。

ステップ 6 [Create VLAN Pool] ダイアログボックスで、[Submit] をクリックします。

GUI を使用した外部ルーテッド ドメインの作成

デバイスの場所（リーフ ノード/パス）とルート ピアリング用に作成するスタティック VLAN プールを結びつける外部ルーテッド ドメインを作成する必要があります。

ステップ 1 メニュー バーで、[FABRIC] > [Access Policies] の順に選択します。

ステップ 2 [Navigation] ペインで、[Switch Policies] を右クリックし、[Configure Interface, PC and VPC] を選択します。

ステップ 3 [Configure Interface, PC, and VPC] ダイアログボックスで、Application Policy Infrastructure Controller（APIC）に接続されるスイッチ ポートを設定し、次の操作を実行します。

a) スイッチ図の横にある大きい [+] アイコンをクリックし、新しいプロファイルを作成して VLAN を APIC 用に設定します。
b) [Switches] フィールドのドロップダウン リストから、APIC を接続するスイッチのチェックボックスをオンにします
c) [Switch Profile Name] フィールドに、プロファイルの名前を入力します。
d) [+] アイコンをクリックして、ポートを設定します。
e) [Interface Type] 領域で、[Individual] オプション ボタンが選択されていることを確認します。
f) [Interfaces] フィールドで、APIC が接続されるポートを入力します。
g) [Interface Selector Name] フィールドに、ポート プロファイルの名前を入力します。
h) [Interface Policy Group] フィールドで、[Create One] オプション ボタンをクリックします。
i) [Attached Device Type] ドロップダウン リストで、[External Routed Devices] を選択します。
j) [Domain] オプション ボタンでは、[Create One] オプション ボタンをクリックします。
k) [Domain Name] フィールドに、ドメイン名を入力します
l) VLAN プールを前に作成していた場合は、[VLAN] オプション ボタンとして、[Choose One] オプション ボタンをクリックします。その他の場合は、[Create One] オプション ボタンをクリックします。
既存の VLAN プールを選択する場合は、[VLAN Pool] ドロップダウン リストで、VLAN プールを選択します。

VLAN プールを作成する場合は、[VLAN Range] フィールドに VLAN 範囲を入力します。

m) [Save] をクリックし、[Save] をもう一度クリックします。
n) [Submit] をクリックします。

GUI を使用した外部ルーテッド ネットワークの作成

外部ルーテッド ネットワークで、Cisco Application Centric Infrastructure（ACI）ファブリック内にルート ピアリング用のルーティング設定を指定します。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、*[tenant_name]* > [Networking] > [External Routed Networks] を選択します。

ステップ 4 [Work] ペインで、[Actions] > [Create Routed Outside] を選択します。

ステップ 5 [Create Routed Outside] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) ダイナミック ルーティングの場合は、[BGP] チェックボックスまたは [OSPF] チェックボックスをオンにします。
Open Shortest Path Firs （OSPF ）の場合は、追加の OSPF 固有のフィールドに入力します。

b) [Private Network] ドロップダウン リストで、デバイスがルートを交換するプライベート ネットワークを選択します。

c) [External Routed Domain] ドロップダウン リストで、ルート ピアリング用に作成した外部ルーテッド ドメインを選択します。

d) [Nodes and Interfaces Protocol Profiles] セクションで、[+] をクリックします。

ステップ 6 [Create Node Profile] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [Nodes] セクションで、[+] をクリックします。

ステップ 7 [Select Node] ダイアログボックスで、下記に指定している項目を除き、必要に応じてフィールドに入力します。

a) [Node ID] ドロップダウン リストで、デバイスを接続するノード ID を選択します。

- 物理デバイスの場合は、物理デバイスをファブリックに接続するノードの ID にする必要があります。
- 仮想デバイスの場合は、仮想マシンをホストしているサーバが接続するノードの ID にする必要があります。

b) [Router ID] フィールドに、ACI ファブリックがルーティング プロトコル プロセスで使用するルータ ID を入力します。

c) ACI ファブリックとデバイス間にスタティック ルーティングを使用する場合は、[Static Routes] セクションで [+] をクリックします。それ以外の場合は、ステップ 10, （51 ページ） に進みます。

ステップ 8 [Create Static Route] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [Prefix] セクションには、静的ルートのプレフィックスを入力します。

b) [Next Hop Addresses] セクションでは、[+] をクリックします。
c) 静的ルートのネクスト ホップ IP アドレスを入力します。
d) [Update] をクリックします。

ステップ 9 [OK] をクリックします。

ステップ 10 [Select Node] ダイアログボックスで、[OK] をクリックします。

ステップ 11 ダイナミック ルーティング プロトコルとしてデバイスで BGP を使用する場合は、[BGP Peer Connectivity Profiles] セクションで、[+] をクリックします。それ以外の場合は、ステップ 14, （51 ページ） に進みます。

ステップ 12 [Create Peer Connectivity Profile] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [Peer Address] フィールドで、BGP セッションを確立するデバイスの IP アドレスであるピア アドレスを入力します。

ステップ 13 [Create Peer Connectivity Profile] ダイアログボックスで、[OK] をクリックします。

ステップ 14 [Interface Profiles] セクションで、[+] をクリックします。

ステップ 15 [Create Interface Profile] ダイアログボックスで、必要に応じてフィールドに入力します。

a) ダイナミック ルーティング プロトコルとして OSPF を使用する場合は、OSPF プロファイル情報を入力します。

ステップ 16 [Interface] セクションでは、[SVI] タブを選択します。

ステップ 17 [Interface] セクションで、[+] をクリックします。

ステップ 18 [Select SVI Interface] ダイアログボックスで、下記に指定している項目を除き、必要に応じてフィールドに入力します。

a) [Path Type] オプション ボタンでは、デバイスのファブリックへの接続方法と一致するタイプを選択します。
b) [Path] ドロップダウン リストで、デバイスをファブリックに接続するパスを選択します。

- 物理デバイスの場合は、物理デバイスをファブリックに接続するパスです。
- 仮想デバイスの場合は、仮想マシンをホストしているサーバを接続するパスです。

c) [Encap] フィールドで、カプセル化 VLAN を指定します。
d) [IP Address] フィールドで、ファブリック SVI インターフェイスで使用する IP アドレスを指定します。
e) [MTU (bytes)] フィールドで、最大伝送ユニット サイズをバイト単位で指定します。
デフォルト値は「inherit」で、ACI ではデフォルト値の「9000」を使用し、リモート デバイスでは通常はデフォルト値の「1500」を使用します。さまざまな MTU 値を指定すると、ACI とリモート デバイス間をピアリングした場合に問題を発生させる可能性があます。リモート デバイスの MTU 値を「1500」に設定した場合は、リモート デバイスの `L3Out` オブジェクトの MTU 値を「9000」に設定して ACI の MTU 値と一致させます。

ステップ 19 [OK] をクリックします。

ステップ 20 [Create Interface Profile] ダイアログボックスで、[OK] をクリックします。

ステップ 21 [Create Node Profile] ダイアログボックスで、[OK] をクリックします。

ステップ 22 [Create Routed Outside] ダイアログボックスで、[Next] をクリックします。

ステップ 23 [External EPG Networks] セクションで、[+] をクリックします。

ステップ 24 [Create External Network] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [Subnet] セクションで、[+] をクリックします。

ステップ 25 [Create Subnet] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [IP Address] フィールドに IP アドレスまたはサブネット マスクを入力します。
サブネットマスクは、従来のルーティングプロトコル設定で定義するネットワーク ステートメントと同等です。

ステップ 26 [OK] をクリックします。

ステップ 27 （任意） 必要に応じて、さらにサブネットを作成します。

ステップ 28 [Create External Network] ダイアログボックスで、[OK] をクリックします。

ステップ 29 [Create Routed Outside] ダイアログボックスで、[Finish] をクリックします。

GUI を使用したルータ設定の作成

ルーティング プロトコル設定の一部として、デバイスで使用するルータ ID を指定する必要があります。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、*[tenant_name]* > [Networking] > [External Routed Networks] > [Router Configuration] の順に選択します。

ステップ 4 [Work] ペインの [Router Configurations] テーブルで、[+] をクリックします。

ステップ 5 デバイスでルータ ID として使用する IP アドレスを入力します。

ステップ 6 [Update] をクリックします。

GUI を使用したサービス グラフ アソシエーションの作成

サービス グラフのアソシエーションを作成する必要があります。これには、外部ルーテッド ネットワーク ポリシーおよびルータ設定とデバイス選択ポリシーの関連付けが含まれます。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、*[tenant_name]* > [L4-L7 Services] > [Device Selection Policies] > *[device_selection_policy]* の順に選択します。[*device_selection_policy*] は、Cisco Application Centric Infrastructure（ACI）ファブリックでルート ピアリングを実行するデバイス選択ポリシーです。

ステップ 4 [Work] ペインの [properties] セクションにある [Router Config] ドロップダウン リストで、ルーティング ピアリング用に作成したルータ設定を選択します。

ステップ 5 [Navigation] ペインで、選択したデバイス選択ポリシーを展開し、ACI とピアリングするインターフェイスを選択します。

ステップ 6 [Work] ペインの [properties] セクションにある [Associated Network] オプション ボタンに [L3 External Network] を選択します。

ステップ 7 [L3 External Network] ドロップダウン リストで、ルート ピアリング用に作成した外部ルーテッド ネットワークを選択します。

次のように変更されます。

- 外部ルーテッド ネットワークと関連付けたインターフェイスのカプセル化 VLAN が、外部ルーテッド ネットワーク インターフェイス プロファイルの一部として設定した VLAN と一致するようにプログラミングされる
- 外部ルーテッド ネットワーク インターフェイスとルーティング プロトコル設定がルーフ スイッチにプッシュされる
- ルーティング プロトコル設定がデバイス パッケージを使用してデバイスにプッシュされる

NX-OS スタイルの CLI を使用したルート ピアリングの設定

ここでは、ルート ピアリングを設定する NX OS スタイルの CLI のコマンドの例を示します。

ステップ 1 コンフィギュレーション モードを開始します。

例：
```
apic1# configure
```

ステップ 2 テナントのコンフィギュレーション モードを開始します。

例：
```
apic1(config)# tenant 101
```

ステップ 3 サービス グラフを追加し、それをコントラクトと関連付けます。

例：
```
apic1(config-tenant)# l4l7 graph g1 contract c1
```

ステップ 4 デバイス クラスタに関連付けるノード（サービス）を追加します。

例：
```
apic1(config-graph)# service ASA_FW device-cluster-tenant 101 device-cluster ASA_FW1
```

ステップ 5 サービス機能で、コンシューマ コネクタとプロバイダー クラスタ インターフェイスを設定します。

例：
```
apic1(config-service)# connector consumer cluster-interface provider
```

ステップ 6 クラスタ インターフェイスで、サービス デバイスでのルート ピアリングで使用するレイヤ 3 Outside（l3extOut）とエンドポイント グループ（l3extInstP）を指定し、コネクタのコンフィギュレーション モードを終了します。

例：
```
apic1(config-connector)# l4l7-peer tenant 101 out l101 epg e101 redistribute bgp
apic1(config-connector)# exit
```

ステップ 7 プロバイダ コネクタとコンシューマのクラスタ インターフェイスにステップ 5 とステップ 6 を繰り返します。

例：
```
apic1(config-service)# connector provider cluster-interface consumer
apic1(config-connector)# l4l7-peer tenant 101 out l101 epg e101 redistribute bgp
apic1(config-connector)# exit
```

ステップ 8 （任意） コネクタからエンドポイント グループの関連付けを解除する場合は、**no l4l7-peer** コマンドを使用します。

例：
```
apic1(config-connector)# no l4l7-peer tenant 101 out l101 epg e101 redistribute bgp
```

ステップ 9 ルータ設定ポリシーをテナントに作成し、ピア レイヤ 4 ～ レイヤ 7 デバイスにルータ ID を指定し、コンフィギュレーション モードに戻ります。

例：
```
apic1(config)# tenant 102
apic1(config-tenant)# rtr-cfg bgp1
```

```
apic1(config-router)# router-id 1.2.3.5
apic1(config-router)# exit
```

ステップ 10 ルータ設定ポリシーを特定のサービス デバイスに関連付け、テナント コンフィギュレーション モードに戻ります。

例：

```
apic1(config-tenant)# l4l7 graph g2 contract c2 subject http
apic1(config-graph)# service ASA_FW device-cluster-tenant 102 device-cluster ASA_FW2
apic1(config-service)# rtr-cfg bgp1
apic1(config-service)# exit
apic1(config-graph)# exit
```

ステップ 11 レイヤ 3 Outside をリーフ インターフェイスおよび VRF に関連付けます。

例：

```
apic1(config-tenant)# external-l3 epg e101 l3out l101
apic1(config-tenant-l3ext-epg)# vrf member v101
apic1(config-tenant-l3ext-epg)# match ip 101.101.1.0/24
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant 101 vrf v101 l3out l101
apic1(config-leaf-vrf)# ip route 101.101.1.0/24 99.1.1.2
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# interface ethernet 1/10
apic1(config-leaf-if)# vrf member tenant 101 vrf v101 l3out l101
apic1(config-leaf-if)# vlan-domain member dom101
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# ip address 99.1.1.1/24
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

ルーティング プロトコル（BGP、OSPF）やルート マップなど、名前付きのモードを使用したレイヤ 3 外部接続（レイヤ 3 Outside）の詳細な設定については、『*Cisco APIC NX-OS Style CLI Command Reference*』を参照してください。

（注） CLI での外部レイヤ 3 設定は、2 つのモード（基本モードと名前付きモード）で使用できます。特定のテナントまたは VRF では、すべての外部レイヤ 3 設定にこれらのモードの 1 つのみを使用します。ルート ピアリングは名前付きモードでのみサポートされています。

ルート ピアリングのトラブルシューティング

Cisco Application Centric Infrastructure（ACI）ファブリックにルート ピアリングまたはデータ トラフィックの問題がある場合に、その問題をトラブルシーティングするために ACI ファブリック リーフ上で実行できるコマンドがいくつかあります。

次の表に、ファブリック リーフのスイッチ シェルで実行できるトラブルシューティング コマンドを示します。

コマンド	説明
show ip route vrf all	動的に取得したルートを含む特定のコンテキストのすべてのルートを表示します。
show ip ospf neighbor vrf all	隣接デバイスとの Open Shortest Path First（OSPF）ピアリング セッションを表示します。
show ip ospf vrf all	各コンテキスト内のランタイム OSPF 設定を表示します。
show ip ospf traffic vrf all	Virtual Routing and Forwarding（VRF）の各コンテキストの OSPF トラフィックを確認します。
show system internal policymgr stats	特定のリーフのコントラクト フィルタ ルールを表示し、ルールのパケット ヒット カウントを確認します。

次の表に、**vsh_lc** シェルで実行できるトラブルシューティング コマンドを示します。

コマンド	説明
show system internal aclqos prefix	特定のリーフの IPv4 プレフィックス アソシエーション ルールとルールのトラフィック ヒット カウントを確認します。

シェル コマンドに加えて、トラブルシューティングに役立つ次の点を確認できます。

- デバイスの健全性カウント
- 特定のテナントの下のすべてのエラーと `NwIssues`

CLI を使用したリーフ スイッチのルート ピアリング機能の確認

ファブリック リーフ上でスイッチ シェル コマンドを使用して、リーフ スイッチ設定とルート ピアリング機能を確認することができます。

ステップ 1 デバイスが接続されているファブリック リーフ スイッチで、SVI インターフェイスが設定されていることを確認します。

```
fab2-leaf3# show ip interface vrf user1:global
IP Interface Status for VRF "user1:global"
vlan30, Interface status: protocol-up/link-up/admin-up, iod: 134,
  IP address: 1.1.1.1, IP subnet: 1.1.1.0/30
```

```
  IP broadcast address: 255.255.255.255
  IP primary address route-preference: 1, tag: 0
lo3, Interface status: protocol-up/link-up/admin-up, iod: 133,
  IP address: 10.10.10.1, IP subnet: 10.10.10.1/32
  IP broadcast address: 255.255.255.255
  IP primary address route-preference: 1, tag: 0

fab2-leaf3#
```

インターフェイス vlan30 には SVI インターフェイス設定が含まれており、インターフェイス lo3 には外部ルーテッド ネットワーク設定に指定されているルータ ID が含まれています。

ステップ 2 ファブリック リーフ スイッチの Open Shortest Path First（OSPF）の設定を確認します。

```
fab2-leaf3# show ip ospf vrf user1:global

 Routing Process default with ID 10.10.10.1 VRF user1:global
 Stateful High Availability enabled
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 Table-map using route-map exp-ctx-2949120-deny-external-tag
 Redistributing External Routes from
   static route-map exp-ctx-st-2949120
   bgp route-map exp-ctx-proto-2949120
   eigrp route-map exp-ctx-proto-2949120
 Maximum number of non self-generated LSA allowed 100000
    (feature configured but inactive)
    Current number of non self-generated LSA 1
    Threshold for warning message 75%
    Ignore-time 5 minutes, reset-time 10 minutes
    Ignore-count allowed 5, current ignore-count 0
 Administrative distance 110
 Reference Bandwidth is 40000 Mbps
 SPF throttling delay time of 200.000 msecs,
   SPF throttling hold time of 1000.000 msecs,
   SPF throttling maximum wait time of 5000.000 msecs
 LSA throttling start time of 0.000 msecs,
   LSA throttling hold interval of 5000.000 msecs,
   LSA throttling maximum wait time of 5000.000 msecs
 Minimum LSA arrival 1000.000 msec
 LSA group pacing timer 10 secs
 Maximum paths to destination 8
 Number of external LSAs 0, checksum sum 0x0
 Number of opaque AS LSAs 0, checksum sum 0x0
 Number of areas is 1, 1 normal, 0 stub, 0 nssa
 Number of active areas is 1, 1 normal, 0 stub, 0 nssa
   Area (0.0.0.200)
        Area has existed for 00:17:55
        Interfaces in this area: 1 Active interfaces: 1
        Passive interfaces: 0  Loopback interfaces: 0
        SPF calculation has run 4 times
         Last SPF ran for 0.000273s
        Area ranges are
        Area-filter in 'exp-ctx-proto-2949120'
```

```
        Number of LSAs: 3, checksum sum 0x0
fab2-leaf3#
```

ステップ 3 ファブリック リーフ スイッチの OSPF ネイバーの関係を確認します。

```
fab2-leaf3# show ip ospf neighbors vrf user1:global
 OSPF Process ID default VRF user1:global
 Total number of neighbors: 1
 Neighbor ID     Pri State            Up Time  Address         Interface
 10.10.10.2        1 FULL/BDR         00:03:02 1.1.1.2         Vlan30
fab2-leaf3#
```

ステップ 4 ルートがファブリック リーフ スイッチによって取得されることを確認します。

```
fab2-leaf3# show ip route vrf user1:global
IP Route Table for VRF "user1:global"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

1.1.1.0/30, ubest/mbest: 1/0, attached, direct
    *via 1.1.1.1, vlan30, [1/0], 00:26:50, direct
1.1.1.1/32, ubest/mbest: 1/0, attached
    *via 1.1.1.1, vlan30, [1/0], 00:26:50, local, local
2.2.2.0/24, ubest/mbest: 1/0
    *via 1.1.1.2, vlan30, [110/20], 00:06:19, ospf-default, type-2
10.10.10.1/32, ubest/mbest: 2/0, attached, direct
    *via 10.10.10.1, lo3, [1/0], 00:26:50, local, local
    *via 10.10.10.1, lo3, [1/0], 00:26:50, direct
10.122.254.0/24, ubest/mbest: 1/0
    *via 1.1.1.2, vlan30, [110/20], 00:06:19, ospf-default, type-2
fab2-leaf3#
```

ステップ 5 OSPF がデバイス（この例では Cisco ASAv）に設定されていることを確認します。

```
ciscoasa# show running-config
: Saved
:
: Serial Number: 9AGRM5NBEXG
: Hardware:   ASAv, 2048 MB RAM, CPU Xeon 5500 series 2133 MHz
:
ASA Version 9.3(1)
!
hostname ciscoasa
enable password 8Ry2YjIyt7RRXU24 encrypted
names
!
interface GigabitEthernet0/0
 nameif internalIf
 security-level 100
 ip address 2.2.2.1 255.255.255.0
!
interface GigabitEthernet0/1
 nameif externalIf
 security-level 50
 ip address 1.1.1.2 255.255.255.252
!
```

```
<<..>>
router ospf 1
 router-id 10.10.10.2
 network 1.1.1.0 255.255.255.252 area 200
 area 200
 log-adj-changes
 redistribute connected
 redistribute static
!
```

第 8 章

Direct Server Return の設定

Direct Server Return について

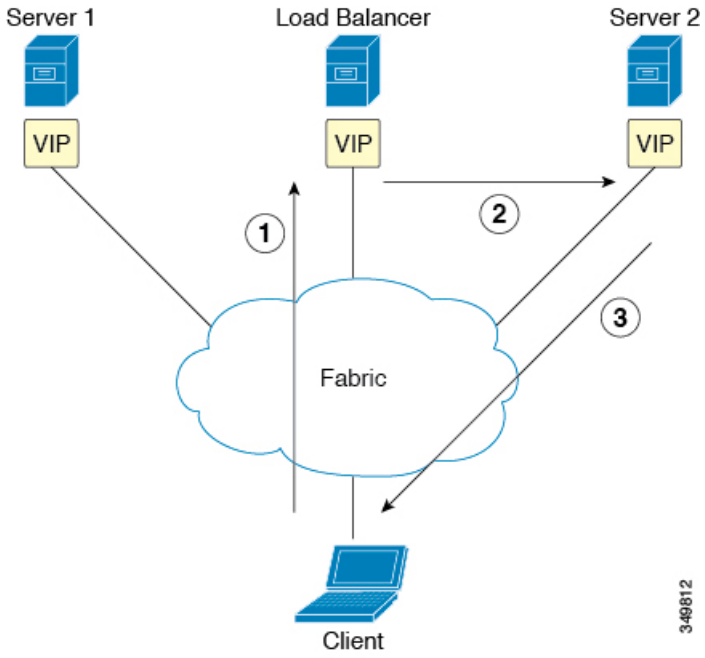
Didrect Server Return 機能により、サーバはロード バランサを通過する必要なく、クライアントに直接応答できます。これにより、サーバからクライアントへのパスにおけるボトルネックが解消されます。従来のロードバランサの導入では、ロードバランサは、クライアントとサーバとの通信のパス（クライアントからサーバへの要求パスとサーバからクライアントへの応答パスの両方）に存在します。クライアントからサーバ方向の要求内のデータの量は比較的少ないものの、サーバからクライアントへの応答トラフィックはかなり大きく、クライアントからサーバへの要求データの約 10 倍になります。この大量の応答トラフィックがあるパス内のロード バランサがボトルネックになり、通信に悪影響を及ぼします。

Direct Server Return の導入では、ロード バランサとサーバとで仮想 IP アドレスが共有されます。クライアントは、ロードバランサに到達することを目的とした仮想IPアドレスに常に要求を送信し、また、サーバからクライアントへの直接応答ではこの仮想IPアドレスを送信元アドレスとして使用します。IP送信元アドレスのデータパスの取得が有効になっているCisco Application Centric Infrastructure（ACI）は、サーバからクライアントへのトラフィックの仮想IPアドレスを取得すると問題を引き起こし、クライアントからロードバランサへの要求トラフィックを途絶させることになります。Direct Server Returnの導入を適切に動作させるには、ACIファブリックが通信中のエンドポイント間の要求と応答のトラフィックを目的の宛先に正しく配信されるようにする必要があります。これには、リーフ上でのデータパスIPアドレスの取得を、クライアントからロードバ

ランサへのトラフィック、ロードバランサからサーバへのトラフィック、およびサーバからクライアントへのトラフィックに割り込みを生じさせないように制御することが必要です。

次の図に、Direct Server Return の導入のデータ パスを示します。

図 7 : *Direct Server Return* の全体的なフロー



1. ロードバランサとすべてのバックエンドサーバが仮想 IP アドレスで設定されています。ロードバランサのみが、この仮想 IP アドレス宛の Address Resolution Protocol（ARP）要求に応答します。クライアント要求のロードバランシング後に、ロードバランサはパケット内の宛先 MAC アドレスを書き換えて、その MAC アドレスをバックエンドサーバの 1 つに転送します。
2. 仮想 IP アドレスはバックエンドサーバ上に設定されますが、ARP が無効になっているため、この仮想 IP アドレス宛の ARP 要求にバックエンドサーバは応答できません。
3. サーバはリターン トラフィックをクライアントに直接送信してロードバランサをバイパスします。

レイヤ 2 の Direct Server Return

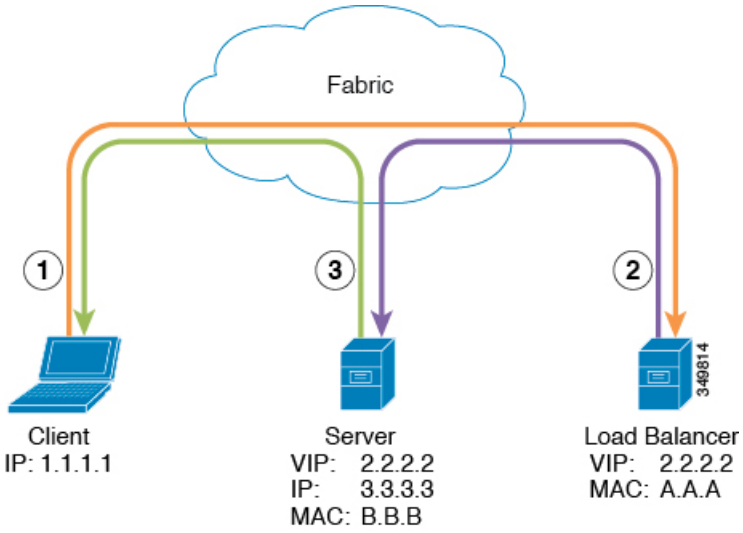
レイヤ 2 の Direct Server Return は一般的な導入または従来型の導入であり、ダイレクト ルーティング、SwitchBack、または nPath とも呼ばれます。この導入では、ロードバランサとサーバで仮想 IP アドレスが共有されます。ロードバランサとサーバはレイヤ 2 隣接である必要があります。レイヤ 2 の Direct Server Return の導入には、次の制限があります。

- サーバ配置の柔軟性が失われる

- クライアントの仮想 IP アドレス要求への Address Resolution Protocol（ARP）応答を抑制するために、追加のサーバ設定が必要になる
- ポート選択はレイヤ 3 で行われ、プロトコルに依存する。ポート選択はレイヤ 2（サーバ通信に対するロード バランサ）で行われない

レイヤ 2 の Direct Server Return の導入には、次のトラフィック フローがあります。

図 *8*：レイヤ *2* の *Direct Server Return* のトラフィック フロー



1 クライアントからロード バランサへ

送信元 IP アドレス	1.1.1.1
宛先 IP アドレス	2.2.2.2
宛先 MAC アドレス	A.A.A

2 ロード バランサからサーバへ

送信元 IP アドレス	1.1.1.1
宛先 IP アドレス	2.2.2.2
宛先 MAC アドレス	B.B.B

3 サーバからクライアントへ

送信元 IP アドレス	2.2.2.2

宛先 IP アドレス	1.1.1.1
宛先 MAC アドレス	デフォルト ゲートウェイの MAC アドレス

Cisco Application Centric Infrastructure でのレイヤ 2 Direct Server Return の導入について

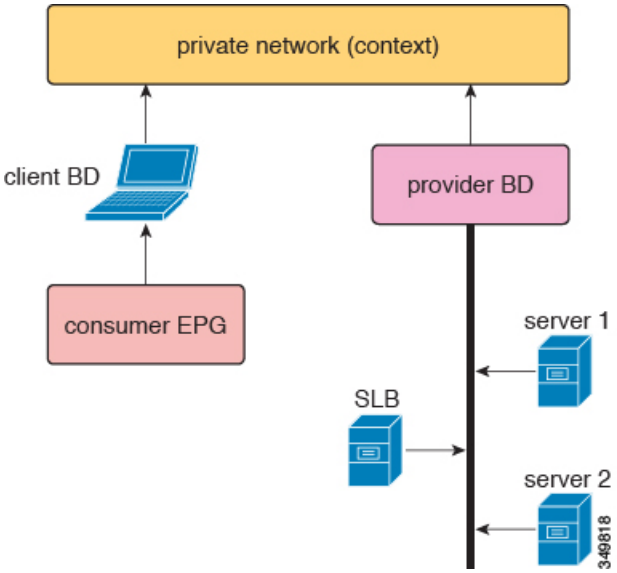
次に、Cisco Application Centric Infrastructure（ACI）でのレイヤ 2 Direct Server Return の導入に適用される情報を示します。

- 仮想 IP アドレス（2.2.2.2）は ACI ファブリック内を移動する
  - 同じ送信元仮想 IP アドレス（2.2.2.2）を持つロード バランサからサーバおよびサーバからクライアントへのトラフィック
  - サーバからクライアントへのトラフィックはルーティングされ、トラフィックはファブリック内のゲートウェイ MAC アドレス宛になる
  - サーバからの送信元 IP アドレスのデータ パスの取得はファブリック内の仮想 IP アドレスに移動する
- 異なる送信元から表示されるクライアント IP アドレス（1.1.1.1）についての問題はない
  - クライアント IP アドレスはファブリック内のクライアントとロード バランサの両方からの送信元 IP アドレスとして表示される
  - ロード バランサとサーバは、レイヤ 2 隣接であり、ロード バランサからサーバへのトラフィックはレイヤ 2 に転送される
  - ファブリック内のレイヤ 2 転送トラフィックからのデータパス IP アドレスの取得はない
  - クライアント IP アドレスがファブリック内のロード バランサからの送信元 IP アドレスとして表示された場合も、クライアント IP アドレスは取得されない

サポートされている Direct Server Return の設定

次の図に、サポートされている Direct Server Return の設定を示します。

図 *9*：サポートされている *Direct Server Return* の設定



サポートされている設定に次の情報が適用されます。

- サーバ ロード バランサとサーバは同じサブネットとブリッジ ドメインにある
- サーバ ロード バランサは 1 ARM モードで動作する必要があり、サーバ ロード バランサの内部レッグと外部レッグは同じブリッジ ドメインを指している必要がある
- コンシューマ エンドポイント グループとプロバイダー エンドポイント グループは、同じプライベート ネットワークの下にある必要がある。共有サービス設定はサポートされていない

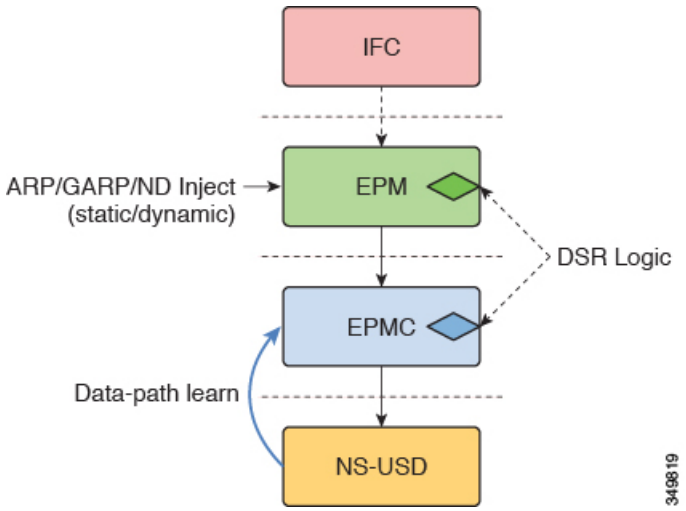
Direct Server Return のアーキテクチャ

ロード バランサとサーバが共有する仮想 IP アドレスをテナントの Virtual Routing and Forwarding（VRF）のファブリック内で静的に設定し、レイヤ 2 Direct Server Return を有効にすることができます。仮想 IP アドレスを静的にすることで、仮想 IP アドレスのデータ パスの取得が阻止されます。スイッチ側のエンドポイント マネージャは、具象モデルの {VRF、VIP、S クラス} タプル形式のポリシー エンジンからの静的設定を処理します。エンドポイント マネージャ（EPM）、エンドポイント マネージャ クライアント（EPMC）および転送シリコンにアーキテクチャは変更されていませんが、一方でこれらのレイヤはすべて変更されており、仮想 IP アドレスの静的設定が許可・維持されています。このアーキテクチャでは、仮想 IPv4 と仮想 IPv6 の両方に対して静的設定が可能です。アドレス解析および初期セットアップ以外は、アーキテクチャ上および設計全体にわたって、仮想 IPv4 と仮想 IPv6 の両方が同じコード パスで処理されます。

Direct Server Return の設計フローと設定フローは EPM/EPMC/USD のフローの一部です。このためのフローは、ノースバウンドからサウスバウンド、つまり、[policy engine] > [EPM] > [EPMC] > [forwarding silicon]の場合にのみ存在します。この場合の転送シリコンは North Star です。これは、この目的に対応しているのは North Star のローカル ステーション テーブルに限られることが理由です。同じエンドポイントの作成/変更/削除フローを、追加の静的 IP アドレス エンドポイント フラグと共に使用します。

新しい IP アドレス エンドポイントのすべての取得要求は、静的仮想 IP アドレス エンドポイント チェックを受けます。取得/処理要求がすでに存在する {VRF、VIP、S クラス} タプルに対するものである場合は、Direct Server Return の前処理コードによって変更前処理が行われ、適切なフラグを使用した一般的なエンドポイント処理に戻されます。

図 *10*：スイッチ側の処理



次のリストに、Direct Server Return の設計ポイントに関する概要を示します。

- すべての仮想 IPv4 および IPv6 の追加/変更/削除設定は、エンドポイント マネージャによって処理される
  - Direct Server Return は、プレフィクスではなく、完全な仮想 IP アドレス（/32、/128）を使用する
  - アドレス ファミリおよびアドレスの最上位レベルのセットアップ以外、コード パスは Direct Server Return 用にマージされる
- EPM と EPMC は完全な（/32 または /128）仮想 IP アドレスを North Star ローカル ステーション テーブルの送信元アドレスにインストールする
  - キー/データは、設定された 3 タプル {VRF、VIP、S クラス} 情報から取得する
  - ローカル ステーション テーブルの送信元アドレスに「静的」としてエントリが挿入される

- EPM は ARP/GARP/ND IP-MAC バインディングと MAC の取得を通じてロード バランサのエンドポイントを検出できる
- EPM と EPMC は、{VRF、VIP} タプルの疑似 North Star データパスの取得を阻止する
- EPM と EPMC では、取得したエントリの S クラスがポリシー エンジンで設定した {VRF、VIP、S クラス} タプルと一致しなければ、IP-MAC バインディング アソシエーションを許可しない。これは、ARP/GARP/ND パスとデータパス取得パスの両方に適用される
- EPM は、COOP へのロード バランサの（ARP/GARP/ND を通じた）検出伝播を代替しない
- エントリの S クラスが一致しない場合、EPM と EPMC は既存の取得済みエントリ（ARP/GARP/ND）を設定時にクリーンアップする
- 仮想 IP アドレスを設定すると、EPM と EPMC は、データパスの取得を通じて作成された同じ {VRF、VIP} タプルの既存のエントリを常にクリーンアップする
- ARP/ND/MAC エージングはこれらの変更に対応していないが、EPM と EPMC が、{VRF、VIP} タプルの設定が削除されない限り、ローカル ステーション テーブルに維持されている静的エントリは削除されない
- この機能を実装する際に、既存のエントリを削除するのではなく、同じエントリ設定をポリシー エンジンから取得する場合、ARP/GARP/ND を通じて取得した {VRF、VIP} タプルを保持するアプローチを取る。これは、既存のエントリの S クラスが設定されたエントリの S クラスと同じである場合に限る。このアプローチにより、エントリの削除が原因で発生するファブリック全体にわたる大規模な変動を回避する
- S クラスとエントリに関連するその他の情報は IP アドレス情報の一部として保持される。つまり、情報はエンドポイント レベルではなく、エンドポイントの IP アドレスレベルで保持される
- {BD、仮想 IP のプレフィックス、S クラス} タプルとポリシー エンジンで設定した {VRF、VIP、S クラス} タプル間に重複がある場合は、{VRF、仮想 IP、S クラス} タプルが優先される

静的なサービス導入のための Direct Server Return の XML POST の例

次に、Direct Server Return の静的なサービス導入の例を示します。

```
<fvAp name="dev">
      <fvAEPg name="loadbalancer">
        <fvRsDomAtt tDn="uni/phys-{{tenantName}}"/>
        <fvRsBd tnFvBDName="lab"/>
        <fvVip addr="121.0.0.{{net}}" />
        <fvRsPathAtt tDn="topology/pod-1/paths-104/pathep-[eth1/1]" encap="vlan-33" />
        <fvRsProv tnVzBrCPName="loadBalancer"/>
        <fvRsCons tnVzBrCPName="webServer"/>
      </fvAEPg>
        <fvAEPg name="webServer">
          <fvRsDomAtt tDn="uni/phys-{{tenantName}}"/>
            <fvRsBd tnFvBDName="lab"/>
          <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/1]" encap="vlan-34"/>
```

```
            <fvRsProv tnVzBrCPName="webServer"/>
          </fvAEPg>
        <fvAEPg name="client">
          <fvRsDomAtt tDn="uni/phys-{{tenantName}}"/>
          <fvRsBd tnFvBDName="lab"/>
          <fvRsPathAtt tDn="topology/pod-1/paths-103/pathep-[eth1/4]" encap="vlan-1114"/>
          <fvRsCons tnVzBrCPName="loadBalancer"/>
        </fvAEPg>
</fvAp>
```

Direct Server Return 設定は、Web サーバに対する `toEPG` コントラクトが存在するすべての Top-of-Rack 型スイッチ（ToR）にダウンロードされます。この例では、Direct Server Return の仮想 IP アドレス設定が「paths-101」という ToR にダウンロードされます。ノード ID 104 で示された ToR 上にダウンロードされた仮想 IP アドレスのスタティック設定は表示されません。

Direct Server Return 設定は、`toEPG` コントラクトがなくても発行できます。コントラクトが作成されると、Direct Server Return の仮想 IP アドレスがすでに設定されている場合はそのアドレスが Web サーバ `toEPG` コントラクトが存在する ToR に自動的にダウンロードされます。

静的なサービス導入のための Direct Server Return

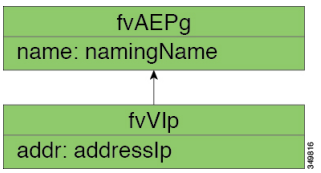
静的なサービス導入モードでは、適切なアプリケーションエンドポイントグループとコントラクトをホップごとに作成することによって、サービス フローを設定します。

静的なサービス導入の論理モデル用の Direct Server Return

アプリケーション エンドポイント グループ（`fvAEPg`）の下に `fvVip` オブジェクトを使用することによって、ロード バランサが使用する仮想 IP アドレスを設定できます。

次の図に、静的なサービス導入の論理モデルを示します。

図 *11*：静的なサービス導入の論理モデル



サービス グラフを挿入するための Direct Server Return

Cisco Application Centric Infrastructure（ACI）は、ベンダー パッケージとサービス グラフを使用してサービスの挿入を自動化します。このモードでは、内部と外部のエンドポイント グループなど、サービスデバイスのレッグに対して作成されるエンドポイントグループが、オペレータによる設定を必要とせずに、ACI によって作成されます。

サービス グラフの挿入では、次の XML POST の例に示すように、サービス デバイスの適切な論理インターフェイス コンテキストの下に仮想 IP アドレスを設定する必要があります。

```
<vnsLDevCtx ctrctNameOrLbl="webCtrct"
                        graphNameOrLbl="G1"
                        nodeNameOrLbl="SLB">

     <vnsRsLDevCtxToLDev tDn="uni/tn-coke/lDevVip-InsiemeCluster"/>

     <vnsLIfCtx connNameOrLbl="inside">
         <vnsRsLIfCtxToBD tDn="uni/tn-coke/BD-cokeBD1"/>
         <vnsRsLIfCtxToLIf tDn="uni/tn-coke/lDevVip-InsiemeCluster/lIf-inside"/>
     </vnsLIfCtx>

     <vnsLIfCtx connNameOrLbl="outside">
         <vnsRsLIfCtxToBD tDn="uni/tn-coke/BD-cokeBD1"/>
         <vnsRsLIfCtxToLIf tDn="uni/tn-coke/lDevVip-InsiemeCluster/lIf-outside"/>
         <vnsSvcVip addr="9.9.9.9" />
         <vnsSvcVip addr="11.11.11.11" />
     </vnsLIfCtx>
 </vnsLDevCtx>
```

この要求の例では、2 つの仮想 IP アドレス（9.9.9.9 と 11.11.11.11）をサーバ ロード バランサの外部レッグ上に設定します。仮想 IP アドレスの定義は、静的な Direct Server Return 設定と同様に、エンドポイント グループの下ではなく、`LIfCtx` の下になります。これは、静的サービスの導入の場合とは異なり、サービス グラフの場合は、オペレータにデバイス レッグのエンドポイント グループへの直接アクセス権がないためです。

Direct Server Return 共有レイヤ 4 ～ レイヤ 7 サービスの設定

サービス デバイスを共通のテナントまたは管理テナントに設定した場合、暗黙モデルには若干の違いがあります。`vnsEPpInfo` の代わりに、サービス仮想 IP アドレスの更新管理対象オブジェクトが `vnsREPpInfo` の子として作成されます。1 つの `vnsSvcEpgCont`の 管理対象オブジェクトが `vnsRsEPpInfo` ごとに作成されて複数のテナント間で共有 `SvcVip` を追跡します。

Direct Server Return 用の Citrix サーバ ロード バランサの設定

次に、Direct Server Return 用に Citrix サーバ ロード バランサを設定する方法の概要を示した手順を説明します。

ステップ 1 バックエンド サーバがパケットを受け入れるようにバックエンド サーバのループバックに仮想 IP アドレスを設定します。

ステップ 2 バックエンド サーバの仮想 IP アドレスに対する Address Resolution Protocol（ARP）応答を無効にします。

ステップ 3 必要に応じて、ロード バランシング仮想サーバにバインドされたサービスのプロキシ ポートを無効にします。プロキシ ポートはデフォルトで無効になっています。

ステップ 4 ロード バランシング仮想サーバの m パラメータを「MAC」に設定します。

ステップ 5 グローバルか、またはサービスごとに USIP モードを有効にします。

ステップ 6 「L3」モード、「USNIP」モード、および「MBF」モードを有効にします。

ステップ 7 バックエンド サーバのルートを直接インターネットに到達できるように設定します。

Direct Server Return 用の Linux サーバの設定

次に、Direct Server Return 用に Linux サーバを設定する方法の概要を示した手順を説明します。

ステップ 1 次のコンテンツを使用し、Centos 内に `/etc/sysconfig/network-scripts/ifcfg-lo` ファイルを作成して、ループバック インターフェイス上に仮想 IP アドレスを設定します。

```
DEVICE=lo:1
IPADDRESS=10.10.10.99
NETMASK=255.255.255.255
NETWORK=10.10.10.99
BROADCAST=10.10.10.99
ONBOOT=yes
NAME=loopback
```

この例では、10.10.10.99 が仮想 IP アドレスです。

ステップ 2 クライアント要求への応答に使用するサーバ インターフェイスの arp_ignore と arp_announce の値を設定します。

```
echo 1 > /proc/sys/net/ipv4/conf/eth1/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/eth1/arp_announce
```

この例では、`eth1` がクライアント要求への応答に使用するサーバ インターフェイスです。

ARP の設定の詳細については、次の Linux 仮想サーバの Wiki ページを参照してください。
http://kb.linuxvirtualserver.org/wiki/Using_arp_announce/arp_ignore_to_disable_ARP

第 9 章

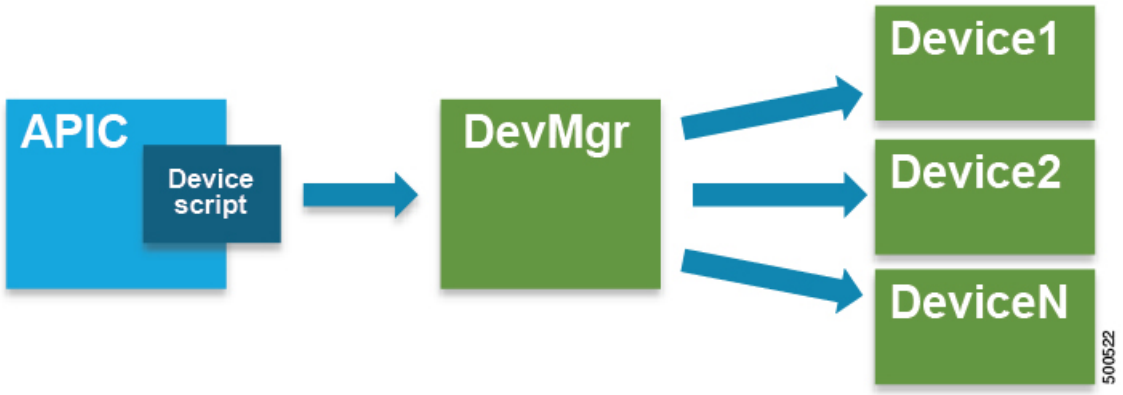
デバイスおよびシャーシ マネージャの設定

デバイス マネージャとシャーシ マネージャについて

デバイス マネージャのみで、Cisco Application Centric Infrastructure（ACI）ファブリック内の一連のクラスタを設定できます。管理状態または動作状態はデバイスのネイティブの GUI に表示されます。デバイス マネージャは、個々のデバイスの設定を処理することで、Application Policy Infrastructure Controller（APIC）での設定を容易にします。デバイス マネージャにテンプレートを作成してから、インスタンス固有の値を APIC からデバイス マネージャに挿入しますが、必要な値はごくわずかです。

次の図に、クラスタ内で複数のデバイスを制御するデバイス マネージャを示します。

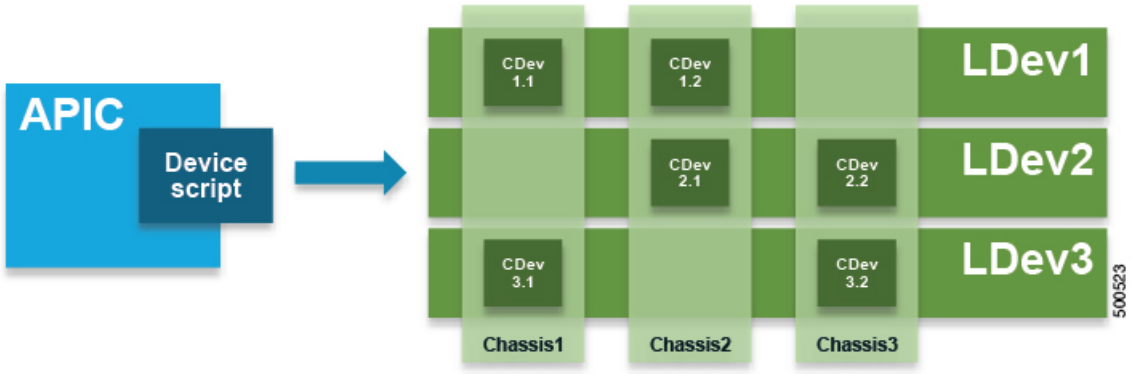
図 *12*：デバイス マネージャでのデバイスの制御



シャーシマネージャは、処理リソースの物理または仮想「コンテナ」です。シャーシマネージャは CDev オブジェクトとして表される、いくつかの仮想サービスデバイスをサポートします。シャーシマネージャがネットワーキングを処理し、CDev がプロセスを処理します。シャーシマネージャによって、仮想処理ノードのオンデマンド作成が可能になります。仮想デバイスでは、サービス（特に VLAN）の一部を、仮想マシンではなく、シャーシに適用する必要があります。これを実現するには、シャーシ管理 IP アドレスとクレデンシャルをコールアウトに含める必要があります。

次の図に、処理リソースのコンテナとして機能するシャーシ マネージャを示します。

図 *13*：デバイス マネージャでのデバイスの制御



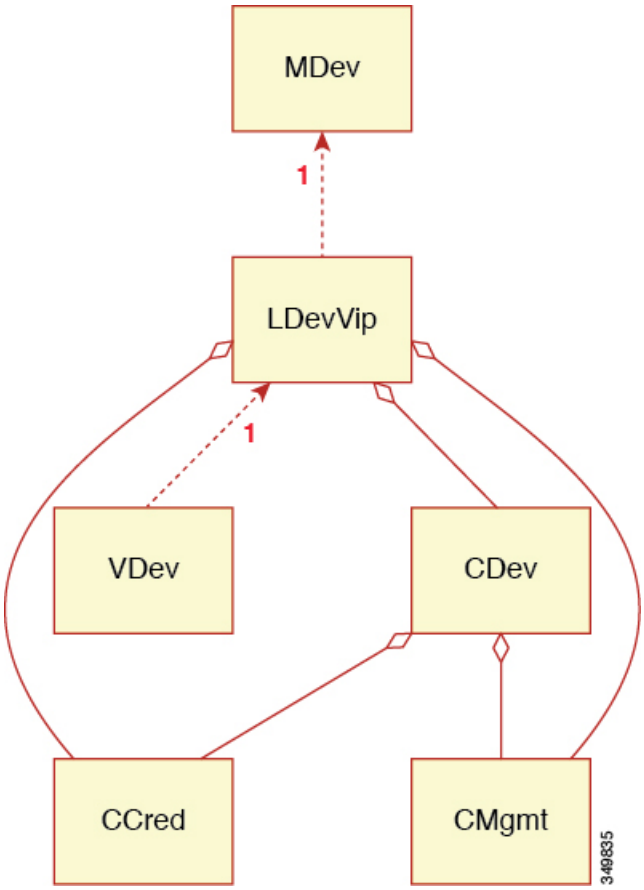
デバイス マネージャまたはシャーシ マネージャを使用せず、サービス デバイスのモデルに次の主要な管理対象オブジェクトを含めます。

- MDev：デバイス タイプ（ベンダー、モデル、バージョン）を表します。
- LDevVIP：クラスタ、つまりハイ アベイラビリティ（HA）を実現するために同一に設定された一連のデバイスを表します。デバイスにアクセスするための CMgmt と CCred が含まれます。

- CDev：物理または仮想のいずれかのクラスタのメンバーを表します。デバイスにアクセスするための CMgmt と CCred が含まれます。
- VDev：サーバ上の仮想マシンと同様のクラスタのコンテキストを表します。

次の図に、CMgmt（管理接続）と CCred（クレデンシャル）が含まれた、主要な管理対象オブジェクトのモデルを示します。

図 *14*：デバイス マネージャまたはシャーシ マネージャを含まない管理対象オブジェクト モデル



CMgmt（ホスト＋ポート）と CCred（ユーザ名＋パスワード）により、スクリプトでデバイスとクラスタにアクセスできます。

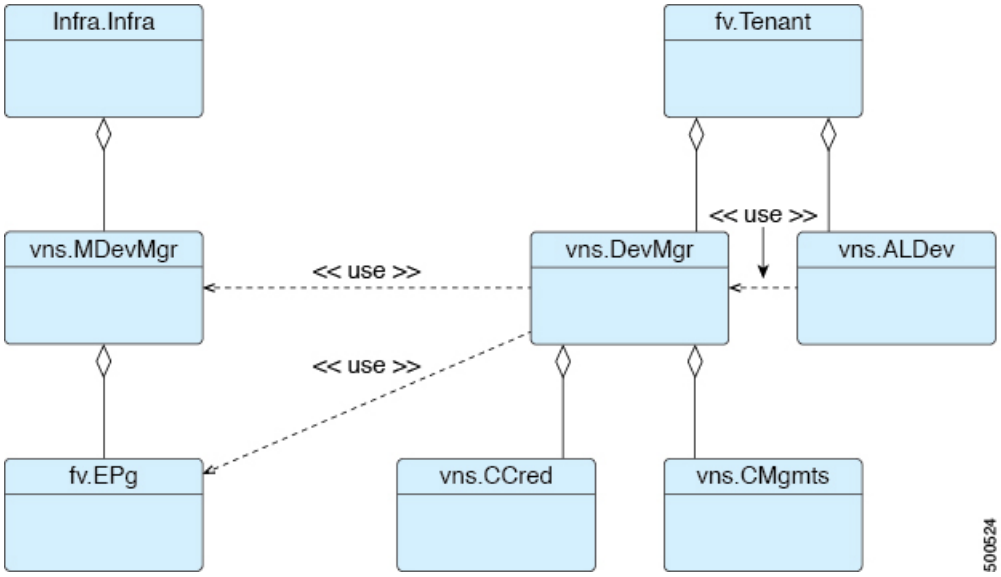
デバイスマネージャとシャーシマネージャは、集中管理ステーションからのクラスタとデバイスの設定を制御できるようにします。シャーシは並列階層を MDev オブジェクトと ALDev オブジェクトに追加し、特定のシャーシに属しているというタグを CDev オブジェクトに付けることができます。次の管理対象オブジェクトがモデルに追加され、デバイスおよびシャーシ マネージャの概念をサポートします。

- MDevMgr：デバイス マネージャのタイプを表します。MDevMgr は、同じベンダーの通常は異なる製品である一連の異なる MDev を管理できます。

- `DevMgr`：デバイス マネージャを表します。マネージャにアクセスするには、含まれている `CMgmt` と `CCred` の管理対象オブジェクトを使用します。各クラスタは 1 つの `DevMgr` のみと関連付けることができます。
- `MChassis`：シャーシのタイプを表します。通常、この管理対象デバイスはパッケージに含まれています。
- `Chassis`：シャーシ インスタンスを表します。これには、`CMgmt` と `CCred`[Secret] の管理対象オブジェクトが含まれており、シャーシへの接続を提供します。

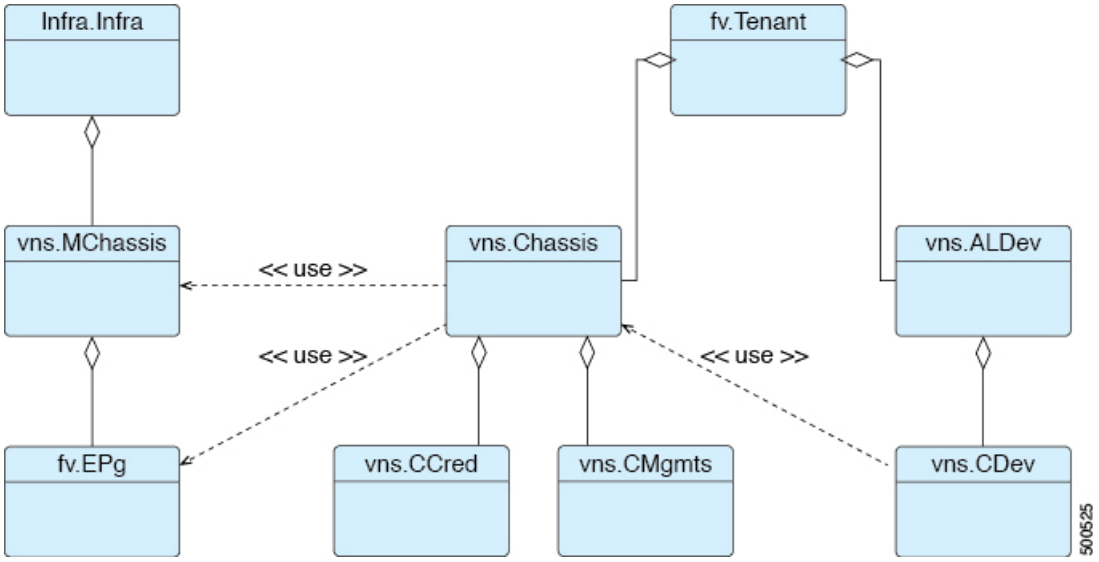
次の図に、デバイス マネージャのオブジェクト モデルを示します。

図 *15*：デバイス マネージャのオブジェクト モデル



次の図に、シャーシ マネージャのオブジェクト モデルを示します。

図 *16*：シャーシ マネージャのオブジェクト モデル



デバイス マネージャとシャーシ マネージャの動作

デバイス マネージャとシャーシ マネージャに関し、次の動作が適用されます。

- `DevMgr` オブジェクトは必要ありません。`LDevVip` から `DevMgr` への関係がない場合、システムはデバイス マネージャを定義せずにコールアウトを実行します。
- `Policymgr` が健全性チェックを実行し、`LDevVip` から `MDev` への関係が `LDevVip` から `DevMgr` と `MDevMgr` を経由して `MDev` までの 1 つのリレーション パスに一致することを保証します。これに当てはまらない場合はエラーが発生し、それ以降のコールアウトが阻止されます。
- `LDevVip` から `DevMgr`、`DevMgr` から `MDevMgr`、または `MDevMgr` から正しい `MDev` への関係が追加または変更された場合は、クラスタがリセットされ、最初から設定されます。
- `Chassis` オブジェクトは必要ありません。`CDev` から `Chassis` への関係がない場合、システムはシャーシを定義せずにコールアウトを実行します。
- `Policymgr` が健全性チェックを実行し、`CDev` から `LDevVip` を経由して `MDev` までの関係が、`CDev` から `Chassis` と `MChassis` を経由して `MDev` までの 1 つのリレーション パスに一致することを保証します。これに当てはまらない場合はエラーが発生し、それ以降のコールアウトが阻止されます。
- `CDev` から `Chassis`、`Chassis` から `MChassis`、または `MChassis` から正しい `MDev` への関係が追加または変更された場合は、クラスタがリセットされ、最初から設定されます。

GUI を使用したデバイス マネージャの作成

GUI を使用して、テナントにデバイス マネージャを作成することができます。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。
ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。
ステップ 3 [Navigation] ペインで、*[tenant_name]* > [L4-L7 Services] > [Device Managers] の順に選択します。
ステップ 4 [Work] ペインで、[Actions] > [Create Device Manager] の順に選択します。
ステップ 5 [Create Device Manager] ダイアログボックスで、必要に応じてフィールドに入力します。
ステップ 6 [Submit] をクリックします。

GUI を使用したシャーシの作成

GUI を使用して、テナントにシャーシを作成することができます。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。
ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。
ステップ 3 [Navigation] ペインで、*[tenant_name]* > [L4-L7 Services] > [Chassis] の順に選択します。
ステップ 4 [Work] ペインで、[Actions] > [Create Chassis] の順に選択します。
ステップ 5 [Create Chassis] ダイアログボックスで、必要に応じてフィールドに入力します。
ステップ 6 [Submit] をクリックします。

デバイス マネージャとシャーシ マネージャの XML の例

以降の項の XML の例は、`Insieme` パッケージがロードされており、`uni/infra/mDev-Insieme-Generic-1.0` 識別名が提供されていることを前提としています。

MDevMgr オブジェクトを作成する XML の例

`MDevMgr` オブジェクトは `MDev` オブジェクトと類似しており、naming プロパティとして vendor、model、および version があります。マネージャでさまざまなタイプのクラスタを管理できる場合

は、複数の MDevMgrToMDev の関係を作成できます。次に、MDevMgr オブジェクトを作成する XML の例を示します。

```
<polUni>
  <infraInfra>
    <vnsMDevMgr
        vendor="Insieme"
        model="DevMgr"
        version="1.0"
        >
      <vnsRsMDevMgrToMDev tDn="uni/infra/mDev-Insieme-Generic-1.0"/>
    </vnsMDevMgr>
  </infraInfra>
</polUni>
```

次に、テナント tenant1 内に MDevMgr オブジェクトを作成する XML の例を示します。

```
<polUni>
    <fvTenant name="tenant1">
      <vnsDevMgr name="Foo">
        <vnsCMgmts
            host="10.10.11.11"
            port="1234"/>
        <vnsCMgmts
            host="10.10.11.12"
            port="1234"/>
        <vnsCMgmts
            host="10.10.11.13"
            port="1234"/>
        <vnsCCred name="username" value="admin"/>
        <vnsCCredSecret name="password" value="letmein"/>
        <vnsRsDevMgrToMDevMgr tDn="uni/infra/mDevMgr-Insieme-DevMgr-1.0"/>
    </vnsDevMgr>
  </fvTenant>
</polUni>
```

LDevVip オブジェクトを DevMgr オブジェクトと関連付ける XML の例

LDevVip オブジェクトと DevMgr オブジェクトは、次の XML の例に示すように、LDevVip から DevMgr への関係を作成することによって関連付けます。

```
<polUni>
    <fvTenant name="tenant1">
        <vnsLDevVip name="InsiemeCluster" devtype="VIRTUAL">
            ...
            <vnsRsMDevAtt tDn="uni/infra/mDev-Insieme-Generic-1.0"/>
            <vnsRsALDevToDevMgr tDn="uni/tn-tenant1/devMgr-Foo"/>
            ...
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

MChassis オブジェクトを作成する XML の例

MChassis オブジェクトは MDev オブジェクトと類似しており、naming プロパティとして vendor、model、および version があります。MChassisToMDev の関係によってデバイス タイプが決まります。次に、MChassis オブジェクトを作成する XML の例を示します。

```
<polUni>
  <infraInfra>
    <vnsMChassis vendor="Insieme" model="DevMgr" version="1.0">
      <vnsRsMChassisToMDev tDn="uni/infra/mDev-Insieme-Generic-1.0"/>
    </vnsMChassis>
```

```
  </infraInfra>
</polUni>
```

シャーシ オブジェクトを作成する XML の例

次に、Chassis オブジェクトを作成する XML の例を示します。

```
<polUni>
  <fvTenant name="tenant1">
    <vnsChassis name="Foo">
        <vnsCMgmts
            host="10.10.11.11"
            port="1234"/>
        <vnsCMgmts
            host="10.10.11.12"
            port="1234"/>
        <vnsCMgmts
            host="10.10.11.13"
            port="1234"/>
      <vnsCCred name="username" value="admin"/>
      <vnsCCredSecret name="password" value="letmein"/>
      <vnsRsChassisToMChassis tDn="uni/infra/mChassis-Insieme-DevMgr-1.0"/>
    </vnsChassis>
  </fvTenant>
</polUni>
```

CDev オブジェクトをシャーシ オブジェクトと関連付ける XML の例

CDev オブジェクトと Chassis オブジェクトは、次の XML の例に示すように、CDev から Chassis への関係を作成することによって関連付けます。

```
<polUni>
    <fvTenant name="tenant1">
        <vnsLDevVip name="InsiemeCluster" devtype="VIRTUAL">
           ...
           <vnsCDev name="Generic1" devCtxLbl="C1">
              ...
              <vnsRsCDevToChassis tnVnsChassisName="Foo"/>
            </vnsCDev>
            <vnsRsALDevToDevMgr tDn="uni/tn-coke/devMgr-Foo"/>
           ...
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

デバイスとシャーシのコールアウト

ここでは、デバイスおよびシャーシ マネージャのパラメータを含むデバイス、クラスタ、およびサービスのコールアウトの例を示します。パラメータはすべてのコールアウトに追加されます。

デバイスの deviceValidate コールアウトの例

次の deviceValidate コールアウトの例に、デバイス固有のコードを太字で示します。

```
2014-10-03 17:38:51,035 DEBUG 140230105585408 [42.42.42.101, 0]:  deviceValidate
{'args': ('1.0',),
 'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
```

```
            'host': '42.42.42.101',
            'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                        'hosts': {'10.10.11.11': {'port': 1234},
                                  '10.10.11.12': {'port': 1234},
                                  '10.10.11.13': {'port': 1234}},
                        'name': 'Foo'},
            'port': 80,
            'version': '1.0',
            'virtual': True}}
```

デバイスの deviceAudit コールアウトの例

次の deviceAudit コールアウトの例に、デバイス固有のコードを太字で示します。

```
2014-10-03 17:38:56,072 DEBUG 140230088800000 [42.42.42.100, 2]:  deviceAudit
{'args': ({(11, '', 'ext'): {'label': 'in', 'state': 0},
    (11, '', 'int'): {'label': 'out', 'state': 0}},
    {(4, 'oneFolder', 'foo'): {'ackedState': 0,
        'state': 0,
        'transaction': 0,
        'value': {(5, 'oneParam', 'foo'): {'ackedState': 0,
            'state': 0,
            'transaction': 0,
            'value': 'bar'}}}}),
'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
    'host': '42.42.42.100',
    'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                'hosts': {'10.10.11.11': {'port': 1234},
                          '10.10.11.12': {'port': 1234},
                          '10.10.11.13': {'port': 1234}},
                'name': 'Foo'},
    'port': 80,
    'version': '1.0',
    'virtual': True}}
```

デバイスの clusterAudit コールアウトの例

次の clusterAudit コールアウトの例に、デバイス固有のコードを太字で示します。

```
2014-10-03 17:39:01,097 DEBUG 140229734295296 [42.42.42.99, 4]:  clusterAudit
{'args': ({(12, '', 'ext'): {'cifs': {'Generic1': 'ext', 'Generic2': 'ext'},
                             'label': 'in',
                             'state': 0},
           (12, '', 'inside'): {'cifs': {'Generic1': 'ext',
                                         'Generic2': 'ext'},
                                'label': 'in',
                                'state': 0},
           (12, '', 'int'): {'cifs': {'Generic1': 'int', 'Generic2': 'int'},
                             'label': 'out',
                             'state': 0},
           (12, '', 'outside'): {'cifs': {'Generic1': 'int',
                                          'Generic2': 'int'},
                                 'label': 'out',
                                 'state': 0}},
          {}),
 'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
            'devs': {'Generic1': {'creds': {'password': '<hidden>',
                                            'username': 'nsroot'},
                                  'host': '42.42.42.100',
                                  'port': 80,
                                  'virtual': True},
                     'Generic2': {'creds': {'password': '<hidden>',
                                            'username': 'nsroot'},
                                  'host': '42.42.42.101',
                                  'port': 80,
                                  'virtual': True}},
```

```
            'host': '42.42.42.99',
            'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                        'hosts': {'10.10.10.11': {'port': 1234},
                                  '10.10.10.12': {'port': 1234},
                                  '10.10.10.13': {'port': 1234}},
                        'name': 'Foo'},
            'port': 80,
            'version': '1.0',
            'virtual': True}}
```

デバイスの serviceAudit コールアウトの例

次の serviceAudit コールアウトの例に、デバイス固有のコードを太字で示します。

```
2014-10-03 17:39:06,169 DEBUG 140229725902592 [42.42.42.99, 5]:  serviceAudit
{'args': ({(0, '', 4474): {'ackedState': 0,
                           'state': 2,
                           'transaction': 0,
                           'txid': 10000,
                           'value': {(1, '', 5787): {
                                       ...
                           }},
 'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
            'devs': {'Generic1': {'creds': {'password': '<hidden>',
                                            'username': 'nsroot'},
                                  'host': '42.42.42.100',
                                  'port': 80,
                                  'virtual': True},
                     'Generic2': {'creds': {'password': '<hidden>',
                                            'username': 'nsroot'},
                                  'host': '42.42.42.101',
                                  'port': 80,
                                  'virtual': True}},
            'host': '42.42.42.99',
            'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                        'hosts': {'10.10.11.11': {'port': 1234},
                                  '10.10.11.12': {'port': 1234},
                                  '10.10.11.13': {'port': 1234}},
                        'name': 'Foo'},
            'port': 80,
            'version': '1.0',
            'virtual': True}}
```

シャーシの deviceValidate コールアウトの例

次の deviceValidate コールアウトの例に、シャーシ固有のコードを太字で示します。

```
2014-11-13 19:33:16,066 DEBUG 140719921972992 [42.42.42.101, 0]: request: deviceValidate
 {'args': ('1.0',),
  'device': {'chassis': {'creds': {'username': 'admin', 'password': '<hidden>'},
                         'hosts': {'10.10.11.11': {'port': 1234},
                                   '10.10.11.12': {'port': 1234},
                                   '10.10.11.13': {'port': 1234}},
                         'name': 'Foo'},
             'creds': {'username': 'nsroot', 'pasword': '<hidden>'},
             'host': '42.42.42.100',
             'port': 80,
             'virtual': True}}
```

シャーシの deviceAudit コールアウトの例

次の deviceAudit コールアウトの例に、シャーシ固有のコードを太字で示します。

```
2014-10-03 17:38:56,072 DEBUG 140230088800000 [42.42.42.100, 2]:  deviceAudit
 {'args': ({(11, '', 'ext'): {'label': 'in', 'state': 0},
            (11, '', 'int'): {'label': 'out', 'state': 0}},
           {(4, 'oneFolder', 'one'): {'ackedState': 0,
                                      'state': 0,
                                      'transaction': 0,
                                      'value': {(5, 'oneParam', 'one'): {'ackedState': 0,
                                                                         'state': 0,
                                                                         'transaction': 0,
                                                                         'value': 'foo'}}}}),
  'device': {'chassis': {'creds': {'username': 'admin', 'password': '<hidden>'},
                         'hosts': {'10.10.11.11': {'port': 1234},
                                   '10.10.11.12': {'port': 1234},
                                   '10.10.11.13': {'port': 1234}},
                         'name': 'Foo'},
             'creds': {'password': '<hidden>', 'username': 'nsroot'},
             'host': '42.42.42.101',
             'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                         'hosts': {'10.10.10.11': {'port': 1234},
                                   '10.10.10.12': {'port': 1234},
                                   '10.10.10.13': {'port': 1234}},
                         'name': 'Foo'},
             'port': 80,
             'version': '1.0',
             'virtual': True}}
```

シャーシの clusterAudit コールアウトの例

次の clusterAudit コールアウトの例に、シャーシ固有のコードを太字で示します。

```
2014-10-03 17:39:01,097 DEBUG 140229734295296 [42.42.42.99, 4]:  clusterAudit
 {'args': ({(12, '', 'ext'): {'cifs': {'Generic1': 'ext', 'Generic2': 'ext'},
                              'label': 'in',
                              'state': 0},
            (12, '', 'inside'): {'cifs': {'Generic1': 'ext',
                                          'Generic2': 'ext'},
                                 'label': 'in',
                                 'state': 0},
            (12, '', 'int'): {'cifs': {'Generic1': 'int', 'Generic2': 'int'},
                              'label': 'out',
                              'state': 0},
            (12, '', 'outside'): {'cifs': {'Generic1': 'int',
                                           'Generic2': 'int'},
                                  'label': 'out',
                                  'state': 0}},
           {}),
  'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
             'devs': {'Generic1': {'chassis': {'creds': {'password': '<hidden>',
                                                         'username': 'admin'},
                                               'hosts': {'10.10.11.11': {'port': 1234},
                                                         '10.10.11.12': {'port': 1234},
                                                         '10.10.11.13': {'port': 1234}},
                                               'name': 'Foo'},
                                   'creds': {'username': 'nsroot', 'pasword': '<hidden>'},
                                   'host': '42.42.42.100',
                                   'port': 80,
                                   'virtual': True},
                      'Generic2': {'chassis': {'creds': {'password': '<hidden>',
                                                         'username': 'admin'},
                                               'hosts': {'10.10.11.11': {'port': 1234},
                                                         '10.10.11.12': {'port': 1234},
                                                         '10.10.11.13': {'port': 1234}},
```

```
                                        'name': 'Foo'},
                                'creds': {'username': 'nsroot', 'pasword': '<hidden>'},
                                'host': '42.42.42.101',
                                'port': 80,
                                'virtual': True}},
'host': '42.42.42.99',
'manager': {'creds': {'password': '<hidden>', 'username': 'admin'},
            'hosts': {'10.10.11.11': {'port': 1234},
                      '10.10.11.12': {'port': 1234},
                      '10.10.11.13': {'port': 1234}},
            'name': 'Foo'},
'port': 80,
'version': '1.0',
'virtual': True}}
```

シャーシの serviceAudit コールアウトの例

次の serviceAudit コールアウトの例に、シャーシ固有のコードを太字で示します。

```
2014-10-03 17:39:06,169 DEBUG 140229725902592 [42.42.42.99, 5]:  serviceAudit
 {'args': ...,
  'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
             'devs': {'Generic1': {'chassis': {'creds': {'username': 'admin',
                                                         'password': '<hidden>'},
                                               'hosts': {'10.10.11.11': {'port': 1234},
                                                         '10.10.11.12': {'port': 1234},
                                                         '10.10.11.13': {'port': 1234}},
                                               'name': 'Foo'},
                                   'creds': {'username': 'nsroot',
                                             'pasword': '<hidden>'},
                                   'host': '42.42.42.100',
                                   'port': 80,
                                   'virtual': True},
                      'Generic2': {'chassis': {'creds': {'username': 'admin',
                                                         'password': '<hidden>'},
                                               'hosts': {'10.10.11.11': {'port': 1234},
                                                         '10.10.11.12': {'port': 1234},
                                                         '10.10.11.13': {'port': 1234}},
                                               'name': 'Foo'},
                                   'creds': {'username': 'nsroot',
                                             'pasword': '<hidden>'},
                                   'host': '42.42.42.101',
                                   'port': 80,
                                   'virtual': True}},
             'host': '42.42.42.99',
             'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                         'hosts': {'10.10.11.11': {'port': 1234},
                                   '10.10.11.12': {'port': 1234},
                                   '10.10.11.13': {'port': 1234}},
                         'name': 'Foo'},
             'port': 80,
             'version': '1.0',
             'virtual': True}}
```

第 10 章

非管理対象モードの設定

非管理対象モードについて

レイヤ 4 ～ レイヤ 7 サービスの挿入機能によって、管理者は 1 つ以上のサービスを 2 つのエンドポイント グループ間に挿入できます。Application Policy Infrastructure Controller（APIC）はサービスにファブリック リソース（VLAN）を割り当て、サービス グラフに指定された設定に従ってファブリック（リーフ）とサービス アプライアンスをプログラミングします。APIC には、サービスをサービス グラフの一部として使用できるようにするため、そのサービスのデバイス パッケージが必要です。また、APIC は、グラフのインスタンス化時にサービス アプライアンスをプログラミングします。

APIC で、サービス グラフに対してネットワーク リソースのみを割り当てて、グラフのインスタンス化時にファブリック側のみをプログラミングすることができます。サービス アプライアンスのプログラミングにより適した既存のオーケストレータまたは dev-op ツールが環境にすでにあるなど、さまざまな理由でこれが必要になる場合があります。また、サービス アプライアンスのデバイス パッケージが使用できない場合もあります。

サービスの非管理対象モードでは、ネットワーク リソースの割り当てや、ファブリックのプログラミングに対する APIC の動作を選択することができます。有効になっている場合、非管理対象モードでは、APIC がサービス アプライアンスに対してネットワーク リソースのみを割り当て、ファブリック（リーフ）のみをプログラミングするように制限します。デバイスの設定については、データセンターの管理者が外部から実行します。

管理対象および非管理対象の論理デバイスについて

非管理対象モードは、次の XML コードに示すように、論理デバイス（LDevVip）に managed 設定を導入します。

```
<!-- Specified if the device is a managed device-->
<property name="managed"
          type="scalar:Bool"
          owner="management"
          mod="explicit">
          <default value="true"/>
</property>
```

デバイスは管理対象にも、非管理対象にもできます。デバイスを管理対象として設定すると、Application Policy Infrastructure Controller（APIC）はデバイスを管理し、グラフのインスタンス化時にデバイスをプログラミングします。デフォルトでは、デバイスを APIC に登録すると、そのデバイスは管理対象モードで設定されます。

デバイスを非管理対象として設定した場合、つまり managed 設定を false に設定すると、APIC はデバイスをプログラミングしません。APICのみがネットワーク リソースを割り当て、ファブリック側で VLAN/VXLAN をプログラミングします。

次の設定は、デバイス クラスタが非管理対象として設定されている場合は必要はありません。

- デバイス パッケージ
- 論理デバイス（vnsLDevVip）とデバイス（CDev）の接続情報（管理 IP アドレス、クレデンシャル、およびインバンド接続情報）
- サポートされる機能タイプ（go-through、go-to）に関する情報
- コンテキスト認識に関する情報（シングル コンテキストかマルチコンテキスト）

この場合も、APICは論理デバイスおよびデバイスのトポロジ情報（LIF、CIF）を把握する必要があります。この情報は、APICがリーフ上で適切なポートをプログラミングできるようにするために必要であり、また、APIC はこの情報をトラブルシューティング ウィザードの目的に使用することもできます。

さらに、APICは、カプセル化の割り当てに使用する DomP との関係も把握する必要があります。

管理対象および非管理対象の機能ノードについて

非管理対象モードは、次の XML コードに示すように、機能ノード（AbsNode）に managed 設定を導入します。

```
<!-- Specified if the function is using a managed device-->
<property name="managed"
               type="scalar:Bool"
               owner="management"
               mod="explicit">
               <default value="true"/>
</property>
```

機能ノードは管理対象にも、非管理対象にもできます。機能ノードを管理対象として設定すると、その機能ノードは管理対象デバイスを使用できます。Application Policy Infrastructure Controller（APIC）は、グラフのインスタンス化時にデバイスをプログラミングします。デフォルトでは、機能ノードをサービス グラフに追加すると、その機能ノードは管理対象モードで設定されます。

機能ノードを非管理対象として設定した場合、つまり managed 設定を false に設定すると、APIC はパラメータを解決せず、デバイスもプログラミングしません。APIC のみがネットワーク リソースを割り当て、ファブリック側で VLAN/VXLAN をプログラミングします。

次の設定は、機能ノードが非管理対象として設定されている場合は必要はありません。

- MFunc の関係
- AbsFuncProfile
- 設定パラメータ（AbsNode またはエンドポイント グループ上）
- サポートされる機能タイプ（go-through、go-to）に関する情報

この場合も、APIC は機能ノードのネットワーク情報（LIF、CIF）を把握する必要があります。この情報は、APIC がリーフ上でネットワークを適切にプログラミングできるようにするために必要であり、また、APIC はこの情報をトラブルシューティング ウィザードのために使用することもできます。

さらに、次の設定が必要です。

- グラフ インスタンス化時に LDevVip の選択を可能にする LDevCtx
- グラフ インスタンス化時に LIf の選択を可能にする LIfCtx
- LIfCtx 内のブリッジ ドメイン
- LIfCtx でのルート ピアリング
- LIfCtx 内のサブネット

レイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループについて

非管理対象モード機能の一部として、Application Policy Infrastructure Controller（APIC）では、グラフのインスタンス化時にグラフ コネクタに使用するエンドポイント グループを指定できます。これにより、グラフ導入のトラブルシューティングが容易になります。APIC では、リーフにカプセル化情報をダウンロードするために指定したレイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループを使用します。また、APIC はこのエンドポイント グループを使用して仮想デバイスの分散仮想スイッチにポート グループも作成します。さらに、レイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループを使用して、グラフ コネクタのエラー情報や統計情報も集約します。

導入されたグラフ リソースへの可視性の向上に加えて、レイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループも使用して、特定のグラフ インスタンスに使用する静的なカプセル化を指定することもできます。このカプセル化は、複数のグラフ インスタンス間でレイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループを共有することによって、複数のグラフ インスタンス間で共有することもできます。

グラフ コネクタと共に レイヤ 4 ～ レイヤ 7 サービスのエンドポイントをどのように使用できるかを示す XML コードの例については、レイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループとコネクタを関連付ける XML の例, （94 ページ）を参照してください。

グラフ コネクタに対する静的なカプセル化の使用

Application Policy Infrastructure Controller（APIC）は、さまざまなサービス グラフのカプセル化を処理時に割り当てます。一部の使用例では、サービス グラフ内の特定のコネクタに使用するカプセル化を明示的に指定できます。これは静的なカプセル化と呼ばれます。静的なカプセル化は、物理サービスを持つサービス デバイス クラスタがあるサービス グラフ コネクタについてのみサポートされます。仮想サービス デバイスがあるサービス デバイス クラスタは、そのサービス デバイス クラスタに関連付けられた VMware ドメインから動的に割り当てられた VLAN を使用します。

静的なカプセル化は、レイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループの一部としてカプセル化値を指定することによってグラフ コネクタで使用できます。レイヤ 4 ～ レイヤ 7 サービスのエンドポイントで静的なカプセル化の使用方法を示す XML コードの例については、レイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループで静的なカプセル化を使用する XML の例, （94 ページ）を参照してください。

NX-OS スタイルの CLI を使用した物理デバイスの作成

次に、NX-OS スタイルの CLI を使用して物理デバイスを作成する手順の例を示します。

ステップ 1 コンフィギュレーション モードを開始します。

例：
```
apic1# configure
```

ステップ 2 テナントのコンフィギュレーション モードを開始します。
```
tenant tenant_name
```

例：
```
apic1(config)# tenant t1
```

ステップ 3 クラスタを作成します。

例：
```
apic1(config-tenant)# l4l7 cluster name ifav108-asa type physical vlan-domain phyDom5 servicetype
 FW
```

ステップ 4 クラスタ デバイスを追加します。

例：
```
apic1(config-cluster)# cluster-device C1
```

ステップ 5 プロバイダー クラスタ インターフェイスを追加します。

例：
```
apic1(config-cluster)# cluster-interface provider
```

ステップ 6 インターフェイスにメンバー デバイスを追加します。

例：
```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

ステップ 7 コンシューマ クラスタ インターフェイスを追加します。

例：
```
apic1(config-cluster)# cluster-interface consumer
```

ステップ 8 コンシューマ インターフェイスに同じメンバー デバイスを追加します。

例：
```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
```

```
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

ステップ 9 クラスタ作成モードを終了します。

例：
```
apic1(config-cluster)# exit
```

NX-OS スタイルの CLI を使用したハイ アベイラビリティ クラスタの作成

次に、NX-OS スタイルの CLI を使用してハイ アベイラビリティ クラスタを作成する手順の例を示します。

ステップ 1 コンフィギュレーション モードを開始します。

例：
```
apic1# configure
```

ステップ 2 テナントのコンフィギュレーション モードを開始します。
```
tenant tenant_name
```

例：
```
apic1(config)# tenant t1
```

ステップ 3 クラスタを作成します。

例：
```
apic1(config-tenant)# l4l7 cluster name ifav108-asa type physical vlan-domain phyDom5 servicetype
 FW
```

ステップ 4 クラスタ デバイスを追加します。

例：
```
apic1(config-cluster)# cluster-device C1
apic1(config-cluster)# cluster-device C2
```

ステップ 5 プロバイダー クラスタ インターフェイスを追加します。

例：
```
apic1(config-cluster)# cluster-interface provider vlan 101
```

ステップ 6 インターフェイスにメンバー デバイスを追加します。

例：
```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
apic1(config-cluster-interface)# member device C2 device-interface Po2
apic1(config-member)# interface vpc VPCPolASA-2 leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

ステップ 7 別のプロバイダー クラスタ インターフェイスを追加します。

例：
```
apic1(config-cluster)# cluster-interface provider vlan 102
```

ステップ 8 最初のインターフェイスからこの新しいインターフェイスに同じメンバー デバイスを追加します。

例：
```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
apic1(config-cluster-interface)# member device C2 device-interface Po2
apic1(config-member)# interface vpc VPCPolASA-2 leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

ステップ 9 クラスタ作成モードを終了します。

例：
```
apic1(config-cluster)# exit
```

NX-OS スタイルの CLI を使用した仮想デバイスの作成

次に、NX-OS スタイルの CLI を使用して仮想デバイスを作成する手順の例を示します。

ステップ 1 コンフィギュレーション モードを開始します。

例：
```
apic1# configure
```

ステップ 2 テナントのコンフィギュレーション モードを開始します。
```
tenant tenant_name
```

例：
```
apic1(config)# tenant t1
```

ステップ 3 クラスタを作成します。

例：
```
apic1(config-tenant)# l4l7 cluster name ifav108-citrix type virtual vlan-domain ACIVswitch servicetype
 ADC
```

ステップ 4 クラスタ デバイスを追加します。

例：
```
apic1(config-cluster)# cluster-device D1 vcenter ifav108-vcenter vm NSVPX-ESX
```

ステップ 5 コンシューマ クラスタ インターフェイスを追加します。

例：
```
apic1(config-cluster)# cluster-interface consumer
```

ステップ 6 コンシューマ インターフェイスにメンバー デバイスを追加します。

例：
```
apic1(config-cluster-interface)# member device D1 device-interface 1_1
apic1(config-member)# interface ethernet 1/45 leaf 102
ifav108-apic1(config-member)# vnic "Network adapter 2"
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

ステップ 7 プロバイダー クラスタ インターフェイスを追加します。

例：
```
apic1(config-cluster)# cluster-interface provider
```

ステップ 8 プロバイダー インターフェイスに同じメンバー デバイスを追加します。

例：
```
apic1(config-cluster-interface)# member device D1 device-interface 1_1
apic1(config-member)# interface ethernet 1/45 leaf 102
ifav108-apic1(config-member)# vnic "Network adapter 2"
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

ステップ 9 クラスタ作成モードを終了します。

例：
```
apic1(config-cluster)# exit
```

非管理対象モードの XML の例

以降の項の XML の例で、非管理対象モードの管理方法を示します。

非管理対象の LDevVip オブジェクトを作成する XML の例

次に、非管理対象の `LDevVip` オブジェクトを作成する XML の例を示します。

```
<polUni>
    <fvTenant name="HA_Tenant1">
        <vnsLDevVip name="ADCCluster1" devtype="VIRTUAL" managed="no">
            <vnsRsALDevToDomP tDn="uni/vmmp-VMware/dom-mininet"/>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

非管理対象の AbsNode オブジェクトを作成する XML の例

次に、非管理対象の `AbsNode` オブジェクトを作成する XML の例を示します。

```
    <fvTenant name="HA_Tenant1">
        <vnsAbsGraph name="g1">
            <vnsAbsTermNodeProv name="Input1">
                <vnsAbsTermConn name="C1">
                </vnsAbsTermConn>
            </vnsAbsTermNodeProv>

            <!-- Node1 provides a service function in un-managed mode -->
            <vnsAbsNode name="Node1" managed="no">
                <vnsAbsFuncConn name="outside" >
                </vnsAbsFuncConn>
                <vnsAbsFuncConn name="inside" >
                </vnsAbsFuncConn>
            </vnsAbsNode>

            <vnsAbsTermNodeCon name="Output1">
                <vnsAbsTermConn name="C6">
                </vnsAbsTermConn>
            </vnsAbsTermNodeCon>

            <vnsAbsConnection name="CON2" >
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeCon-Output1/AbsTConn"/>
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-outside"/>
            </vnsAbsConnection>

            <vnsAbsConnection name="CON1" >
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-inside"/>
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeProv-Input1/AbsTConn"/>
            </vnsAbsConnection>

        </vnsAbsGraph>
    </fvTenant>
```

レイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループとコネクタを関連付ける XML の例

次に、レイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループとコネクタを関連付ける XML の例を示します。

```
<fvTenant name="HA_Tenant1">
   <vnsLDevCtx ctrctNameOrLbl="any" descr="" dn="uni/tn-HA_Tenant1/ldevCtx-c-any-g-any-n-any"

      graphNameOrLbl="any" name="" nodeNameOrLbl="any">
        <vnsRsLDevCtxToLDev tDn="uni/tn-HA_Tenant1/lDevVip-ADCCluster1"/>
        <vnsLIfCtx connNameOrLbl="inside" descr="" name="inside">
            <vnsRsLIfCtxToSvcEPg tDn="uni/tn-HA_Tenant1/ap-sap/SvcEPg-EPG1"/>
            <vnsRsLIfCtxToBD tDn="uni/tn-HA_Tenant1/BD-provBD1"/>
            <vnsRsLIfCtxToLIf tDn="uni/tn-HA_Tenant1/lDevVip-ADCCluster1/lIf-inside"/>
        </vnsLIfCtx>
        <vnsLIfCtx connNameOrLbl="outside" descr="" name="outside">
            <vnsRsLIfCtxToSvcEPg tDn="uni/tn-HA_Tenant1/ap-sap/SvcEPg-EPG2"/>
            <vnsRsLIfCtxToBD tDn="uni/tn-HA_Tenant1/BD-consBD1"/>
            <vnsRsLIfCtxToLIf tDn="uni/tn-HA_Tenant1/lDevVip-ADCCluster1/lIf-outside"/>
        </vnsLIfCtx>
    </vnsLDevCtx>
</fvTenant>
```

レイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループで静的なカプセル化を使用する XML の例

次に、レイヤ 4 ～ レイヤ 7 サービスのエンドポイント グループで静的なカプセル化を使用する XML の例を示します。

```
<polUni>
    <fvTenant name="HA_Tenant1">
        <fvAp name="sap">
            <vnsSvcEPg name="EPG1" encap="vlan-3510">
            </vnsSvcEPg>
        </fvAp>
    </fvTenant>
</polUni>
```

非管理対象モードの動作

非管理対象モードについて次の動作が適用されます。

- パラメータ解決と非管理対象機能：非管理対象機能では、Application Policy Infrastructure Controller（APIC）はパラメータ解決を実行しません。AbsGraph、エンドポイント グループ、またはその他のすべてのレベルでパラメータを設定する必要はありません。
- VDev と非管理対象機能：非管理対象機能では、APIC はパラメータ解決やデバイス側のプログラミングを実行しません。非管理対象サービス グラフ機能では、VDev ツリーは作成されません。
- 非管理対象モードでのルート ピアリング：非管理対象モードはルート ピアリング機能に影響しません。

- 非管理対象モードでの VNIC の自動配置：非管理対象モードは VNIC の配置機能に影響しません。

第 11 章

コンフィギュレーション パラメータ

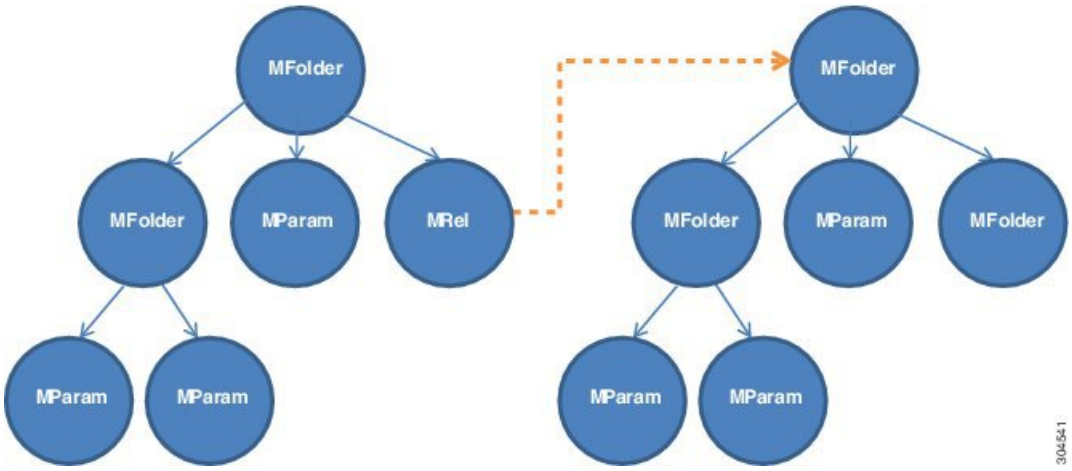
デバイスパッケージ仕様内のコンフィギュレーションパラメータ

デバイス パッケージにはサービス デバイスの仕様を示す XML ファイルが含まれます。この仕様にはデバイス情報およびサービス デバイスによって提供される各種の機能が含まれます。

デバイス仕様の一部として、このファイルにはサービス デバイスによって必要なコンフィギュレーションの宣言が含まれる必要があります。この設定は、グラフのインストール中にサービス デバイスによって提供される各種の機能を設定するために必要です。

次の図は、デバイス パッケージ内のコンフィギュレーション パラメータ階層を示しています。

図 *17*：デバイス パッケージ内のコンフィギュレーション パラメータ階層



MFolder

MFolder は、MParam および他のネストされた MFolder を含む可能性のあるコンフィギュレーション アイテムのグループです。MFolder は次の属性を持ちます。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Description	コンフィギュレーション アイテムを説明します。
Cardinality	コンフィギュレーション アイテムの濃度を指定します。濃度のデフォルト値は `1` です。濃度が `N` であれば、Application Policy Infrastructure Controller（APIC）ではコンフィギュレーション パラメータの N インスタンスの設定が可能です。
ScopedBy	パラメータ解決の範囲を指定します。ScopedBy は、APIC がコンフィギュレーション MO からパラメータを解決する場合にパラメータ値を検索する場所を決定します。 デフォルト値は `Epg` です。サポートされる値は `Tenant`、`Ap`、`Bd`、および `Epg` です。
RsConnector	コンフィギュレーション アイテムを MConn に関連付ける関係。
DevCtx	コンフィギュレーションアイテムをデバイス（LDev）内の特定の物理デバイス（CDev）に関連付けることができます。
Locked	コンフィギュレーション アイテム値がロックされます。一度ロックされると値は変更できません。

MParam

MParam は、単一のコンフィギュレーション パラメータを宣言するコンフィギュレーション パラメータの基本単位です。MParam は次の属性を持ちます。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Description	コンフィギュレーション アイテムを説明します。
Cardinality	コンフィギュレーション アイテムの濃度を指定します。濃度のデフォルト値は 1 です。濃度が N であれば、APIC ではコンフィギュレーション パラメータの N インスタンスの設定が可能です。
RsConnector	コンフィギュレーション アイテムを MConn に関連付ける関係。
Mandatory	コンフィギュレーション アイテムが必須としてマークされます。
Locked	コンフィギュレーション アイテム値がロックされます。一度ロックされると値は変更できません。
Validation	値の検証方法を指定します。

MRel

MRel は 1 つの MFolder が別の MFolder を参照することを可能にします。MFolder 内の MRel を使用して、管理者は含む側の MFolder を、MRel 内に含まれる RsTarget 関係によって MRel からポイントされる MFolder に関連付けることができます。MRel は次の属性を持ちます。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Description	コンフィギュレーション アイテムを説明します。
Cardinality	コンフィギュレーション アイテムの濃度を指定します。濃度のデフォルト値は 1 です。
RsTarget	コンフィギュレーション フォルダを別の MFolder に関連付ける関係。この関係に対する TDn の値はターゲット フォルダの DN です。
RsConnector	コンフィギュレーション アイテムを MConn に関連付ける関係。
Mandatory	コンフィギュレーション アイテムが必須としてマークされます。

デバイス パッケージ仕様の設定スコープ

デバイス仕様ファイルで、コンフィギュレーション アイテムは異なるセクションで配置されます。

MDevCfg

MDevCfg のセクションでは、デバイスを使用するすべてのサービス グラフで共有されるデバイス レベルの設定について説明します。Application Policy Infrastructure Controller（APIC）は、このセクションで説明されるコンフィギュレーション アイテムを使用して作成されたコンフィギュレーション オブジェクトの参照カウントを実行します。オブジェクトは、デバイスを使用しているすべてのグラフ インスタンスが削除された後にのみサービス デバイスから削除されます。

MFuncCfg

MFuncCfg は、サービス機能に対してローカルで、サービス機能に固有なコンフィギュレーションについて説明します。APIC は、このセクションで説明されるコンフィギュレーション アイテムによって作成されたコンフィギュレーション オブジェクトの参照カウントを実行します。オブジェクトが作成され、サービス機能がインスタンス化または削除されたときに削除されます。

MGrpCfg

MGrpCfg は、デバイスを使用するサービス グラフのすべての機能によって共有される設定を説明します。APIC は、このセクションで説明されるコンフィギュレーション アイテムを使用して作成されたコンフィギュレーション オブジェクトの参照カウントを実行します。オブジェクトは、サービス グラフからすべての機能が削除された後にサービス デバイスから削除されます。

デバイス パッケージ内のコンフィギュレーション パラメータの XML の例

次の XML の例は、デバイス パッケージ内のコンフィギュレーション パラメータを示しています。

```
<vnsMFolder key="VServer" scopedBy="epg">
    <vnsRsConnector tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"/>
    <vnsMParam key="vservername" description="Name of VServer" mandatory="true"/>
    <vnsMParam key="vip" description="Virtual IP"/>
    <vnsMParam key="subnet" description="Subnet IP"/>
    <vnsMParam key="port" description="Port for Virtual server"/>
    <vnsMParam key="persistencetype" description="persistencetype"/>
    <vnsMParam key="servicename" description="Service bound to this vServer"/>
    <vnsMParam key="servicetype" description="Service bound to this vServer"/>
    <vnsMParam key="clttimeout" description="Client timeout"/>
    <vnsMFolder key="VServerGlobalConfig"
      description="This references the global configuration">
        <vnsMRel key="ServiceConfig">
            <vnsRsTarget tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Service"/>
        </vnsMRel>
        <vnsMRel key="ServerConfig">
            <vnsRsTarget tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Server"/>
        </vnsMRel>
        <vnsMRel key="VipConfig">
            <vnsRsTarget
```

```
                tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Network/mFolder-vip"/>
              <vnsRsConnector tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"/>
        </vnsMRel>
    </vnsMFolder>
</vnsMFolder>
```

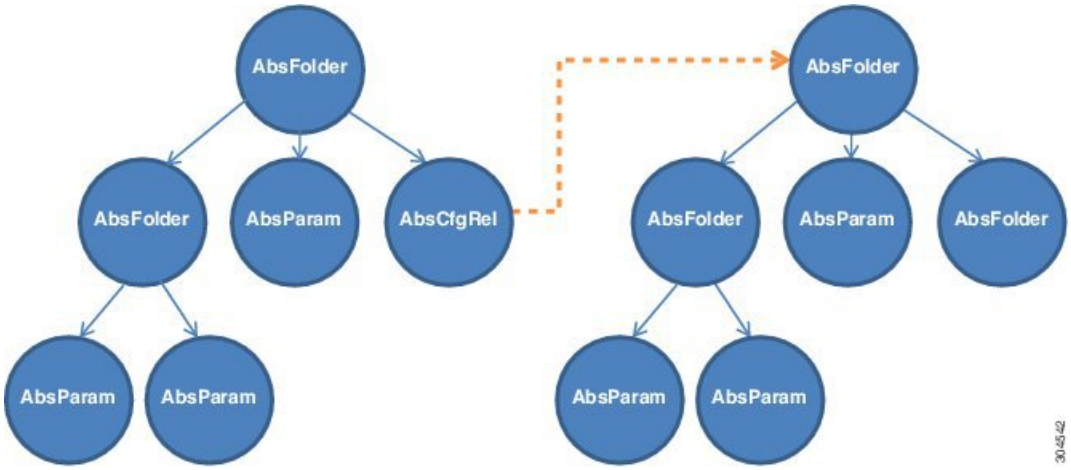
抽象機能プロファイル内のコンフィギュレーションパラメータ

抽象プロファイルを使用すると、管理者はコンフィギュレーション パラメータのデフォルト値を設定できます。抽象機能プロファイルには値を持つコンフィギュレーション パラメータが含まれます。これらの値がグラフ インスタンス作成時にデフォルト値として使用されます。

抽象機能プロファイルはサービス グラフの機能ノードに接続されます。抽象機能プロファイルで指定されたデフォルト値は、グラフのインスタンス化の際にサービス デバイスに機能をレンダリングする場合に使用されます。

次の図は、抽象機能プロファイル内のコンフィギュレーション パラメータ階層を示しています。

図 *18*：抽象機能プロファイル内部のコンフィギュレーション パラメータ階層



AbsFolder

AbsFolder は、AbsParam および他のネストされた AbsFolder を含むことのできるコンフィギュレーション アイテムのグループです。各 AbsFolder に対して、デバイス パッケージ内に MFolder が必要です。Application Policy Infrastructure Controller（APIC）は、各 AbsFolder を検証して、パッケージ内に AbsFolder に対応する MFolder が存在することを確認します。AbsFolder には、次の属性があります。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Description	コンフィギュレーション アイテムを説明します。
Cardinality	コンフィギュレーションアイテムの濃度を指定します。デフォルト値は `1` です。
ScopedBy	パラメータ解決の範囲を指定します。ScopedBy は、APIC がコンフィギュレーション MO からパラメータを解決する場合にパラメータ値を検索する場所を決定します。 デフォルト値は `Epg` です。サポートされる値は `Tenant`、`Ap`、`Bd`、および `Epg` です。
DevCtx	コンフィギュレーションアイテムをデバイス クラスタ（LDev）内の特定の物理デバイス（CDev）に関連付けることができます。
Locked	コンフィギュレーション アイテム値がロックされます。一度ロックされると値は変更できません。

AbsParam

AbsParam はコンフィギュレーション パラメータの基本単位です。 AbsParam は単一のコンフィギュレーション パラメータを定義します。AbsFolder と同様、各 AbsParam に対してデバイス仕様内に対応する MFolder が存在する必要があります。APIC は仕様を検証して、パッケージ内に AbsParam に対応する MFolder が存在することを確認します。AbsParam の値は、MParam 内で指定される検証メソッドを使用して検証されます。AbsParam には、次の属性があります。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Value	特定のコンフィギュレーション アイテムの値を保持します。値は MParam ではサポートされません。
Description	コンフィギュレーション アイテムを説明します。
Cardinality	コンフィギュレーションアイテムの濃度を指定します。デフォルト値は `1` です。
Mandatory	コンフィギュレーション アイテムが必須としてマークされます。
Locked	コンフィギュレーション アイテム値がロックされます。一度ロックされると値は変更できません。

属性	説明
Validation	コンフィギュレーション パラメータの検証に使用する検証メカニズムを指定します。

AbsRel

AbsRel は 1 つの AbsFolder が別の AbsFolder を参照することを可能にします。AbsRel には、次の属性があります。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Value	特定のコンフィギュレーション アイテムの値を保持します。値は MParam ではサポートされません。
Description	コンフィギュレーション アイテムを説明します。
Cardinality	コンフィギュレーションアイテムの濃度を指定します。デフォルト値は1です。
Mandatory	コンフィギュレーション アイテムが必須としてマークされます。

抽象機能プロファイルの設定スコープ

抽象機能プロファイルでは、コンフィギュレーションパラメータはデバイスパッケージ内の場合と似た方法で構成されます。3 種類のスコープがあります。

AbsDevCfg

このセクションは、デバイス パッケージ内のデバイス レベル設定と宣言される、コンフィギュレーションアイテムのデフォルト値を提供します。コンフィギュレーションアイテムはMDevCfgで指定されます。

各コンフィギュレーション アイテムに対して、デバイス パッケージに同等のコンフィギュレーション アイテムが存在する必要があります。

このセクションで説明される設定は、デバイスを使用するサービス グラフで共有されます。Application Policy Infrastructure Controller（APIC）は、このセクションで説明されるコンフィギュレーションアイテムを使用して作成されたコンフィギュレーションオブジェクトの参照カウントを実行します。オブジェクトは、デバイスを使用しているすべてのグラフ インスタンスが削除された後にのみサービス デバイスから削除されます。

AbsGrpCfg

このセクションは、デバイス パッケージ内のデバイス レベル設定と宣言される、コンフィギュレーション アイテムのデフォルト値を提供します。コンフィギュレーション アイテムは MGrpCfg で指定されます。

各コンフィギュレーション アイテムに対して、デバイス パッケージに同等のコンフィギュレーション アイテムが存在する必要があります。

このセクションで説明される設定は、デバイスを使用するサービス グラフのすべての機能で共有されます。APIC は、このセクションで説明されるコンフィギュレーション アイテムを使用して作成されたコンフィギュレーション オブジェクトの参照カウントを実行します。オブジェクトは、デバイスを使用しているすべてのグラフインスタンスが削除された後にのみサービスデバイスから削除されます。

AbsFuncCfg

このセクションは、デバイス パッケージ内の機能レベル設定と宣言される、コンフィギュレーション アイテムのデフォルト値を提供します。コンフィギュレーション アイテムは MFuncCfg で指定されます。

各コンフィギュレーション アイテムに対して、デバイス パッケージに同等のコンフィギュレーション アイテムが存在する必要があります。

このセクションは、サービス機能にローカルな設定を説明するために使用されます。このセクションで説明されている設定は、サービス機能に固有のものです。APIC は、このセクションで説明されるコンフィギュレーションアイテムによって作成されたコンフィギュレーションオブジェクトの参照カウントを実行します。オブジェクトが作成され、サービス機能がインスタンス化または削除されたときに削除されます。

コンフィギュレーションパラメータを持つ抽象機能プロファイルに対する XML POST の例

次の XML POST の例は、コンフィギュレーション パラメータを持つ抽象機能プロファイルを示しています。

```
<vnsAbsFuncProfContr name = "NP">
    <vnsAbsFuncProfGrp name = "Grp1">
        <vnsAbsFuncProf name = "P1">
            <vnsRsProfToMFunc tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB"/>
            <vnsAbsDevCfg name="D1">
                <vnsAbsFolder key="Service" name="Service-Default" cardinality="n">
                    <vnsAbsParam name="servicetype" key="servicetype" value="TCP"/>
                    <vnsAbsParam name="serviceport" key="serviceport" value="80"/>
                    <vnsAbsParam name="maxclient" key="maxclient" value="1000"/>
                    <vnsAbsParam name="maxreq" key="maxreq" value="100"/>
                    <vnsAbsParam name="cip" key="cip" value="enable"/>
                    <vnsAbsParam name="usip" key="usip" value="enable"/>
                    <vnsAbsParam name="sp" key="sp" value=""/>
                    <vnsAbsParam name="svrtimeout" key="svrtimeout" value="60"/>
                    <vnsAbsParam name="clttimeout" key="clttimeout" value="60"/>
                    <vnsAbsParam name="cka" key="cka" value="NO"/>
                    <vnsAbsParam name="tcpb" key="tcpb" value="NO"/>
                    <vnsAbsParam name="cmp" key="cmp" value="NO"/>
                </vnsAbsFolder>
            </vnsAbsDevCfg>
```

```
            <vnsAbsFuncCfg name="SLB">
                <vnsAbsFolder key="VServer" name="VServer-Default">
                    <vnsAbsParam name="port" key="port" value="80"/>
                    <vnsAbsParam name="persistencetype" key="persistencetype"
                      value="cookie"/>
                    <vnsAbsParam name="clttimeout" key="clttimeout" value="100"/>
                    <vnsAbsParam name="servicetype" key="servicetype" value="TCP"/>
                    <vnsAbsParam name="servicename" key="servicename"/>
                </vnsAbsFolder>
            </vnsAbsFuncCfg>
        </vnsAbsFuncProf>
    </vnsAbsFuncProfGrp>
</vnsAbsFuncProfContr>
```

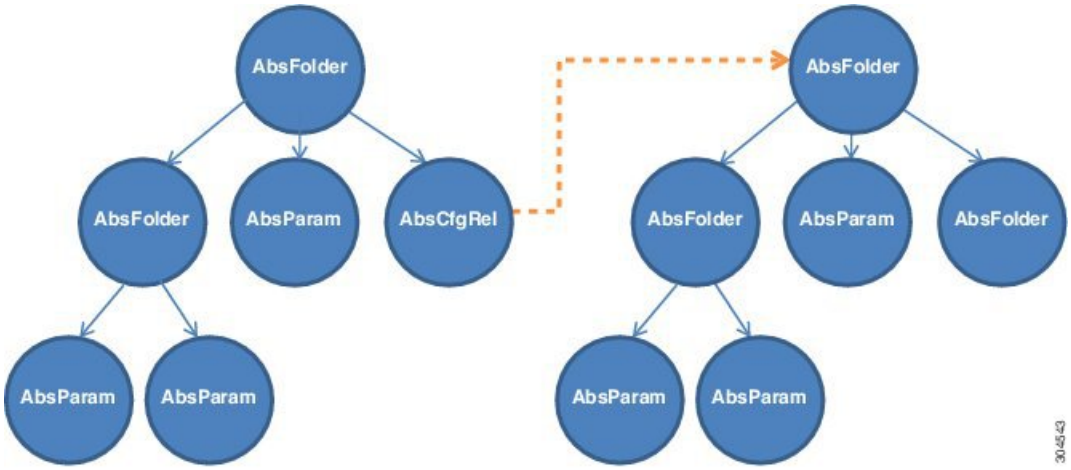
サービス グラフでの抽象機能ノード内のコンフィギュレーション パラメータ

サービス グラフ内の機能ノードを使用して、管理者はコンフィギュレーション パラメータの値を設定できます。これらの値は、グラフのインストール時に使用されます。

抽象機能ノードでは、コンフィギュレーション パラメータは抽象機能プロファイル内の場合と似た方法で構成されます。

次の図は、抽象機能ノード内のコンフィギュレーション パラメータ階層を示しています。

図 *19*：抽象機能ノード内のコンフィギュレーション パラメータ



AbsDevCfg

このセクションは、デバイス パッケージ内のデバイス レベル設定と宣言される、コンフィギュレーション アイテムのデフォルト値を提供するために使用されます。コンフィギュレーション アイテムは MDevCfg で指定されます。

これらの各コンフィギュレーション アイテムに対して、デバイス パッケージに同等のコンフィギュレーション アイテムが存在する必要があります。

AbsGrpCfg

このセクションは、デバイス パッケージ内のデバイス レベル設定と宣言される、コンフィギュレーションアイテムのデフォルト値を提供するために使用されます。コンフィギュレーションアイテムは MGrpCfg で指定されます。

これらの各コンフィギュレーション アイテムに対して、デバイス パッケージに同等のコンフィギュレーション アイテムが存在する必要があります。

このセクションで説明される設定は、デバイスを使用するサービス グラフのすべての機能で共有されます。Application Policy Infrastructure Controller（APIC）は、このセクションで説明されるコンフィギュレーションアイテムを使用して作成されたコンフィギュレーションオブジェクトの参照カウントを実行します。オブジェクトは、サービス グラフからすべての機能が削除された後にサービス デバイスから削除されます。

AbsFuncCfg

このセクションは、デバイス パッケージ内の機能レベル設定と宣言される、コンフィギュレーションアイテムのデフォルト値を提供するために使用されます。コンフィギュレーションアイテムは MFuncCfg で指定されます。

これらの各コンフィギュレーション アイテムに対して、デバイス パッケージに同等のコンフィギュレーション アイテムが存在する必要があります。

このセクションは、サービス機能にローカルな設定を説明するために使用されます。このセクションで説明されている設定は、サービス機能に固有のものです。APICは、このセクションで説明されるコンフィギュレーションアイテムによって作成されたコンフィギュレーションオブジェクトの参照カウントを実行します。オブジェクトが作成され、サービス機能がインスタンス化または削除されたときに削除されます。

AbsFolder

AbsFolderは、AbsParamsおよび他のネストされたAbsFolderを含むことのできるコンフィギュレーション アイテムのグループです。各 AbsFolder に対して、デバイス パッケージ内に MFolder が必要です。APIC は、各 AbsFolder を検証して、パッケージ内に AbsFolder に対応する MFolder が存在することを確認します。AbsFolder には、次の属性があります。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Description	コンフィギュレーション アイテムを説明します。
Cardinality	コンフィギュレーションアイテムの濃度を指定します。デフォルト値は1です。

属性	説明
ScopedBy	パラメータ解決の範囲を指定します。ScopedBy は、APIC がコンフィギュレーション MO からパラメータを解決する場合にパラメータ値を検索する場所を決定します。 デフォルト値は `Epg` です。サポートされる値は `Tenant`、`Ap`、`Bd`、および `Epg` です。
RsCfgToConn	コンフィギュレーション アイテムを AbsConn に関連付ける関係。
DevCtx	コンフィギュレーションアイテムをデバイス（LDev）内の特定の物理デバイス（CDev）に関連付けることができます。
Locked	コンフィギュレーション アイテム値がロックされます。一度ロックされると値は変更できません。

AbsParam

AbsParam はコンフィギュレーション パラメータの基本単位です。 AbsParam は単一のコンフィギュレーション パラメータを定義します。AbsFolder と同様、各 AbsParam に対してデバイス仕様内に対応する MFolder が存在する必要があります。APIC は仕様を検証して、パッケージ内に AbsParam に対応する MFolder が存在することを確認します。AbsParam の値は、MParam 内で指定される検証メソッドを使用して検証されます。AbsParam には次の属性があります。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Value	特定のコンフィギュレーション アイテムの値を保持します。値は MParam ではサポートされません。
Description	コンフィギュレーション アイテムを説明します。
Cardinality	コンフィギュレーションアイテムの濃度を指定します。デフォルト値は `1` です。
RsCfgToConn	コンフィギュレーション アイテムを MConn に関連付ける関係。
Mandatory	コンフィギュレーション アイテムが必須としてマークされます。
Locked	コンフィギュレーション アイテム値がロックされます。一度ロックされると値は変更できません。
Validation	コンフィギュレーション パラメータの検証に使用する検証メカニズムを指定します。

AbsRel

AbsRel は 1 つの AbsFolder が別の AbsFolder を参照することを可能にします。AbsRel には次の属性があります。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Value	特定のコンフィギュレーション アイテムの値を保持します。値は MParam ではサポートされません。
Description	コンフィギュレーション アイテムを説明します。
Cardinality	コンフィギュレーションアイテムの濃度を指定します。デフォルト値は 1 です。
RsCfgToConn	コンフィギュレーション アイテムを MConn に関連付ける関係。
Mandatory	コンフィギュレーション アイテムが必須としてマークされます。
Locked	コンフィギュレーション アイテム値がロックされます。一度ロックされると値は変更できません。

コンフィギュレーション パラメータを持つ抽象機能ノードに対する XML POST の例

次の XML POST の例は、コンフィギュレーション パラメータを持つ抽象機能ノードを示しています。

```
<vnsAbsNode name = "SLB" funcType="GoTo" >
    <vnsRsDefaultScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>

    <vnsAbsFuncConn name = "C4" direction = "input">
        <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external" />
    </vnsAbsFuncConn>
    <vnsAbsFuncConn name = "C5" direction = "output">
        <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-internal" />
    </vnsAbsFuncConn>

    <vnsAbsDevCfg>
        <vnsAbsFolder key="Network" name="Network" scopedBy="epg">
            <!-- Following scopes this folder to input terminal or Src Epg -->
          <vnsRsScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>


            <!-- VIP address -->
            <vnsAbsFolder key="vip" name="vip" scopedBy="epg">
                <vnsAbsParam name="vipaddress" key="vipaddress" value=""/>
            </vnsAbsFolder>

            <!-- SNIP address -->
            <vnsAbsFolder key="snip" name="snip" scopedBy="epg">
                <vnsAbsParam name="snipaddress" key="snipaddress" value=""/>
            </vnsAbsFolder>
```

```
        </vnsAbsFolder>

        <vnsAbsFolder key="Service" name="Service" scopedBy="epg" cardinality="n">
          <vnsRsScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>

            <vnsAbsParam name="servicename" key="servicename" value=""/>
            <vnsAbsParam name="servername" key="servername" value=""/>
            <vnsAbsParam name="serveripaddress" key="serveripaddress" value=""/>
        </vnsAbsFolder>
    </vnsAbsDevCfg>

    <vnsAbsFuncCfg>
        <vnsAbsFolder key="VServer" name="VServer" scopedBy="epg">
          <vnsRsScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>

            <!-- Virtual Server Configuration -->
            <vnsAbsParam name="vip" key="vip" value=""/>
            <vnsAbsParam name="vservername" key="vservername" value=""/>
            <vnsAbsParam name="servicename" key="servicename"/>
            <vnsRsCfgToConn tDn="uni/tn-tenant1/AbsGraph-G3/AbsNode-Node2/AbsFConn-C4" />
        </vnsAbsFolder>
    </vnsAbsFuncCfg>
    <vnsRsNodeToMFunc tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB"/>
</vnsAbsNode>
```

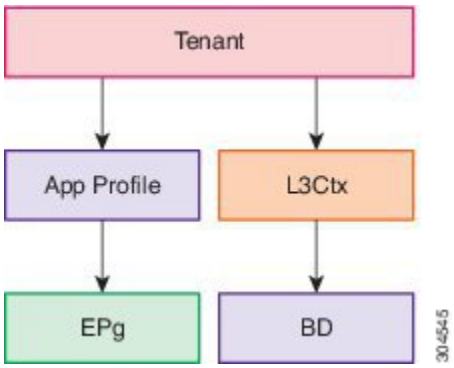
各種の設定 MO 内のコンフィギュレーション パラメータ

管理者は EPG、テナント、BD、または AP などの各種の Application Policy Infrastructure Controller（APIC）MO の一部としてサービス機能に対するコンフィギュレーション パラメータを指定できます。グラフがインスタンス化されると、APIC は各種の場所からパラメータを検索することでグラフに必要な設定を解決します。インスタンス化では、パラメータ値はデバイス スクリプトに解決され、渡されます。

各種の MO 内でコンフィギュレーション パラメータを保持できることの柔軟性により、管理者は単一のサービス グラフを設定し、グラフを異なるテナントまたはエンドポイント グループ（EPG）に対して異なる設定で使用できます。

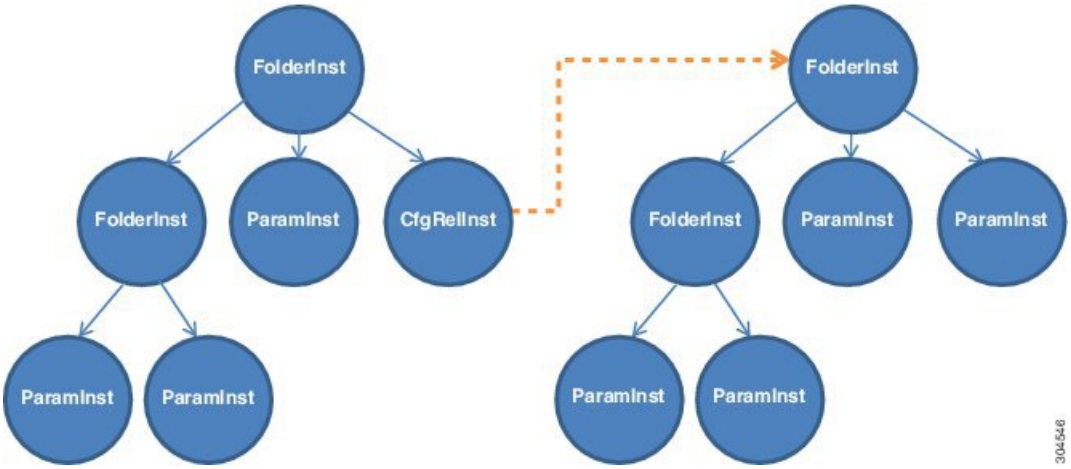
次の図は、APIC MO の階層を示しています。

図 *20*： *APIC MO* の階層



次の図は、各種のコンフィギュレーション MO 内のコンフィギュレーション パラメータを示しています。

図 *21*： 各種の設定 *MO* 内のコンフィギュレーション パラメータ



FolderInst

FolderInst は、ParamInst および他のネストされた FolderInst を含む可能性のあるコンフィギュレーション アイテムのグループです。FolderInst には、次の属性があります。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
ctrctNameOrLbl	パラメータの解決時に一致する FolderInst を検索します。FolderInst をパラメータ解決で使用するには、このプロパティはサービス グラフと関連付けられたコントラクト名と一致する必要があります。一致していない場合、FolderInst はスキップされ、値はこの FolderInst から使用されません。 このフィールドの値を [any] にして、この FolderInst がすべてのコントラクトで使用されるようにできます。
graphNameOrLbl	パラメータの解決時に一致する FolderInst を検索します。FolderInst をパラメータ解決で使用するには、このプロパティはサービス グラフ名と一致する必要があります。一致していない場合、FolderInst はスキップされ、値はこの FolderInst から使用されません。 この FolderInst がすべてのサービス グラフで使用されるようにするには、このフィールドの値を [any] にできます。

属性	説明
nodeNameOrLbl	パラメータの解決時に一致する FolderInst を検索します。FolderInst をパラメータ解決で使用するには、このプロパティはノード名と一致する必要があります。一致していない場合、FolderInst はスキップされ、値はこの FolderInst から使用されません。 このフィールドの値を [any] にして、この FolderInst がサービス グラフ内のすべてのノードで使用されるようにできます。

ParamInst

ParamInst はコンフィギュレーション パラメータの基本単位です。ParamInst は単一のコンフィギュレーション パラメータを定義します。FolderInst と同様に、デバイス仕様内には各 ParamInst に対応する MParam が存在する必要があります。APIC は仕様を検証して、パッケージ内に ParamInst に対応する MParam が存在することを確認します。ParamInst の値は、対応する MParam 内で指定される検証メソッドを使用して検証されます。ParamInst には、次の属性があります。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Value	特定のコンフィギュレーション アイテムの値を保持します。値は MParam ではサポートされません。

CfgRelInst

CfgRelInst には、次の属性があります。

属性	説明
Key	コンフィギュレーション アイテムのタイプを定義します。キーは、デバイス パッケージで定義されており、上書きすることはできません。キーは、検証だけでなく一致基準として使用されます。
Value	ターゲット FolderInst のパスを保持します。

コンフィギュレーション パラメータを持つアプリケーション EPG の XML POST の例

次の XML の例は、デバイス パッケージ内のコンフィギュレーション パラメータを示しています。

```
<fvAEPg dn="uni/tn-acme/ap-myApp/epg-app"  name="app">

   <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any" key="Monitor"

      name="monitor1">
       <vnsRsFolderInstToMFolder tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Monitor"/>

        <vnsParamInst name="weight" key="weight" value="10"/>
    </vnsFolderInst>

   <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any" key="Service"

      name="Service1">
        <vnsParamInst name="servicename" key="servicename" value="crpvgrtst02-8010"/>
        <vnsParamInst name="servicetype" key="servicetype" value="TCP"/>
        <vnsParamInst name="servername" key="servername" value="s192.168.100.100"/>
       <vnsParamInst name="serveripaddress" key="serveripaddress" value="192.168.100.100"/>

        <vnsParamInst name="serviceport" key="serviceport" value="8080"/>
        <vnsParamInst name="svrtimeout" key="svrtimeout" value="9000" />
        <vnsParamInst name="clttimeout" key="clttimeout" value="9000" />
        <vnsParamInst name="usip" key="usip" value="NO" />
        <vnsParamInst name="useproxyport" key="useproxyport" value="" />
        <vnsParamInst name="cip" key="cip" value="ENABLED" />
        <vnsParamInst name="cka" key="cka" value="NO" />
        <vnsParamInst name="sp" key="sp" value="OFF" />
        <vnsParamInst name="cmp" key="cmp" value="NO" />
        <vnsParamInst name="maxclient" key="maxclient" value="0" />
        <vnsParamInst name="maxreq" key="maxreq" value="0" />
        <vnsParamInst name="tcpb" key="tcpb" value="NO" />
        <vnsCfgRelInst name="MonitorConfig" key="MonitorConfig" targetName="monitor1"/>
    </vnsFolderInst>

   <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any" key="Network"

      name="Network">
       <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any" key="vip"

          name="vip">
            <vnsParamInst name="vipaddress1" key="vipaddress" value="10.10.10.200"/>
        </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any"
          devCtxLbl="C1" key="snip" name="snip1">
            <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.200"/>
        </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any"
          devCtxLbl="C2" key="snip" name="snip2">
   <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any" key="Network"

      name="Network">
       <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any" key="vip"

          name="vip">
            <vnsParamInst name="vipaddress1" key="vipaddress" value="10.10.10.100"/>
        </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any"
          devCtxLbl="C1" key="snip" name="snip1">
            <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.100"/>
        </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any"
          devCtxLbl="C2" key="snip" name="snip2">
```

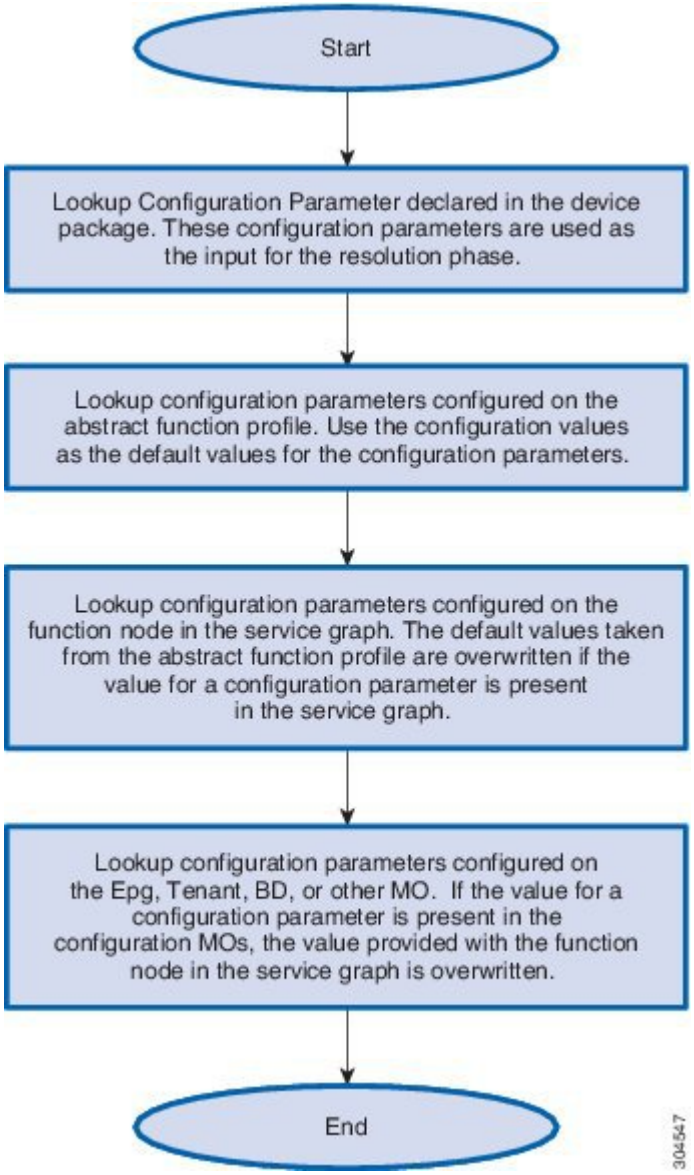
```
            <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.101"/>
          </vnsFolderInst>
          <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any"
            devCtxLbl="C3" key="snip" name="snip3">
            <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.102"/>
          </vnsFolderInst>
      </vnsFolderInst>

      <!-- SLB Configuration -->
     <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any" key="VServer"

        name="VServer">
          <!-- Virtual Server Configuration -->
          <vnsParamInst name="port" key="port" value="8010"/>
          <vnsParamInst name="vip" key="vip" value="10.10.10.100"/>
          <vnsParamInst name="vservername" key="vservername" value="crpvgrtst02-vip-8010"/>
          <vnsParamInst name="servicename" key="servicename" value="crpvgrtst02-8010"/>
          <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
            key="VServerGlobalConfig" name="VServerGlobalConfig">
             <vnsCfgRelInst name="ServiceConfig" key="ServiceConfig" targetName="Service1"/>

            <vnsCfgRelInst name="VipConfig" key="VipConfig" targetName="Network/vip"/>
          </vnsFolderInst>
      </vnsFolderInst>
</fvAEPg>
```

パラメータ解決

グラフ インスタンス作成時に、Application Policy Infrastructure Controller（APIC）はサービス グラフの各機能に対してコンフィギュレーション パラメータを解決します。解決が完了すると、パラメータ値がデバイス スクリプトに渡されます。デバイス スクリプトはこれらのパラメータ値を使用してサービス アプライアンス上でサービスを設定します。

次のフロー チャートは、パラメータの解決手順について説明しています。

図 *22* : パラメータ解決



パラメータ解決時の MO の検索

Application Policy Infrastructure Controller（APIC）は、コンフィギュレーション パラメータを取得する適切なコンフィギュレーション MO の検出に 2 つの主なコンストラクトを使用します。

RsScopeToTerm

機能ノードまたは AbsFolder に対する RsScopeToTerm 関係は、グラフに対するパラメータを持つコンフィギュレーション MO と接続されるサービス グラフの端末ノードを示します。APIC は、グラフ コンフィギュレーション パラメータを検出するために、RsScopeToTerm 内の指定の端末ノードと接続されたコンフィギュレーション MO を使用します。

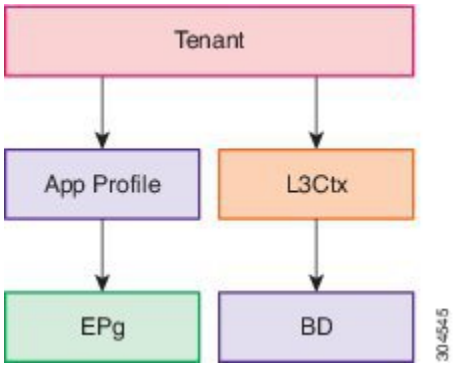
指定された RsScopeToTerm コンフィギュレーションがない場合、APIC はデフォルトでプロバイダー EPG に接続された端末を使用します。

ScopedBy 属性

ScopedBy 属性はパラメータの解決に使用する開始MOの検出に使用されます。たとえば、scopedByに「EPG」の値がある場合、APICはエンドポイント グループからパラメータ解決を開始します。APIC は、階層を上ってパラメータを解決し、アプリケーション プロファイルの次にテナントに上ってコンフィギュレーション パラメータを解決します。

次の図は、APIC MO の階層を示しています。

図 *23*： *APIC MO* の階層



ロールベースのアクセスコントロールルールの拡張について

マルチテナント環境でのレイヤ 4 ～ レイヤ 7 設定では、従来のロールベースのアクセス コントロール（RBAC）ドメインとロール モデルの定義を使用してテナント管理者が作成できない特定のオブジェクトを作成するには、管理者が介入する必要がありました。Application Policy Infrastructure Controller（APIC）では、オブジェクトの作成に必要な権限をテナント管理者に付与できるように、管理情報ツリー（MIT）でRBAC権限をより詳細に指定できます。また、テナント管理者は、管理者の介入なしに、セルフサービスを介してRBACルールを作成し、テナント サブツリーの下にあるリソースの権限をシステム内の他のテナントやユーザに付与することもできます。

ロールベースのアクセス コントロール ルールのアーキテクチャ

ロールベースのアクセス コントロール（RBAC）ルールには、ロールベースのアクセス コントロール（RBAC）モデルを強化して加筆性ルールを許可するブール型の `allowWrites`フィールドがあります。`allowWrites`フィールドがない場合に定義できるのは、読み取り RBAC ルールのみになります。

`RbacRule` クラスは次のように定義します。

```
Class aaa:RbacRule (CONCRETE)
Encrypted: false
Exportable: true
Persistent: true
Configurable: true
Write Access: [aaa, admin]
Read Access: [aaa, admin]
```

RBAC ルールにより、ユーザはセキュリティ ドメインから特定のオブジェクトで始まるサブツリーを読み取ることができます。

```
DN FORMAT:   [1] uni/rbacdb/rule-{[objectDn]}-dom-{domain}
```

表 *1*： *aaa:RbacRule* プロパティの概要

プロパティ	タイプ	クラス	説明
aaa:Boolean	scalar:Enum8	allowWrites（aaa:RbacRule:allowWrites）	読み取り/書き込みまたは読み取りルール。
naming:Name	string:Basic	domain（aaa:RbacRule:domain）	カウント オブジェクトのドメイン。`aaa:ARbacRule:domain` を無効にします。
reference:BinRef		objectDn（aaa:RbacRule:objectDn）	`aaa:ARbacRule:objectDn` を無効にします。

`PartialRbacRule` クラスは `fvTenant` クラスの下に定義され、テナントが RBAC ルール（セルフサービス）を作成できるようにします。`PartialRbacRule` クラスは次のように定義されます。

```
Class aaa:PartialRbacRule (CONCRETE)
Encrypted: false
Exportable: true
Persistent: true
Configurable: true
Write Access: [aaa, admin]
Read Access: [aaa, admin]
```

表 *2*： *aaa:PartialRbacRule* プロパティの概要

プロパティ	タイプ	クラス	説明
aaa:Boolean	scalar:Enum8	allowWrites（aaa:PartialRbacRule:allowWrites）	読み取り/書き込みまたは読み取りルール。

プロパティ	タイプ	クラス	説明
naming:Name	string:Basic	domain（aaa:PartialRbacRule:domain）	カウント オブジェクトのドメイン。
reference:BinRef		monPolDn（aaa:PartialRbacRule:monPolDn）	この監視可能なオブジェクトにアタッチするモニタリング ポリシー。
reference:BinRef		partialObjectDn（aaa:PartialRbacRule:partialObjectDn）	

テナントによる `PartialRbacRule` クラスの作成では、`partialObjectDn` の正当性を確認する必要があります。`partialObjectDn` がテナント サブツリーの下にあれば有効です。親テナント サブツリー外の識別名は許可されていません。

管理者は、システム内の識別名を指す `RbacRule` を作成できます。テナント管理者が作成できるのは、テナント管理者のテナント サブツリー内にある識別名を指す `PartialRbacRule` のみです。

ロールベース アクセス コントロール ルールのシステム フロー

レイヤ 4 ～ レイヤ 7 ポリシーの設定前、設定中、または設定後に、テナント管理者は、特定のファイアウォールとロード バランサ デバイスへのアクセス権を自分のテナント ユーザに付与する `PartialRbacRule` の作成を選択することができます。各リソース グループを表す `aaaDomain` を作成し、個々に割り当てることでアクセスが実現します。次にセットアップの例を示します。

テナント	Acme
ユーザ	acme-admin acme-firewall-1-admin acme-firewall-2-admin acme-loadbalancer-1-admin acme-loadbalancer-2-admin
ファイアウォール デバイス	Firewall1 Firewall2
ロード バランサ デバイス	LB1 LB2

テナント管理者ユーザの acme-admin は、デバイスの Firewall1、Firewall2、LB1、および LB2 を作成したいと考えています。各デバイスに対する完全な書き込みアクセス許可をユーザごとに割り

当てる必要があります。たとえば、ユーザ acme-firewall-1-admin には デバイス Firewall1 ポリシーへの書き込み権限のみが必要ですが、ユーザ acme-loadbalancer-1-admin にはデバイス LB1 ポリシーへの書き込み権限のみが必要です。これを実現するには、acme-admin ユーザが、次のアクセス権を付与する 4 つの `PartialRbacRule` を作成する必要があります。

- Firewall1 識別名：ドメイン acme-firewall1 による書き込みが可能
- Firewall2 識別名：ドメイン acme-firewall2 による書き込みが可能
- LB1 識別名：ドメイン acme-lb1 による書き込みが可能
- LB2 識別名：ドメイン acme-lb2 による書き込みが可能

ユーザには次の権限が割り当てられます。

- ユーザ：acme-firewall-1-admin
  - ドメイン acme：read-all 権限
  - ドメイン acme-firewall1：テナント管理/書き込み
- ユーザ：acme-firewall-2-admin
  - ドメイン acme：read-all 権限
  - ドメイン acme-firewall2：テナント管理/書き込み
- ユーザ：acme-lb-1-admin
  - ドメイン acme：read-all 権限
  - ドメイン acme-lb1：テナント管理/書き込み
- ユーザ：acme-lb-2-admin
  - ドメイン acme：read-all 権限
  - ドメイン acme-lb2：テナント管理/書き込み

上記 4 人のユーザのいずれも、ドメイン acme の権限によって、acme テナント サブツリーを読み取れますが、どのノードにも書き込めません。テナント acme-lb2 のテナント管理/書き込みの権限によって、ユーザは LB2 ポリシー サブツリーのみに書き込むことができます。

第 12 章

サービス グラフ テンプレートの使用

GUI を使用したサービス グラフ テンプレートとコントラクトおよび EPG の関連付け

GUIを使用して、サービスグラフテンプレートをコントラクトとエンドポイントグループ（EPG）に関連付ける必要があります。

（注） サービス グラフ テンプレートをコントラクトと EPG に関連付けるには、GUI のみを使用できます。

サービス グラフ テンプレートをコントラクトと EPG に関連付ける手順については、GUI の使用方法，（153 ページ） を参照してください。

NX-OS スタイルの CLI を使用したサービス グラフ テンプレートの作成

次に、サービス グラフ テンプレートの作成手順を示します。

ステップ 1 コンフィギュレーション モードを開始します。

例：
```
apic1# configure
```

ステップ 2 テナントのコンフィギュレーション モードを開始します。
```
tenant tenant_name
```

例：
```
apic1(config)# tenant t1
```

ステップ 3 サービス グラフをテンプレートに関連付けます。
```
l417 graph graph_name contract contract_name
```

パラメータ	説明
graph	サービス グラフ テンプレートの名前。
contract	サービス グラフ テンプレートで使用するコントラクトの名前。

例：
```
apic1(config-tenant)# l417 graph GraphL3asa contract ContractL3ASA
```

ステップ 4 機能ノードを追加します。
```
service node_name [device-cluster-tenant tenant_name] [device-cluster device_name] [mode
deployment_mode]
```

パラメータ	説明
service	追加するサービス ノードの名前。
device-cluster-tenant	デバイス クラスタのインポート元のテナント。グラフを設定するテナントと同じテナントにデバイス クラスタがない場合にのみ、このパラメータを指定します。
device-cluster	このサービス ノードに使用するデバイス クラスタの名前。

パラメータ	説明
mode	導入モード。値は次のとおりです。 • ADC_ONE_ARM：ワンアーム モードを指定します。 • ADC_TWO_ARM：ツーアーム モードを指定します。 • FW_ROUTED：ルーテッド（GoTo）モードを指定します。 • FW_TRANS：トランスペアレント（GoThrough）モードを指定します。 • OTHERS モードを指定しないと、導入モードは使用されません。

例：
```
apic1(config-graph)# service Node1 device-cluster-tenant common device-cluster ifav108-asa-2 mode
 FW_ROUTED
```

ステップ 5 コンシューマ コネクタを追加します。
```
connector connector_type [cluster-interface interface_type]
```

パラメータ	説明
connector	サービス グラフ内のコネクタのタイプ。値は次のとおりです。 • provider • consumer
cluster-interface	デバイス クラスタ インターフェイスのタイプ。値は次のとおりです。 • provider • consumer テナント Common 内のサービス グラフ テンプレートの場合は、このパラメータを指定しないでください。

例：
```
apic1(config-service)# connector consumer cluster-interface consumer
```

ステップ 6 テナントをコネクタに関連付け、コネクタ コンフィギュレーション モードを終了します。
```
l4l7-peer tenant tenant_name out L3OutExternal epg epg_name
  redistribute redistribute_property
exit
```

パラメータ	説明
tenant	コネクタに関連付けるテナントの名前。
out	レイヤ 3 Outside の名前。
epg	エンドポイント グループの名前。
redistribute	再配布プロトコルのプロパティ。

例：
```
apic1(config-connector)# l4l7-peer tenant t1 out L3OutExternal epg L3ExtNet
  redistribute connected,ospf
apic1(config-connector)# exit
```

ステップ 7 プロバイダーに対してステップ 5 と 6 を繰り返します。

例：
```
apic1(config-service)# connector provider cluster-interface provider
apic1(config-connector)# l4l7-peer tenant t1 out L3OutInternal epg L3IntNet
  redistribute connected,ospf
apic1(config-connector)# exit
```

ステップ 8 （任意） ルータを追加し、ノード コンフィギュレーション モードを終了します。
```
rtr-cfg router_ID
exit
```

パラメータ	説明
rtr-cfg	ルータの ID。

テナント Common でサービス グラフ テンプレートを作成する場合は、この手順をスキップします。

例：
```
apic1(config-service)# rtr-cfg router-id1
apic1(config-service)# exit
```

ステップ 9 1 つの接続をコンシューマ コネクタに、もう 1 つをプロバイダー コネクタに関連付けてから、サービス グラフ コンフィギュレーション モードを終了します。
```
connection connection_name terminal terminal_type service node_name
  connector connector_type
exit
```

パラメータ	説明
connection	コネクタに関連付ける接続の名前。

パラメータ	説明
terminal	端末のタイプ。値は次のとおりです。 • provider • consumer
service	サービス グラフのノードの名前。
connector	コネクタのタイプ。値は次のとおりです。 • provider • consumer

例：

```
apic1(config-graph)# connection C1 terminal consumer service Node1 connector consumer
apic1(config-graph)# connection C2 terminal provider service Node1 connector provider
apic1(config-graph)# exit
```

ステップ 10 コンフィギュレーション モードを終了します。

例：

```
apic1(config-tenant)# exit
apic1(config)# exit
```

オブジェクト モデルの CLI を使用したサービス グラフ テンプレートの設定

オブジェクト モデルの GUI を使用して、サービス グラフ テンプレートを設定することができます。

次に、設定の例を示します。

```
admin@apic1:contracts> moconfig running

# contract
cd '/aci/tenants/acme/security-policies/contracts'
mocreate 'webCtrct'
cd 'webCtrct'
moset scope 'tenant'
moconfig commit

# contract-subject
cd '/aci/tenants/acme/security-policies/contracts/webCtrct/subjects'
```

```
mocreate 'http'
moconfig commit

# subj-graphatt
cd '/aci/tenants/acme/security-policies/contracts/webCtrct/subjects/http'
mocreate subj-graphatt
cd 'subj-graphatt'
moset name 'WebGraph'
moconfig commit

# vz-rssubjfiltatt
cd '/aci/tenants/acme/security-policies/contracts/webCtrct/subjects/http/common-filters'
mocreate 'wildcard'
moconfig commit
```

REST API を使用したサービス グラフ テンプレートの設定

次の REST API を使用してサービス グラフ テンプレートを設定できます。

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
      <!—L3 Network-->
      <fvCtx name="MyNetwork"/>
        <!-- Bridge Domain for MySrvr EPG -->
        <fvBD name="MySrvrBD">
           <fvRsCtx tnFvCtxName="MyNetwork" />
           <fvSubnet ip="10.10.10.10/24">
           </fvSubnet>
        </fvBD>
        <!-- Bridge Domain for MyClnt EPG -->
        <fvBD name="MyClntBD">
          <fvRsCtx tnFvCtxName="MyNetwork" />
          <fvSubnet ip="20.20.20.20/24">
          </fvSubnet>
        </fvBD>
        <fvAp dn="uni/tn-acme/ap-MyAP" name="MyAP">
            <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MyClnt" name="MyClnt">
                <fvRsBd tnFvBDName="MySrvrBD" />
                <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs" />
                <fvRsProv tnVzBrCPName="webCtrct">
                </fvRsProv>
              <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]" encap="vlan-202"/>

              <fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]" encap="vlan-202"/>

            </fvAEPg>
            <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MySRVR" name="MySRVR">
                <fvRsBd tnFvBDName="MyClntBD" />
                <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs" />
                <fvRsCons tnVzBrCPName="webCtrct">
                </fvRsCons>
              <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]" encap="vlan-203"/>

              <fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]" encap="vlan-203"/>

            </fvAEPg>
        </fvAp>
    </fvTenant>
</polUni>
```

REST API を使用したセキュリティ ポリシーの作成

次の REST API を使用してセキュリティ ポリシーを作成することができます。

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vzFilter name="HttpIn">
            <vzEntry name="e1" prot="6" dToPort="80"/>
```

```
        </vzFilter>
        <vzBrCP name="webCtrct">
            <vzSubj name="http">
                <vzRsSubjFiltAtt tnVzFilterName="HttpIn"/>
            </vzSubj>
        </vzBrCP>
    </fvTenant>
</polUni>
```

第 13 章

サービス グラフのモニタリング

GUI を使用したサービス グラフ インスタンスのモニタリング

サービス グラフ テンプレートを設定し、エンドポイント グループ（EPG）およびコントラクトにグラフをアタッチした後は、サービス グラフ インスタンスをモニタできます。モニタリングには、グラフ インスタンスの状態、グラフ インスタンスの機能、機能に割り当てられたリソース、および機能に指定されたパラメータの表示が含まれます。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、サービス グラフをモニタするテナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、*[tenant_name]* > [L4-L7 Services] > [Deployed Graph Instancess] の順に選択します。
[Work] ペインは、アクティブなサービス グラフ インスタンスに関する次の情報を表示します。

名前	説明
[Service Graph] カラム	サービス グラフ テンプレートの名前。

名前	説明
[Contract] カラム	サービス グラフ テンプレートに表示されるコントラクトの名前。
[Contained By] カラム	サービス グラフ テンプレートを含むネットワークの名前。
[State] カラム	サービス グラフ テンプレートの状態。[applied] の状態は、グラフが適用され、グラフポリシーがファブリックおよびサービス デバイス内でアクティブであることを意味します。
[Description] カラム	サービス グラフの説明

ステップ 4 [Deployed Service Graphs] ブランチを展開します。アクティブなサービス グラフ インスタンスがブランチの下にリストされます。

ステップ 5 サービス グラフ インスタンスをクリックして、[Work] ペインにそのインスタンスに関する追加情報を表示します。デフォルト ビューはグラフのトポロジです。[Work] ペインのタブのいずれかをクリックして、そのグラフのビューを変更できます。

ステップ 6 グラフ インスタンスのいずれかのブランチを展開します。グラフ インスタンスの機能は、インスタンスの下に表示されます。

ステップ 7 機能をクリックして、[Work] ペインにその機能に関する追加情報を表示します。デフォルト ビューはその機能のポリシーです。[Work] ペインのタブのいずれかをクリックして、その機能のビューを変更できます。[Work] ペインには、ポリシーに関する次の情報が表示されます。

名前	説明
[POLICY] タブ	機能のプロパティ、機能に割り当てられたリソース、および機能のパラメータ。
[FAULTS] タブ	機能ノードで生じている問題。
[HISTORY] タブ	機能ノードで発生したイベントの履歴。

ステップ 8 [Navigation] ペインで、[Deployed Device] をクリックします。[Work] ペインにデバイスのインスタンスに関する情報が表示されます。

GUI を使用したサービス グラフ エラーのモニタリング

サービス グラフテンプレートを設定し、エンドポイント グループ（EPG）およびコントラクトにグラフをアタッチした後は、サービス グラフ テンプレートのエラーをモニタできます。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。
ステップ 2 [Work] ペインで、サービス グラフをモニタするテナントの名前をダブルクリックします。
ステップ 3 [Navigation] ペインで、*[tenant_name]* > [L4-L7 Services] > [Deployed Graph Instancess] の順に展開します。
ステップ 4 エラーを表示するグラフ インスタンスのブランチを展開します。グラフ インスタンスの機能は、インスタンスの下に表示されます。
ステップ 5 機能のいずれかをクリックします。デフォルトで、[Work] ペインはその機能のポリシーを示します。
ステップ 6 [Work] ペインの [FAULTS] タブをクリックします。[Work] ペインが機能ノードのエラーを表示します。

サービス グラフ エラーの解決

1 つ以上のサービス グラフ テンプレート エラーを発見した場合、問題の解決はエラーによって異なります。次の表は、エラーの説明とエラーを解決する方法を説明しています。

表 *3*：コネクタのエラー

エラー	CLI ラベル	説明と解決法
missing-connection	connection associated with a connector not found	グラフ コネクタの設定が無効です。コネクタに関連付けられた接続が見つかりませんでした。
missing-nodeinst	NodeInst associated with a connector not found	グラフ コネクタの設定が無効です。コネクタに関連付けられた NodeInst が見つかりませんでした。
conn-nonrenderable	Graph connector could not be rendered.	グラフ コネクタの設定が無効です。グラフをレンダリングできませんでした。
invalid-bd	BD associated with a connector is not valid	グラフ コネクタの設定が無効です。コネクタの関連ブリッジドメインが無効です。

エラー	CLI ラベル	説明と解決法
invalid-ctx	Ctx associated with a connector is not valid.	グラフ コネクタの設定が無効です。コネクタの関連する Ctx が無効です。
missing-peer-conn	Peer connector associated with a connector not found.	グラフ コネクタの設定が無効です。接続のピア コネクタが見つかりませんでした。

表 *4* : ***AbsGraph*** および ***GraphInst*** エラー

エラー	CLI ラベル	説明と解決法
invalid-abstract-graph-config	invalid abstract graph config	抽象グラフ設定が無効です。
missing-mandatory-param	mandatory param not found	必要な設定パラメータが解決できませんでした。パッケージの必須パラメータをチェックし、AbsGraph にパラメータがあることを確認します。
param-cardinality-error	invalid param cardinality	コンフィギュレーション パラメータは、濃度の要件を満たしていません。cardinality=n を指定しないでパラメータの複数のインスタンスが指定されているかどうかを確認します。
epp-download-failure	epp download failure	グラフ ポリシーがスイッチのダウンロードに失敗しました。
param-duplicate-name-failure	duplicate param name	同じ名前のパラメータの複数の同一コピーが検出されました。
id-allocation-failure	id allocation failure	一意のネットワーク リソース（VLAN VXLAN）を割り当てることができませんでした。
missing-ldev	No cluster found	クラスタが見つかりませんでした。
context-cardinality-violation-failure	invalid cluster context cardinality	クラスタは必要なテナント機能（マルチテナントまたはシングル テナント）をサポートしていません。

エラー	CLI ラベル	説明と解決法
function-type-mismatch-failure	invalid function type	機能タイプが選択したデバイスでサポートされていません。AbsNode機能タイプと解決された LDevVip 機能タイプが一致するか確認します。
invalid-abstract-graph-config-param	invalid abstract graph config param	抽象グラフ コンフィギュレーションパラメータが無効です。
missing-mparam	No parameter definition found	必要なパラメータ定義が見つかりませんでした。
missing-abs-graph	no abs graph found	抽象グラフ設定がグラフ インスタンスにありません。
invalid-param-config	invalid param config	パラメータ設定が無効です。
invalid-param-scope	invalid parameter scope	パラメータ スコープが無効です。AbsGraph の vnsRsScopeToTerm パラメータが正しいかどうか確認します。
invalid-ldev	Invalid cluster	クラスタ設定が無効です。解決した LDevVip のステータスを確認して、エラーを解決します。
missing-tenant	no tenant found	グラフに対してテナントが見つかりませんでした。
internal-error	internal error	内部エラーがグラフ処理中に発生しました。
resource-allocation-failure	resource allocation failure	グラフ処理中に必要なリソースを割り当てることができませんでした。
missing-abs-function	no abstract function found	抽象機能の定義が見つかりません。
param-validation-failed	param validation failure	コンフィギュレーション パラメータ値が無効です。
missing-mconn	No connector found	必要なコネクタが見つかりませんでした。

エラー	CLI ラベル	説明と解決法
cdev-missing-mgmt-ip	no mgmt ip found for cdev	具象デバイスに対して管理 IP アドレスが見つかりませんでした。vnsCMgmt が解決する vnsCDev に存在するかどうかを確認します。
invalid-graphinst	invalid graphinst config	グラフ インスタンスが無効です。
missing-interface	no interface found	インターフェイスが見つかりませんでした。
missing-bd	no bd found	ブリッジ ドメインが見つかりませんでした。
missing-terminal	Terminal node is missing a terminal	端末ノードに端末がありません。端末ノードの設定を確認してください。
missing-namespace	no vlan/vxlan namespace found	VLAN または VXLAN の割り当てに必要なネームスペースが見つかりません。解決された fvnsVlanInstp と関係がある phyDomp パラメータまたは vmmDomp パラメータが解決された vnsLDevVip に設定されていることを確認します。
missing-mfunc	No function found in device package	デバイス パッケージで必要な機能が見つかりません。パッケージ内にすべての AbsNode 機能タイプがあることを確認します。
missing-lif	no cluster interface found	必要なクラスタ インターフェイスが見つかりませんでした。vnsLDevVip の vnsLIf パラメータが正しく設定されていることを確認します。
invalid-absfunc-profile	Abstract Function Profile config is invalid	抽象機能のプロファイル設定が無効です。このエラーは、プロファイルで指定されている無効なコンフィギュレーション パラメータが要因として考えられます。

エラー	CLI ラベル	説明と解決法
missing-cdev	No device found	具象デバイスがクラスタ内に見つかりませんでした。有効な vnsCDev が解決された vnsLDevVip の下に存在することを確認してください。
inappropriate-devfolder	Illegal folder in configuration	対応するフォルダがデバイス パッケージで見つかりませんでした。
invalid-devctx	Device context is not legal for this folder	デバイス パッケージではこのフォルダにデバイス コンテキストを指定することはできません。
insufficient-devctx	Folder must have one value for each associated CDev	フォルダは具象デバイスに固有です。フォルダは、各具象デバイスに対して少なくとも 1 つの値を持つ必要があります。
cdev-missing-cif	No interface defined	具象デバイスには少なくとも 1 つのインターフェイスを定義する必要があります。
cdev-missing-pathinfo	Missing path for interface	物理サービス アプライアンスでは、インターフェイスがどのリーフ ポートに接続されているかを把握する必要があります。vnsCifPathAtt パラメータが、解決された vnsCDev の下のすべての vnsCIf に存在することを確認します。
missing-cif	Device interfaces does not match cluster	デバイスインターフェイスは、クラスタに設定されているインターフェイスに一致させる必要があります。vnsCIf パラメータおよび vnsLIf パラメータが、解決された vnsLDevVip の下に存在することを確認します。
ldevvip-missing-mgmt-ip	No Mgmt ip found for LDevVip	LDevVip に対して管理 IP アドレスが見つかりませんでした。
lif-invalid-MIf	LIf has an invalid MifLbl	LIf に含まれる MIfLbl がデバイス パッケージに存在しません。

エラー	CLI ラベル	説明と解決法
lif-invalid-CIf	LIf has an invalid CIf	LIf に含まれる CIf がありません。具象デバイスおよびCIfの設定を確認します。
missing-function-node	Abstract graph missing function node	抽象グラフには、少なくとも 1 つの機能ノードが存在する必要があります。
graph-loop-detected	Abstract graph config has a loop	抽象グラフ設定が無効です。設定にループがあります。
gothrough-routing-enabled-both	Both the legs of go through node has routing enabled	通過ノードの両方のレッグでルーティングが有効になっています。
invalid-terminal-nodes	Abstract graph has invalid number of terminal nodes	抽象グラフは少なくとも 2 つの端末ノードを持つ必要があります。
missing-ldev-ctx	No device context found for LDev	デバイスのデバイス コンテキストが見つかりませんでした。vnsLDevCtx にコントラクト、グラフおよびノードに一致する値があることを確認します。
arp-flood-enabled	ARP flood is enabled on the management end point group	ARP フラッディングは管理エンドポイントのグループに対して無効です。
folderinst-validation-failed	FolderInst has key, that is not found in MFolder	FolderInst のキーおよび値は MFolder 仕様を尊重する必要があります。
paraminst-validation-failed	ParamInst has key and/or value, that are not found in MParam	ParamInst のキーおよび値は MParam 仕様を尊重する必要があります。
invalid-mfolder	FolderInst points to an invalid MFolder	FolderInst は有効な MFolder をポイントする必要があります。
invalid-mparam	ParamInst points to an invalid MParam	ParamInst は有効な MParam をポイントする必要があります。
devfolder-validation-failed	DevFolder has key, that is not found in MFolder	DevFolders のキーおよび値は MFolder 仕様を尊重する必要があります。

エラー	CLI ラベル	説明と解決法
devparam-validation-failed	DevParam has key and/or value, that are not found in MParam	DevParam のキーおよび値は MParam 仕様を尊重する必要があります。
cdev-missing-virtual-info	Virtual Object Info is missing in CDev	LDevVip のタイプが Virtual の場合は仮想オブジェクト情報を指定する必要があります。
invalid-rsmconnatt	Relationship to metaconnector is invalid	メタコネクタの DN を修正し、正しい MDev 階層にバインドすることを確認します。

GUI を使用した仮想デバイスのモニタリング

サービス グラフ テンプレートを設定し、エンドポイント グループ（EPG）およびコントラクトにグラフをアタッチした後は、テナントの仮想デバイスをモニタできます。仮想デバイスをモニタリングすると、どのデバイスが使用中か、どの VLAN がデバイス用に設定されているかや、デバイスに渡されるパラメータ、デバイスの統計、およびデバイスの健全性を確認できます。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、サービス グラフをモニタするテナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、*[tenant_name]* > [L4-L7 Services] > [Deployed Devices] の順に展開します。

ステップ 4 導入されたデバイスのいずれかをクリックします。デフォルトでは、[Work] ペインに導入済みのデバイスのポリシーが表示されます。ビューを変更するには、[Work] ペインのタブをクリックします。タブは、仮想デバイスに関する以下の情報を表示します。

タブ	説明
[POLICY] タブ	使用中のデバイス、デバイス内で設定された VLAN、およびデバイスに渡されたパラメータ。
[OPERATIONAL] タブ	さまざまなデバイスから受信する統計情報。
[HEALTH] タブ	デバイスの状態。

NX-OSスタイルのCLIを使用したデバイスクラスタとサービス グラフ ステータスのモニタリング

この項のコマンドで、NX-OS スタイルの CLI を使用してデバイス クラスタとサービス グラフ ステータスをモニタする例を示します。

デバイス クラスタの動作情報の表示

次に、デバイス クラスタの動作情報を表示するコマンドを示します。

```
show l4l7-cluster tenant tenant_name cluster device_cluster_name
```

例：

```
apic1# show l4l7-cluster tenant HA_Tenant1 cluster Firewall
tenant-graph : HA_Tenant1-g2,HA_Tenant1-g1


Device Cluster    : Firewall
Cluster Interface : consumer1
Encap             : vlan-501
Pctag             : 32773
Devices           : FW2(int),FW1(int)
Graphs            : HA_Tenant1-g1
Contracts         : HA_Tenant1-c1


Device Cluster    : Firewall
Cluster Interface : provider1
Encap             : vlan-502
Pctag             : 32774
Devices           : FW2(ext),FW1(ext)
Graphs            : HA_Tenant1-g1
Contracts         : HA_Tenant1-c1
```

デバイス クラスタの動作ステータスの表示

次に、デバイス クラスタの動作ステータスを表示するコマンドを示します。

```
apic1# show l4l7-graph tenant tenant_name [graph graph_name]
```

例：

次に、HA_Tenant1 テナントのステータスの高レベル出力を提供する例を示します。

```
apic1# show l4l7-graph tenant HA_Tenant1
Graph           : g1
Total Instances : 1
Encaps Used     : vlan-501,vlan-502,vlan-503,vlan-504
Device Used     : uni/tn-HA_Tenant1/lDevVip-Firewall

Graph           : g2
Total Instances : 1
Encaps Used     : vlan-501,vlan-502,vlan-503,vlan-504
Device Used     : uni/tn-HA_Tenant1/lDevVip-Firewall
```

次に、HA_Tenant1 に関連付けられた g1 サービス グラフの詳細出力を提供する例を示します。

```
apic1# show l4l7-graph tenant HA_Tenant1 graph g1
Graph           : HA_Tenant1-g1
Graph Instances : 1

Consumer EPg  : HA_Tenant1-consEPG1
Provider EPg  : HA_Tenant1-provEPG1
```

```
Contract Name : HA_Tenant1-c1
Config status : applied

Function Node Name : Node1
 Connector   Encap       Bridge-Domain  Device Interface
 ----------  ----------  ----------     --------------------
 consumer    vlan-3001   provBD1        consumer
 provider    vlan-3335   consBD1        provider
```

デバイス クラスタのエラーの表示

次に、デバイス クラスタのエラーを表示するコマンドを示します。

```
show faults l4l7-cluster
```

例：

```
apic1# show faults l4l7-cluster
Code             : F0772
Severity         : minor
Last Transition : 2015-09-01T01:41:13.767+00:00
Lifecycle        : soaking-clearing
Affected object : uni/tn-ts1/lDevVip-d1/lIf-ext/fault-F0772
Description      : LIf configuration ext for L4-L7 Devices d1 for tenant ts1
                   is invalid.

Code             : F1085
Severity         : cleared
Last Transition : 2015-09-01T01:39:04.696+00:00
Lifecycle        : retaining
Affected object : uni/tn-ts1/lDevVip-d1/rsmDevAtt/fault-F1085
Description      : Failed to form relation to MO uni/infra/mDev-CiscoInternal-
                   NetworkOnly-1.0 of class vnsMDev

Code             : F1690
Severity         : minor
Last Transition : 2015-09-01T01:39:04.676+00:00
Lifecycle        : soaking
Affected object : uni/tn-ts1/lDevVip-d1/vnsConfIssue-missing-
                   namespace/fault-F1690
Description      : Configuration is invalid due to no vlan/vxlan namespace
                   found
```

サービス グラフのエラーの表示

次に、サービス グラフのエラーを表示するコマンドを示します。

```
show faults l4l7-graph
```

例：

```
apic1# show faults l4l7-graph
Code             : F1690
Severity         : minor
Last Transition : 2015-11-25T20:07:33.635+00:00
Lifecycle        : raised
DN               : uni/tn-HA_Tenant1/AbsGraph-WebGraph/vnsConfIssue-invalid-
                   abstract-graph-config-param/fault-F1690
Description      : Configuration is invalid due to invalid abstract graph
                   config param
```

デバイス クラスタの実行コンフィギュレーションの表示

次に、デバイス クラスタの実行コンフィギュレーションを表示するコマンドを示します。

```
show running-config tenant tenant_name l4l7 cluster
```

例：

```
apic1# show running-config tenant common l4l7 cluster
# Command: show running-config tenant common l4l7 cluster
# Time: Thu Nov 26 00:35:59 2015
  tenant common
    l4l7 cluster name ifav108-asa type physical vlan-domain phyDom5 service FW function
go-through
      cluster-device C1
      cluster-interface consumer_1
        member device C1 device-interface port-channel1
          interface vpc VPCPolASA leaf 103 104
          exit
        exit
      cluster-interface provider_1
        member device C1 device-interface port-channel1
          interface vpc VPCPolASA leaf 103 104
          exit
        exit
      exit
```

サービス グラフの実行コンフィギュレーションの表示

次に、サービス グラフの実行コンフィギュレーションを表示するコマンドを示します。

```
show running-config tenant tenant_name l4l7 graph
```

例：

```
apic1# show running-config tenant common l4l7 graph
# Command: show running-config tenant common l4l7 graph
# Time: Thu Nov 26 00:35:59 2015
  tenant T1
    l4l7 graph Graph-Citrix contract Contract-Citrix
     service N1 device-cluster-tenant common device-cluster ifav108-citrix mode ADC_ONE_ARM

        connector provider cluster-interface pro
          bridge-domain tenant common name BD4-Common
          exit
        connector consumer cluster-interface pro
          bridge-domain tenant common name BD4-Common
          exit
        exit
      connection C1 terminal consumer service N1 connector consumer
      connection C2 terminal provider service N1 connector provider
      exit
```

オブジェクト モデルの CLI を使用したデバイス クラスタとサービス グラフ ステータスのモニタリング

この項のコマンドで、オブジェクト モデルの CLI を使用してデバイス クラスタとサービス グラフ ステータスをモニタする例を示します。

すべてのデバイス クラスタのステータスの表示

次に、すべてのデバイス クラスタのステータスを表示するコマンドを示します。

```
apic1# show l4l7-cluster
Device Cluster        : InsiemeCluster
Total Graph Instances : 2
Active Graphs         : G2,G1
Encaps Used           : vlan-202,vlan-268
```

指定したテナント内のすべてのデバイス クラスタのステータスの表示

次に、テナント tnt1 内のすべてのデバイス クラスタのステータスを表示するコマンドを示します。

```
apic1# show l4l7-cluster tenant tnt1
Device Cluster        : InsiemeCluster
Total Graph Instances : 2
Active Graphs         : G2,G1
Encaps Used           : vlan-202,vlan-268
```

指定したテナント内の指定したデバイス クラスタのステータスの表示

次に、テナント tnt1 内のデバイス クラスタ InsiemeCluster のステータスを表示するコマンドを示します。

```
apic1# show l4l7-cluster tenant tnt1 cluster InsiemeCluster
tenant-graph : G1,G2

Device Cluster    : InsiemeCluster
Cluster Interface : outside
Encap             : vlan-202
Pctag             : 49155
Devices           : Generic1(int),Generic2(int)
Graphs            : G1,G2
Contracts         : webCtrct

Device Cluster    : InsiemeCluster
Cluster Interface : inside
Encap             : vlan-237
Pctag             : 32772
Devices           : Generic2(ext),Generic1(ext)
Graphs            : G1,G2
Contracts         : webCtrct
```

すべてのサービス グラフのステータスの表示

次に、すべてのサービス グラフのステータスを表示するコマンドを示します。

```
apic1# show l4l7-graph
Graph           : G2
Total Instances : 1
Encaps Used     : vlan-202,vlan-268
Device Used     : uni/tn-tnt1/lDevVip-InsiemeCluster

Graph           : G1
Total Instances : 1
Encaps Used     : vlan-202,vlan-268
Device Used     : uni/tn-tnt1/lDevVip-InsiemeCluster
```

指定したサービス グラフのステータスの表示

次に、サービス グラフ G1 のステータスを表示するコマンドを示します。

```
apic1# show l4l7-graph graph G1
Graph           : G1
Total Instances : 1

Consumer EPg  : uni/tn-tnt1/ap-sap/epg-app
Provider EPg  : uni/tn-tnt1/ap-sap/epg-web
Contract Name : uni/tn-tnt1/brc-webCtrct

Function Connector : outside
Encap              : vlan-202
BD                 : tnt1BD2
Device             : uni/tn-tnt1/lDevVip-InsiemeCluster
```

```
Function Connector : inside
Encap              : vlan-268
BD                 : tnt1BD1
Device             : uni/tn-tnt1/lDevVip-InsiemeCluster
```

指定したテナントのすべてのサービス グラフのステータスの表示

次に、テナント tnt1 のすべてのサービス グラフのステータスを表示するコマンドを示します。

```
apic1# show l4l7-graph tenant tnt1
Graph           : G2
Total Instances : 1
Encaps Used     : vlan-202,vlan-268
Device Used     : uni/tn-tnt1/lDevVip-InsiemeCluster

Graph           : G1
Total Instances : 1
Encaps Used     : vlan-202,vlan-268
Device Used     : uni/tn-tnt1/lDevVip-InsiemeCluster
```

指定したテナントの指定したサービス グラフのステータスの表示

次に、テナント tnt1 のサービス グラフ G2 のステータスを表示するコマンドを示します。

```
apic1# show l4l7-graph tenant tnt1 graph G2
Graph           : G2
Total Instances : 1

Consumer EPg  : uni/tn-tnt1/ap-sap/epg-app
Provider EPg  : uni/tn-tnt1/ap-sap/epg-web
Contract Name : uni/tn-tnt1/brc-webCtrct

Function Connector : outside
Encap              : vlan-202
BD                 : tnt1BD2
Device             : uni/tn-tnt1/lDevVip-InsiemeCluster

Function Connector : inside
Encap              : vlan-268
BD                 : tnt1BD1
Device             : uni/tn-tnt1/lDevVip-InsiemeCluster
```

第 14 章

サービス コンフィギュレーションの管理に対する管理ロールの設定

権限について

Application Policy Infrastructure Controller（APIC）で設定したロールに権限を付与できます。権限は、ロールが実行できるタスクを決定します。管理者ロールには次の特権を付与できます。

特権	説明
nw-svc-policy	ネットワーク サービス ポリシー権限では次を実行できます。 • サービス グラフ テンプレートの作成 • アプリケーション エンドポイント グループ（EPG）およびコントラクトへのサービス グラフ テンプレートのアタッチ • サービス グラフのモニタ

特権	説明
nw-svc-device	ネットワーク サービス デバイス権限では次を実行できます。 • デバイスの作成 • 具象デバイスの作成 • デバイス コンテキストの作成

（注） インフラストラクチャ管理者のみがデバイス パッケージを APIC にアップロードできます。

デバイス管理のロールの設定

デバイスを管理するためのロールを有効化するには、そのロールに次の特権を付与する必要があります。

- nw-svc-device

サービス グラフ テンプレート管理のロールの設定

サービス グラフ テンプレートを管理するためのロールを有効化するには、そのロールに次の特権を付与する必要があります。

- nw-svc-policy

デバイス パッケージのアップロードのロールの設定

デバイス パッケージは、APIC インフラ管理者特権でのみアップロードできます。インフラ管理者はデバイス パッケージをアップロードします。他のすべてのテナント管理者はデバイス パッケージに対して読み取り専用アクセスを持ちます。テナント管理者は、デバイス パッケージで使用可能なさまざまな機能にアクセスできます。

デバイスをエクスポートするためのロールの設定

デバイスをエクスポートして、テナント間でデバイスを共有することができます。**nw-device** ロールを持つテナントはデバイスを作成できます。デバイスを所有するテナントがこれらを別のテナントと共有したい場合、共有には **nw-svc-devshare** 特権が必要です。

nw-svc-devshare 特権を使用すると、テナントはデバイスをエクスポートできます。

（注） インポートされたデバイスを使用できるようにするには、インポートされたデバイスを持つ他のテナントが **nw-svc-policy** 特権を持つ必要があります。

第 15 章

自動化の開発

REST API について

自動化は、Application Policy Infrastructure Controller（APIC）のノースバウンド Representational State Transfer （REST）API を使用します。APIC UI を通じて実行できる処理はすべて、ノースバウンド API を使用した XML ベースの REST POST を使用して実行できます。たとえば、これらの API 経由でのイベントのモニタ、EPG のダイナミックな有効化、およびポリシーの追加などを実行できます。

また、ノースバウンド REST API を使用して、デバイスがオンボードになったことの通知や、エラーをモニタできます。いずれの場合も、モニタリングを特定のアクションをトリガーするイベントとして使用できます。たとえば、特定のアプリケーション層で発生したエラーを検出し、接続の切断がありリーフ ノードがダウンした場合、これらのアプリケーションを他の場所に再展開するアクションをトリガーできます。パケット ドロップが検出された特定のコントラクトがある場合、これらのコントラクトの複数のコピーを特定のアプリケーション上で有効化できます。また、レポートされた問題に基づいて特定のカウンタをモニタできる統計モニタリング ポリシーを使用できます。

『*Cisco APIC REST API User Guide*』には、API の使用方法の情報に加えて、ノースバウンド REST API のリストが含まれます。これらの API を使用して、APIC 上でサービスを設定するために独自の統合ツールを記述できます。

APIC ノースバウンド API にサブミットされた XML ファイルを構成する方法については、『*Cisco APIC Layer 4 to Layer 7 Device Package Development Guide*』を参照してください。

『*Cisco APIC Management Information Model Reference*』で定義されている次の Python API は、ノースバウンド API を使用した REST POST コールのサブミットに使用できます。

- `vns:LDevVip`：デバイス クラスタをアップロードします
- `vns:CDev`：デバイスをアップロードします

- `vns:LIf`：論理インターフェイスを作成します
- `vns:AbsGraph`：グラフを作成します
- `vz:BrCP`：コントラクトにグラフをアタッチします

ノースバウンド REST API を使用して、POST を構成する方法（Python での実行方法も含まれます）についての詳細は、『*Cisco APIC REST API User Guide*』を参照してください。

REST API を使用した自動化の例

ここでは、REST API を使用してタスクを自動化する例を示します。

次の REST 要求は、ブロードキャスト ドメインを持つテナント、レイヤ 3 ネットワーク、アプリケーション エンドポイント グループ、およびアプリケーション プロファイルを作成します。

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">

        <!—L3 Network-->
        <fvCtx name="MyNetwork"/>

        <!-- Bridge Domain for MySrvr EPG -->
        <fvBD name="MySrvrBD">
            <fvRsCtx tnFvCtxName="MyNetwork"/>
            <fvSubnet ip="10.10.10.10/24">
            </fvSubnet>
        </fvBD>

        <!-- Bridge Domain for MyClnt EPG -->
        <fvBD name="MyClntBD">
            <fvRsCtx tnFvCtxName="MyNetwork"/>
            <fvSubnet ip="20.20.20.20/24">
            </fvSubnet>
        </fvBD>

        <fvAp dn="uni/tn-acme/ap-MyAP" name="MyAP">

            <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MyClnt" name="MyClnt">
                <fvRsBd tnFvBDName="MySrvrBD"/>
                <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs"/>
                <fvRsProv tnVzBrCPName="webCtrct"> </fvRsProv>
                <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]"
                  encap="vlan-202"/>
                <fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]"
                  encap="vlan-202"/>
           </fvAEPg>

            <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MySRVR" name="MySRVR">
                <fvRsBd tnFvBDName="MyClntBD"/>
                <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs"/>
                <fvRsCons tnVzBrCPName="webCtrct"> </fvRsCons>
                <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]"
                  encap="vlan-203"/>
                <fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]"
                 encap="vlan-203"/>
            </fvAEPg>
        </fvAp>
    </fvTenant>
</polUni>
```

次の REST 要求は VLAN ネームスペースを作成します。

```
<polUni>
    <infraInfra>
        <fvnsVlanInstP name="MyNS" allocMode="dynamic">
```

```
            <fvnsEncapBlk name="encap" from="vlan-201" to="vlan-300"/>
        </fvnsVlanInstP>
    </infraInfra>
</polUni>
```

次の REST 要求は VMM ドメインを作成します。

```
<polUni>
    <vmmProvP vendor="Vendor1">
        <vmmDomP name="MyVMs">
            <infraRsVlanNs tDn="uni/infra/vlanns-MyNS-dynamic"/>
            <vmmUsrAccP name="admin" usr="administrator" pwd="in$1eme"/>
            <vmmCtrlrP name="vcenter1" hostOrIp="192.168.64.186">
                <vmmRsAcc tDn="uni/vmmp-Vendor1/dom-MyVMs/usracc-admin"/>
            </vmmCtrlrP>
        </vmmDomP>
    </vmmProvP>
</polUni>
```

次の REST 要求は物理ドメインを作成します。

```
<polUni>
    <physDomP name="phys">
        <infraRsVlanNs tDn="uni/infra/vlanns-MyNS-dynamic"/>
    </physDomP>
</polUni>
```

次の REST 要求は管理対象デバイス クラスタを作成します。

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1" contextAware=1>
            <vnsRsMDevAtt tDn="uni/infra/mDev-Acme-ADC-1.0"/>
            <vnsRsDevEpg tDn="uni/tn-acme/ap-services/epg-ifc"/>
            <vnsRsALDevToPhysDomP tDn="uni/phys-phys"/>

            <vnsCMgmt name="devMgmt" host="42.42.42.100" port="80"/>

            <vnsCCred name="username"value="admin"/>

            <vnsCCredSecret name="password" value="admin"/>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

次の REST 要求は非管理対象デバイス クラスタを作成します。

```
<polUni>
    <fvTenant name="HA_Tenant1">

        <vnsLDevVip name="ADCCluster1" devtype="VIRTUAL" managed="no">
            <vnsRsALDevToDomP tDn="uni/vmmp-VMware/dom-mininet"/>
        </vnsLDevVip>

    </fvTenant>
</polUni>
```

次の REST 要求はデバイス クラスタ コンテキストを作成します。

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevCtx ctrctNameOrLbl="webCtrct" graphNameOrLbl="G1" nodeNameOrLbl="Node1">
            <vnsRsLDevCtxToLDev tDn="uni/tn-acme/lDevVip-ADCCluster1"/>
            <vnsLIfCtx connNameOrLbl="ssl-inside">
                <vnsRsLIfCtxToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-int"/>
            </vnsLIfCtx>
            <vnsLIfCtx connNameOrLbl="any">
                <vnsRsLIfCtxToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-ext"/>
            </vnsLIfCtx>
        </vnsLDevCtx>
    </fvTenant>
</polUni>
```

次の要求は、ルーティング ピアリングに使用されるデバイス クラスタ コンテキストを作成します。

```
<polUni>
    <fvTenant dn="uni/tn-coke{{tenantId}}" name="coke{{tenantId}}">
        <vnsRtrCfg name="Dev1Ctx1" rtrId="180.0.0.12"/>
            <vnsLDevCtx ctrctNameOrLbl="webCtrct1" graphNameOrLbl="WebGraph"
              nodeNameOrLbl="FW">
                <vnsRsLDevCtxToLDev tDn="uni/tn-tenant1/lDevVip-Firewall"/>
                <vnsRsLDevCtxToRtrCfg tnVnsRtrCfgName="FwRtrCfg"/>
                <vnsLIfCtx connNameOrLbl="internal">
                  <vnsRsLIfCtxToInstP tDn="uni/tn-tenant1/out-OspfInternal/instP-IntInstP"

                      status="created,modified"/>
                    <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-internal"/>
                </vnsLIfCtx>
                <vnsLIfCtx connNameOrLbl="external">
                   <vnsRsLIfCtxToInstP tDn="uni/tn-common/out-OspfExternal/instP-ExtInstP"

                      status="created,modified"/>
                    <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-external"/>
            </vnsLIfCtx>
        </vnsLDevCtx>
    </fvTenant>
</polUni>
```

（注） テナント（レイヤ 3 Outside）の外部接続の設定については、『*Cisco APIC Getting Started Guide*』を参照してください。

次の REST 要求はデバイス クラスタの論理インターフェイスを追加します。

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1">
            <vnsLIf name="C5">
                <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-outside"/>
                <vnsRsCIfAtt tDn="uni/tn-acme/lDevVip-ADCCluster1/cDev-ADC1/cIf-int"/>
            </vnsLIf>
            <vnsLIf name="C4">
                <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-inside"/>
                <vnsRsCIfAtt tDn="uni/tn-acme/lDevVip-ADCCluster1/cDev-ADC1/cIf-ext"/>
            </vnsLIf>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

次の REST 要求は物理デバイス クラスタの具象デバイスを追加します。

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1">
            <vnsCDev name="ADC1" devCtxLbl="C1">
                <vnsCIf name="int">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/22]"/>
                </vnsCIf>
                <vnsCIf name="ext">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]"/>
                </vnsCIf>
                <vnsCIf name="mgmt">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/20]"/>
                </vnsCIf>
                <vnsCMgmt name="devMgmt" host="172.30.30.100" port="80"/>
                <vnsCCred name="username" value="admin"/>
                <vnsCCred name="password" value="admin"/>
            </vnsCDev>
            <vnsCDev name="ADC2" devCtxLbl="C2">
                <vnsCIf name="int">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/23]"/>
                </vnsCIf>
```

```
                <vnsCIf name="ext">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/24]"/>
                </vnsCIf>
                <vnsCIf name="mgmt">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/30]"/>
                </vnsCIf>
                <vnsCMgmt name="devMgmt" host="172.30.30.200" port="80"/>
                <vnsCCred name="username" value="admin"/>
                <vnsCCred name="password" value="admin"/>
            </vnsCDev>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

次の REST 要求は仮想デバイス クラスタの具象デバイスを追加します。

```
<polUni>
    <fvTenant dn="uni/tn-coke5" name="coke5">
        <vnsLDevVip name="Firewall5" devtype="VIRTUAL">
            <vnsCDev name="ASA5" vcenterName="vcenter1" vmName="ifav16-ASAv-scale-05">
                <vnsCIf name="Gig0/0" vnicName="Network adapter 2"/>
                <vnsCIf name="Gig0/1" vnicName="Network adapter 3"/>
                <vnsCIf name="Gig0/2" vnicName="Network adapter 4"/>
                <vnsCIf name="Gig0/3" vnicName="Network adapter 5"/>
                <vnsCIf name="Gig0/4" vnicName="Network adapter 6"/>
                <vnsCIf name="Gig0/5" vnicName="Network adapter 7"/>
                <vnsCIf name="Gig0/6" vnicName="Network adapter 8"/>
                <vnsCIf name="Gig0/7" vnicName="Network adapter 9"/>
                <vnsCMgmt name="devMgmt" host="3.5.3.170" port="443"/>
                <vnsCCred name="username" value="admin"/>
                <vnsCCredSecret name="password" value="insieme"/>
            </vnsCDev>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

次の REST 要求は管理対象サービス グラフを作成します。

```
<polUni>
    <fvTenant name="acme">
        <vnsAbsGraph name = "G1">

        <vnsAbsTermNode name = "Input1">
            <vnsAbsTermConn name = "C1" direction = "output">
            </vnsAbsTermConn>
        </vnsAbsTermNode>

        <!-- Node1 Provides SLB functionality -->
        <vnsAbsNode name = "Node1" funcType="GoTo" >
            <vnsRsDefaultScopeToTerm
                tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Output1/outtmnl"/>

            <vnsAbsFuncConn name = "C4" direction = "input">
               <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"/>

                <vnsRsConnToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-C4"/>
            </vnsAbsFuncConn>

            <vnsAbsFuncConn name = "C5" direction = "output">
               <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-internal"/>

                <vnsRsConnToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-C5"/>
            </vnsAbsFuncConn>

            <vnsRsNodeToMFunc tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB"/>
        </vnsAbsNode>

        <vnsAbsTermNode name = "Output1">
            <vnsAbsTermConn name = "C6" direction = "input">
            </vnsAbsTermConn>
        </vnsAbsTermNode>

        <vnsAbsConnection name = "CON1">
```

```
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Input1/AbsTConn"/>
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node1/AbsFConn-C4"/>
        </vnsAbsConnection>

        <vnsAbsConnection name = "CON3">
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node1/AbsFConn-C5"/>
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Output1/AbsTConn"/>
        </vnsAbsConnection>
    </vnsAbsGraph>
  </fvTenant>
</polUni>
```

次の REST 要求は非管理対象モードでサービス グラフを作成します。

```
<polUni>
    <fvTenant name="HA_Tenant1">
        <vnsAbsGraph name="g1">

            <vnsAbsTermNodeProv name="Input1">
                <vnsAbsTermConn name="C1">
                </vnsAbsTermConn>
            </vnsAbsTermNodeProv>

            <!-- Node1 Provides LoadBalancing functionality -->
            <vnsAbsNode name="Node1" managed="no">
                <vnsRsDefaultScopeToTerm
                  tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeProv-Input1/outtmnl"/>
                <vnsAbsFuncConn name="outside" attNotify="true">
                </vnsAbsFuncConn>
                <vnsAbsFuncConn name="inside" attNotify="true">
                </vnsAbsFuncConn>
            </vnsAbsNode>

            <vnsAbsTermNodeCon name="Output1">
                <vnsAbsTermConn name="C6">
                </vnsAbsTermConn>
            </vnsAbsTermNodeCon>

            <vnsAbsConnection name="CON2" adjType="L3" unicastRoute="yes">
                <vnsRsAbsConnectionConns
                  tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeCon-Output1/AbsTConn"/>
                <vnsRsAbsConnectionConns
                  tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-outside"/>
            </vnsAbsConnection>

            <vnsAbsConnection name="CON1" adjType="L2" unicastRoute="no">
                <vnsRsAbsConnectionConns
                  tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-inside"/>
                <vnsRsAbsConnectionConns
                  tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeProv-Input1/AbsTConn"/>
            </vnsAbsConnection>

        </vnsAbsGraph>
    </fvTenant>
</polUni>
```

次の REST 要求はセキュリティ ポリシー（コントラクト）を作成します。

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vzFilter name="HttpIn">
            <vzEntry name="e1" prot="6" dToPort="80"/>
        </vzFilter>

        <vzBrCP name="webCtrct">
            <vzSubj name="http">
                <vzRsSubjFiltAtt tnVzFilterName="HttpIn"/>
            </vzSubj>
        </vzBrCP>
```

```
    </fvTenant>
</polUni>
```

次の REST 要求はアプリケーション EPG からのグラフ コンフィギュレーション パラメータを提供します。

```
<polUni>
   <fvTenant dn="uni/tn-acme" name="acme">

      <!-- Application Profile -->
      <fvAp dn="uni/tn-acme/ap-MyAP" name="MyAP">

          <!-- EPG 1  -->
          <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MyClnt" name="MyClnt">
              <fvRsBd tnFvBDName="MyClntBD"/>
              <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs"/>
              <fvRsProv tnVzBrCPName="webCtrct">
              </fvRsProv>
            <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/20]" encap="vlan-201"/>

              <fvSubnet name="SrcSubnet" ip="192.168.10.1/24"/>
          </fvAEPg>

          <!-- EPG 2 -->
          <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MySRVR"  name="MySRVR">
             <fvRsBd tnFvBDName="MyClntBD"/>
             <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs"/>
             <fvRsCons tnVzBrCPName="webCtrct">
             </fvRsCons>

             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
               key="Monitor" name="monitor1">
                 <vnsParamInst name="weight" key="weight" value="10"/>
             </vnsFolderInst>

             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
               key="Service" name="Service1">
                 <vnsParamInst name="servicename" key="servicename"
                   value="crpvgrtst02-8010"/>
                 <vnsParamInst name="servicetype" key="servicetype" value="TCP"/>
                 <vnsParamInst name="servername" key="servername"
                   value="s192.168.100.100"/>
                 <vnsParamInst name="serveripaddress" key="serveripaddress"
                   value="192.168.100.100"/>
                 <vnsParamInst name="serviceport" key="serviceport" value="8080"/>
                 <vnsParamInst name="svrtimeout" key="svrtimeout" value="9000"/>
                 <vnsParamInst name="clttimeout" key="clttimeout" value="9000"/>
                 <vnsParamInst name="usip" key="usip" value="NO"/>
                 <vnsParamInst name="useproxyport" key="useproxyport" value=""/>
                 <vnsParamInst name="cip" key="cip" value="ENABLED"/>
                 <vnsParamInst name="cka" key="cka" value="NO"/>
                 <vnsParamInst name="sp" key="sp" value="OFF"/>
                 <vnsParamInst name="cmp" key="cmp" value="NO"/>
                 <vnsParamInst name="maxclient" key="maxclient" value="0"/>
                 <vnsParamInst name="maxreq" key="maxreq" value="0"/>
                 <vnsParamInst name="tcpb" key="tcpb" value="NO"/>
                 <vnsCfgRelInst name="MonitorConfig" key="MonitorConfig"
                   targetName="monitor1"/>
              </vnsFolderInst>

              <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                nodeNameOrLbl="any" key="Network" name="Network">
                 <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                   nodeNameOrLbl="any" key="vip" name="vip">
                     <vnsParamInst name="vipaddress1" key="vipaddress"
                       value="10.10.10.100"/>
                 </vnsFolderInst>
                 <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                   nodeNameOrLbl="any" devCtxLbl="C1" key="snip" name="snip1">
                     <vnsParamInst name="snipaddress" key="snipaddress"
                       value="192.168.1.100"/>
                 </vnsFolderInst>
                 <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
```

```
                    nodeNameOrLbl="any" devCtxLbl="C2" key="snip" name="snip2">
                      <vnsParamInst name="snipaddress" key="snipaddress"
                        value="192.168.1.101"/>
                    </vnsFolderInst>
                    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                      nodeNameOrLbl="any" devCtxLbl="C3" key="snip" name="snip3">
                      <vnsParamInst name="snipaddress" key="snipaddress"
                        value="192.168.1.102"/>
                    </vnsFolderInst>
                  </vnsFolderInst>

                  <!-- SLB Configuration -->
                  <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                   nodeNameOrLbl="any" key="VServer" name="VServer">
                    <!-- Virtual Server Configuration -->
                    <vnsParamInst name="port" key="port" value="8010"/>
                    <vnsParamInst name="vip" key="vip" value="10.10.10.100"/>
                    <vnsParamInst name="vservername" key="vservername"
                      value="crpvgrtst02-vip-8010"/>
                    <vnsParamInst name="servicename" key="servicename"
                      value="crpvgrtst02-8010"/>
                    <vnsParamInst name="servicetype" key="servicetype" value="TCP"/>
                    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                    nodeNameOrLbl="any" key="VServerGlobalConfig" name="VServerGlobalConfig">

                      <vnsCfgRelInst name="ServiceConfig" key="ServiceConfig"
                        targetName="Service1"/>
                      <vnsCfgRelInst name="VipConfig" key="VipConfig"
                        targetName="Network/vip"/>
                    </vnsFolderInst>
                  </vnsFolderInst>
              </fvAEPg>
        </fvAp>
    </fvTenant>
</polUni>
```

次の REST 要求はコントラクトにサービス グラフをアタッチします。

```
<polUni>
    <fvTenant name="acme">
        <vzBrCP name="webCtrct">
            <vzSubj name="http">
                <vzRsSubjGraphAtt graphName="G1" termNodeName="Input1"/>
            </vzSubj>
        </vzBrCP>
    </fvTenant>
</polUni>
```

第 16 章

GUI の使用方法

GUI を使用したレイヤ 4 ～ レイヤ 7 サービスの導入

GUI を使用して、レイヤ 4 ～ レイヤ 7 サービスを導入することができます。次の順序で手順を実行します。

1 デバイス パッケージをインポートします。

GUI を使用したデバイス パッケージのインポート，（154 ページ）を参照してください。

2 機能プロファイルを作成します。

GUI を使用した機能プロファイルの作成，（155 ページ）を参照してください。

3 サービス グラフ テンプレートを作成します。

GUI を使用したレイヤ 4 ～ レイヤ 7 サービス グラフ テンプレートの作成，（156 ページ）を参照してください。

（オプション）標準的なサービス グラフ テンプレートの代わりに、拡張サービス グラフ テンプレートを作成します。

レイヤ 4 ～ レイヤ 7 サービス グラフ テンプレート（拡張版）の作成を参照してください。

4 デバイスを作成します。

GUI を使用したデバイスの作成，（9 ページ）を参照してください。

（オプション）デバイスを変更します。

デバイスの変更，（157 ページ）を参照してください。

5 エンドポイント グループ（EGP）にサービス グラフ テンプレートを適用します。

GUI を使用したエンドポイント グループへのサービス グラフ テンプレートの適用，（159 ページ）を参照してください。

GUI を使用したデバイス パッケージのインポート

サービス グラフに基づいて設定を実行する前に、適切なデバイス パッケージをApplication Policy Infrastructure Controller（APIC）にダウンロードしてインストールする必要があります。デバイス パッケージは、所有しているデバイスと、そのデバイスで何が実行できるかを APIC に対して指定します。

ステップ 1 適切なデバイス パッケージをダウンロードします。パートナーのリストは、次の URL にあります。
http://www.cisco.com/c/en/us/solutions/data-center-virtualization/ecosystem.html

この URL は、適切なデバイス パッケージをダウンロードできる [Partner Ecosystem] ページです。

ステップ 2 プロバイダー管理者として APIC にログインします。

ステップ 3 メニュー バーで、[L4-L7 Services] > [Packages] を選択します。

ステップ 4 [Navigation] ペインで、[L4-L7 Service Device Types] をクリックします。

ステップ 5 [Work] ペインで、[Actions] > [Import Device Package] を選択します。[Import Device Package] ダイアログ ボックスが表示されます。

ステップ 6 [Browse...] をクリックし、使用するデバイス パッケージを参照します。
デバイス パッケージの作成については、『*Cisco APIC Layer 4 to Layer 7 Device Package Development Guide*』を参照してください。

ステップ 7 [Open] をクリックします。

ステップ 8 [Submit] をクリックします。

GUI を使用した機能プロファイルの作成

機能プロファイルはサービス グラフテンプレートにデフォルト値を提供します。次の手順で、新しい機能プロファイルの作成方法を説明します。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、[Tenant]*[tenant_name]* > [L4-L7 Services] > [Function Profiles] の順に選択します。

ステップ 4 [Function Profiles] を右クリックし、[Create L4-L7 Services Function Profile] を選択します。

ステップ 5 [Create L4-L7 Services Function Profile] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [Profile Group] ドロップダウン リストで、[Create Function Profile Group] を選択します。
プロファイル グループは、プロファイルをグループ化して整理するための機能です。たとえば、Web アプリケーション、レガシー アプリケーション、電子メール アプリケーション用に 1 つのプロファイルを作成することもできます。グループを作成し、そのグループにプロファイルを配置できます。使用可能な既存のグループがあるか確認します。存在しない場合は [Create L4-L7 Services Function Profile Group] ウィンドウでグループに名前を付け、説明を入力して新しいグループを作成できます。

ステップ 6 [Create L4-L7 Services Function Profile Group] ダイアログボックスでは、必要に応じてフィールドに入力します。

ステップ 7 [Submit] をクリックします。
これで、プロファイル グループが正常に作成され、保存されたことが [Create L4-L7 Services Function Profile] ダイアログボックスに表示されます。

サービス プロファイルを特定の機能用に作成します。[Create L4-L7 Services Function Profile] の [Device Function] ドロップダウン リストから選択した機能に対して、プロファイルを作成します。デバイス パッケージをインポートした後、ドロップダウン リストにはデバイス パッケージと、Application Policy Infrastructure Controller（APIC）で使用可能なサービス機能のリストが表示されます。

ステップ 8 [Create L4-L7 Services Function Profile] ダイアログボックスで、[Copy Existing Profile Parameters] チェックボックスをオフにします。

ステップ 9 [Device Function] ドロップダウン リストで、機能を持つデバイス パッケージを選択します。
その機能に含まれる各パラメータと共に、オプションが表示されます。プロファイルはパラメータにデフォルト値を提供することを目的としています。

（注） この時点ではこれらのパラメータのいずれにも値はありませんが、値を追加することで、それらがデフォルト値として使用されます。機能プロファイルは、これらの値を指定した後に、グラフ テンプレートで使用できるようになります。これらの値はデフォルト値としてグラフ テンプレートに適用されます。つまり、グラフ テンプレートを使用していて特定のパラメータに値を指定しなければ、APIC がプロファイルをルックアップし、値がそこにあるか確認します。値があれば、APIC はそれを使用します。

ステップ 10 [Create L4-L7 Services Function Profile] ウィンドウの下部にある [Features and Parameters] セクションで値を追加します。このセクションには 2 つのタブ、[Basic Parameters] と [All Parameters] があります。[Basic

Parameters] タブには、パッケージに必須（必要）とマークされたパラメータのリストが含まれています。[All Parameters] タブには基本パラメータと共に、高度な設定を行うための追加/オプションのいくつかのパラメータのリストが含まれています。[Basic Parameters] パラメータが公開されている理由は、それらが基本設定の一部であり、管理者にはこれらをすべてを入力することが期待されているからです。[All Parameters] はオプションであるため、機能をカスタマイズしない限り、これらのパラメータは省略することもできます。

ステップ 11 [Submit] をクリックします。
これで、機能プロファイルは作成・保存されました。

GUI を使用した既存の機能ファイルを使用しての新しい機能プロファイルの作成

この手順では、既存の機能プロファイルを使用して新しい機能プロファイルを作成します。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。
ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。
ステップ 3 [Navigation] ペインで、[Tenant]*[tenant_name]* > [L4-L7 Services] > [Function Profiles] の順に選択します。
ステップ 4 [Function Profiles] を右クリックし、[Create L4-L7 Services Function Profile] を選択します。
ステップ 5 [Create L4-L7 Services Function Profile] ダイアログボックスで、下記で指定している項目を除き、必要に応じてフィールドに入力します。

a) [Profile] ドロップダウン リストで、ベンダーが指定する既存のプロファイルを選択します。
選択したプロファイルに基づいて、新しいプロファイルにパラメータが挿入されます。
b) 必要に応じて、この既存のプロファイルに変更を加えたり、パラメータを追加したりします。

ステップ 6 [Submit] をクリックします。

GUI を使用したレイヤ 4 ～ レイヤ 7 サービス グラフ テンプレートの作成

サービス グラフ テンプレートは、機能プロファイルを使用して提供可能な一連のレイヤ 4 ～ レイヤ 7 機能またはデバイスと、それらの関連付けられた設定です。サービス グラフ テンプレートは、レイヤ 4 ～ レイヤ 7 デバイス上およびファブリック上に「レンダリングされる」または設定されるコントラクトと関連付ける必要があります。

はじめる前に

- テナントを作成しておく必要があります。

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、[Tenant]*[tenant_name]* > [L4-L7 Services] > [L4-L7 Service Graph Templates] の順に選択します。

ステップ 4 [Work] ペインで、[Actions] > [Create a L4-L7 Service Graph Template] の順に選択します。

ステップ 5 [Create a L4-L7 Service Graph Template] ダイアログボックスの [Device Clusters] セクションで、デバイス クラスタを選択します。

ステップ 6 次のフィールドに入力します。

名前	説明
[Graph Name] フィールド	サービス グラフ テンプレートの名前を入力します。
[Graph Type] オプション ボタン	新しいサービス グラフテンプレートを作成するか、または既存のサービス グラフ テンプレートを複製します。
[Existing Graphs] ドロップダウン リスト	（既存のサービス グラフテンプレートを複製する場合のみ）複製する既存のサービス グラフ テンプレートを選択します。

ステップ 7 （新しいサービス グラフ テンプレートを作成する場合のみ）[Device Clusters] セクションからデバイスをドラッグし、コンシューマ エンドポイント グループとプロバイダー エンドポイント グループの間にドロップしてサービス ノードを作成します。

ステップ 8 （任意） （既存のサービス グラフ テンプレートを複製する場合のみ）既存のノードを削除し、別のデバイス クラスタをノード領域にドラッグしてサービス ノードを作成します。

ステップ 9 [Submit] をクリックします。

ステップ 10 （任意） [Navigation] ペインで、サービス グラフ テンプレートをクリックします。
そのサービス グラフ テンプレートのグラフィック トポロジが画面に表示されます。

デバイスの変更

デバイスを作成した後で、そのデバイスを変更することができます。

（注） デバイスを作成するか、または既存のクラスタにデバイスを追加するには、「デバイスの作成」の手順を使用する必要があります。

ステップ 1 メニュー バーで、[TENANTS] タブをクリックします。[Tenant] ウィンドウが表示されます。

ステップ 2 [Navigation] ペインで、[Tenant] ブランチを展開し、[L4-L7 Services] ブランチを展開して、[L4-L7 Devices] をクリックします。
[L4-L7 Devices] ウィンドウが展開され、デバイスが表示されます。

ステップ 3 変更するメイン デバイス グループをクリックします。
L4-L7 Devices] ウィンドウが表示され、[General] ページが表示されます。このページには、全般情報とともに、このデバイスの接続とクレデンシャルも表示されます。さらに、デバイスの設定状態も表示されます。

ステップ 4 この [General] ウィンドウのセクションの多くのパラメータは変更できます。
このウィンドウでインターフェイスのパスは変更できますが、ここでインターフェイスを追加することはできません。

a) パスを変更するには、変更するパスをダブルクリックします。
[Properties] ウィンドウが表示されます。

b) 次のフィールドに入力します。

名前	説明
[Path] ドロップダウン リスト	使用するパスをクリックします。
[Logical Interface] ドロップダウン リスト	作業しているコネクタに応じて、[external] または [internal] をクリックします。

c) [Submit] をクリックします。

ステップ 5 [General] ウィンドウに戻ります。

ステップ 6 [General] ページの上部にある [Parameters] タブをクリックします。
このパッケージとこのデバイス グループで使用可能な機能とパラメータが画面に表示されます。基本パラメータが表示された [Basic] タブと、デバイス パッケージで使用可能なすべてのパラメータが表示された [All Parameters] タブが表示されます。（基本パラメータは [All Parameters] に含まれています）。

ステップ 7 [Features] セクションで、パラメータを変更する一連の機能をクリックします。
画面には表示パラメータと機能が表示されます。使用する特定のパッケージと選択した特定の機能に応じて、一連のパラメータは変化します。

ステップ 8 必要な機能の値を次のように指定または変更します。

（注） 親フォルダを最初に更新する必要があります。

a) 変更するフィールドをダブルクリックします。

b) 表示されたフィールドに必要な情報を入力します。
c) [Update] ボタンをクリックします。

（注） [Reset] をクリックしてこれらの値を以前の値に戻します。

ステップ 9 単一デバイスのパラメータを表示または変更するには、[Navigation] ペインの [Tenant] ブランチを展開し、[L4-L7 Services] ブランチを展開して、[L4-L7 Devices] をクリックします。
[L4-L7 Devices] ウィンドウが展開され、デバイスが表示されます。

ステップ 10 特定のデバイスが属するメインのデバイス グループをクリックします。
[Navigation] ペインに、メイン デバイス グループ内のデバイスが表示されます。

ステップ 11 パラメータを表示または変更するデバイスをクリックします。
指定したデバイスのグラフィック トポロジが両方ともに表示されます。

ステップ 12 [Parameters] タブをクリックします。
画面には、現在の機能とパラメータの両方が表示されます。この画面で値を変更できます。基本パラメータが表示された [Basic] タブと、デバイス パッケージで使用可能なすべてのパラメータが表示された [All Parameters] タブが表示されます。（基本パラメータは [All Parameters] に含まれています）。

ステップ 13 [Features] セクションで、パラメータを変更する一連の機能をクリックします。
画面には、単一デバイスのパラメータと機能が表示されます。使用する特定のパッケージと選択した特定の機能に応じて、一連のパラメータは変化します。

ステップ 14 必要な機能の値を次のように指定または変更します。

a) 変更するフィールドをダブルクリックします。
b) 表示されたフィールドに必要な情報を入力します。
c) [Update] ボタンをクリックします。

（注） [Reset] をクリックしてこれらの値を以前の値に戻します。

GUI を使用したエンドポイント グループへのサービス グラフ テンプレートの適用

次の手順で、エンドポイント グループへのサービス グラフ テンプレートの適用法を説明します。

はじめる前に

次を作成しておく必要があります。

- アプリケーション エンドポイント グループ

- サービス グラフ テンプレート

ステップ 1 メニュー バーで、[Tenants] > [All Tenants] の順に選択します。

ステップ 2 [Work] ペインで、テナントの名前をダブルクリックします。

ステップ 3 [Navigation] ペインで、[Tenant]*[tenant_name]* > [L4-L7 Services] > [L4-L7 Service Graph Templates] > *[template_name]* の順に選択します。

ステップ 4 [Work] ペインで、[Actions] > [Apply L4-L7 Service Graph Template] の順に選択します。
レイヤ 4 ～ レイヤ 7 サービス グラフ テンプレートをコンシューマ エンドポイント グループとプロバイダー エンドポイント グループに関連付けます。

ステップ 5 [Apply L4-L7 Service Graph Template To EPGs] ダイアログの [EPG Information] セクションで、次のフィールドに入力します。

名前	説明
[Consumer EPG/External Network] ドロップダウン リスト	コンシューマ エンドポイント グループを選択します。
[Provider EPG/External Network] ドロップダウン リスト	プロバイダー エンドポイント グループを選択します。

ステップ 6 [Contract Information] セクションで、次のフィールドに入力します。

名前	説明
[Contract] オプション ボタン	コントラクトの作成を選択するか、または既存のコントラクトを選択します。
[Contract Name] フィールド	（コントラクトを作成する場合のみ）コントラクトの名前を入力します。
[No Filter (Allow All Traffic)] チェックボックス	（コントラクトを作成する場合のみ）すべてのトラフィックを許可する場合はチェックボックスをオンにし、トラフィックをフィルタリングする場合はチェックボックスをオフにします。
[Filter Entries]	（トラフィックをフィルタリングする場合のみ）[+] をクリックしてフィルタ情報を入力し、[Update] をクリックします。
[Existing Contract With Subjects] ドロップダウンリスト	（既存のコントラクトを選択する場合のみ）既存のコントラクトを選択します。

ステップ 7 [Next] をクリックします。

ステップ 8 [Device Clusters] セクションで、デバイス クラスタを選択します。

ステップ 9 次のフィールドに入力します。

名前	説明
[Graph Template] ドロップダウン リスト	グラフ テンプレートを選択します。

ステップ 10 （オプション）既存のノードを削除し、別のデバイス クラスタをノード領域にドラッグしてサービス ノードを作成します。

ステップ 11 [unmanaged information] セクションで、次のフィールドに入力します。

名前	説明
[Cluster Interface For Consumer Connector] ドロップダウン リスト	コンシューマ コネクタのインターフェイスを選択します。
[Cluster Interface For Provider Connector] ドロップダウン リスト	プロバイダー コネクタのインターフェイスを選択します。
[General] チェックボックス	ブリッジ ドメインを選択できるようにするには、チェックボックスをオンにします。
[BD For Consumer Connector] ドロップダウン リスト	（[General] チェックボックスをオンにした場合のみ）コンシューマ コネクタのブリッジ ドメインを選択します。ブリッジ ドメインは、データパス トラフィックに使用されます。
[BD For Provider Connector] ドロップダウン リスト	（[General] チェックボックスをオンにした場合のみ）プロバイダー コネクタのブリッジ ドメインを選択します。ブリッジ ドメインは、データパス トラフィックに使用されます。
[Route Peering] チェックボックス	ルート ピアリングを有効にするには、チェックボックスをオンにします。

Application Policy Infrastructure Controller（APIC）は、選択したサービス グラフ テンプレートで必要な機能ノード間のデータ パス トラフィックに選択したブリッジ ドメインを使用します。このブリッジ ドメインの使用方法を詳しく知るには、サービス グラフ テンプレートのオンライン ヘルプを参照してください。

ステップ 12 （管理対象デバイスの場合のみ） [Next] をクリックします。

ステップ 13 （管理対象デバイスのみ）[Parameters] セクションの [Required Parameters] タブで、必要に応じて、すべての必須パラメータの名前と値を入力します。

ステップ 14 [Finish] をクリックします。
サービス グラフ テンプレートがアクティブになりました。APIC は、選択した機能プロファイルと色に基づいて、レイヤ 4 ～ レイヤ 7 プラメータを挿入し、正しく設定されている場合は必須パラメータを緑色で表示します。