テナント ポリシーの設定

基本的なテナント設定

REST API を使用したテナント、VRF、およびブリッジ ドメインの作成

手順

ステップ 1 テナントを作成します。

例：

```
 POST https://apic-ip-address/api/mo/uni.xml
<fvTenant name="ExampleCorp"/>
```

POST が成功すると、作成したオブジェクトが出力に表示されます。

ステップ 2 VRF およびブリッジ ドメインを作成します。

（注） ゲートウェイ アドレスは、IPv4 または IPv6 アドレスにすることができます。IPv6 ゲートウェイ アドレスの詳細については、関連する KB 記事、「*KB: Creating a Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery*」を参照してください。

例：

```
 URL for POST: https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml

<fvTenant name="ExampleCorp">
   <fvCtx name="pvn1"/>
   <fvBD name="bd1">
      <fvRsCtx tnFvCtxName="pvn1"/>
      <fvSubnet ip="10.10.100.1/24"/>
   </fvBD>
</fvTenant>
```

（注） 外部ルーテッドを設定するときにパブリック サブネットがある場合は、ブリッジ ドメインを外部設定と関連付ける必要があります。

複数のプライベート ネットワークのテナント

テナント間通信を行う複数のプライベート ネットワークについて

- この使用例は一般的に、ACIの管理者が、テナント間通信の機能を持つ複数のテナントを作成し、サポートしたい場合に使用されます。

  この方法には、次の利点と欠点があります。

  利点：

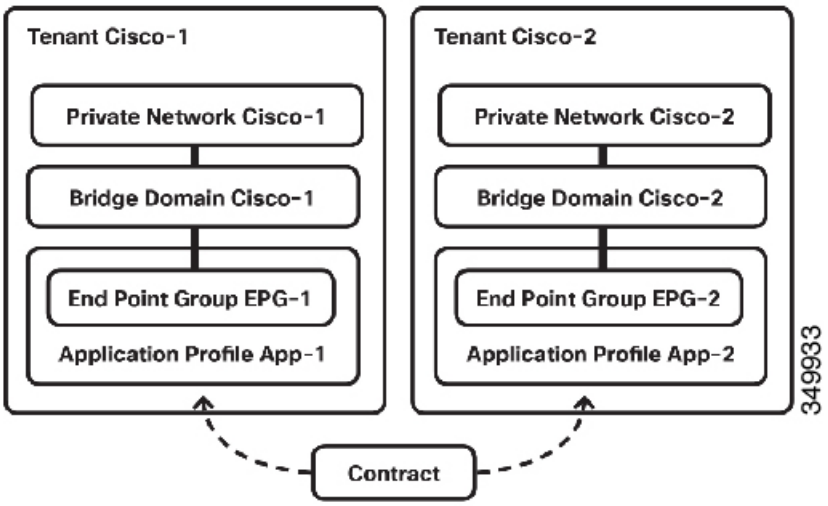
  - 各テナント コンテナの個別管理が可能
  - テナント間の最大分離を実現

  欠点：

  - テナントのアドレス空間は一意である必要がある

  オブジェクト間の関係とコントラクトの観点から、このトポロジは次のように表されます。

図 *1:* テナント間通信を行う複数のプライベート ネットワーク



REST API を使用してテナント間の通信に複数のプライベート ネットワークの設定

次の手順で、REST API を使用して、それらの間の通信を Cisco 1 および Cisco 2 のプライベート ネットワークを設定します。

手順

ステップ 1 APIC REST API に post 送信されて、次の XML を使用してテナントを Cisco 1 を設定します。

例：

```
<fvTenant dn="uni/tn-Cisco1" name="Cisco1">

<vzBrCP name="ICMP" scope="global">
<vzSubj consMatchT="AtleastOne" name="icmp" provMatchT="AtleastOne"
revFltPorts="yes">
<vzRsSubjFiltAtt tnVzFilterName="icmp"/>
</vzSubj>
</vzBrCP>

<vzCPIf dn="uni/tn-Cisco1/cif-ICMP" name="ICMP">

<vzRsIf consMatchT="AtleastOne" name="icmp" provMatchT="AtleastOne"
revFltPorts="yes">
<vzRsSubjFiltAtt tDn="uni/tn-Cisco2/brc-default"/>
</vzRsIf>
</vzCPIf>
<fvCtx knwMcastAct="permit" name="CiscoCtx" pcEnfPref="enforced"/>
```

```
<fvBD arpFlood="yes" mac="00:22:BD:F8:19:FF" name="CiscoBD2" unicastRoute="yes"
unkMacUcastAct="flood" unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx2"/>
</fvBD>
<fvBD arpFlood="yes" name="CiscoBD" unicastRoute="yes" unkMacUcastAct="flood"
unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx"/>
</fvBD>
<fvAp name="CCO">
<fvAEPg matchT="AtleastOne" name="EPG1">
<fvRsPathAtt encap="vlan-1202" instrImedcy="immediate" mode="native"
tDn="topology/pod-1/paths-202/pathep-[eth1/2]"/>
<fvSubnet ip="172.16.1.1/24" scope="private,shared"/>

<fvRsDomAtt instrImedcy="lazy" resImedcy="lazy" tDn="uni/phys-
PhysDomainforCisco"/>

<fvRsBd tnFvBDName="CiscoBD"/>
<fvRsProv matchT="AtleastOne" tnVzBrCPName="ICMP"/>
</fvAEPg>
</fvAp>
</fvTenant>

<fvRsBd tnFvBDName="CiscoBD"/>
<fvRsProv matchT="AtleastOne" tnVzBrCPName="ICMP"/>
</fvAEPg>
</fvAp>
</fvTenant>
```

ステップ 2 APIC REST API に post 送信されて、次の XML を使用して Cisco 2 tenanat を設定します。

例：

```
<fvTenant dn="uni/tn-Cisco2" name="Cisco2">
<fvCtx knwMcastAct="permit" name="CiscoCtx" pcEnfPref="enforced"/>
<fvBD arpFlood="yes" mac="00:22:BD:F8:19:FF" name="CiscoBD2" unicastRoute="yes"
unkMacUcastAct="flood" unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx"/>
</fvBD>
<fvBD arpFlood="yes" name="CiscoBD" unicastRoute="yes" unkMacUcastAct="flood"
unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx"/>
</fvBD>
<fvAp name="CCO">
<fvAEPg matchT="AtleastOne" name="EPG2">
<fvRsPathAtt encap="vlan-1202" instrImedcy="immediate" mode="native"
tDn="topology/pod-1/paths-201/pathep-[eth1/2]"/>
<fvSubnet ip="172.16.1.1/24" scope="private,shared"/>

<fvRsDomAtt instrImedcy="lazy" resImedcy="lazy" tDn="uni/phys-
PhysDomainforCisco"/>

<fvRsBd tnFvBDName="CiscoBD"/>
<fvRsConsIf matchT="AtleastOne" tnVzBrCPIfName="ICMP"/>
</fvAEPg>
</fvAp>
</fvTenant>
```

テナント内通信を行う複数のプライベート ネットワークについて

サポートすることが望ましい別の使用例として、複数のプライベート ネットワークを持つ単一テナントを所有するというオプションがあります。これは、管理レベルではなく、ネットワーク レベルでマルチテナント機能を提供する必要があるためです。また、合併やその他のビジネス上の変更により、単一テナント内で重複しているサブネットをサポートする場合にも発生します。

この方法には、次の利点と欠点があります。

利点：

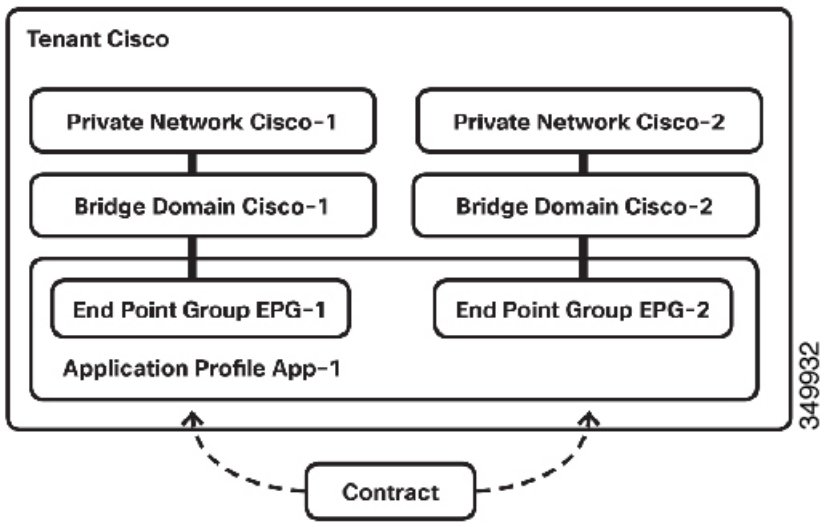
- 単一テナント内で重複するサブネットを所有できる

欠点：

- 重複するサブネットに存在する EPG は互いに適用し合うポリシーを所有できない

この特定の設定のオブジェクト間の関係は次のように表すことができます。

図 *2:* テナント内通信を行う複数のプライベート ネットワーク



REST API を使用して、テナント内の通信に複数のテナントの設定

手順

APIC REST API に post 送信されて、次の XML を使用して、Cisco 1 および Cisco 2 ネットワークと、テナント Cisco の設定します。

例：

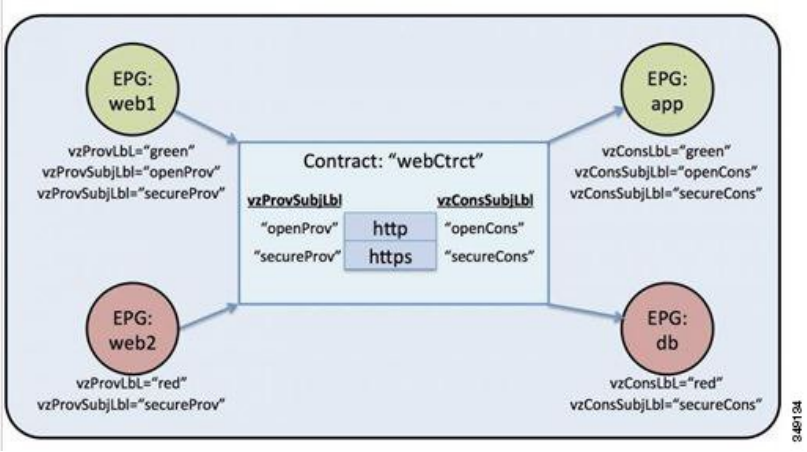
```
<fvTenant dn="uni/tn-Cisco" name="Cisco">
<vzBrCP name="ICMP" scope="tenant">
<vzSubj consMatchT="AtleastOne" name="icmp" provMatchT="AtleastOne"
revFltPorts="yes">
<vzRsSubjFiltAtt tnVzFilterName="icmp"/>
</vzSubj>
</vzBrCP>
<fvCtx knwMcastAct="permit" name="CiscoCtx" pcEnfPref="enforced"/>
<fvCtx knwMcastAct="permit" name="CiscoCtx2" pcEnfPref="enforced"/>
<fvBD arpFlood="yes" mac="00:22:BD:F8:19:FF" name="CiscoBD2" unicastRoute="yes"
unkMacUcastAct="flood" unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx2"/>
</fvBD>
<fvBD arpFlood="yes" mac="00:22:BD:F8:19:FF" name="CiscoBD" unicastRoute="yes"
unkMacUcastAct="flood" unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx"/>
</fvBD>
<fvAp name="CCO">
<fvAEPg matchT="AtleastOne" name="Web">
<fvRsCons tnVzBrCPName="ICMP"/>
<fvRsPathAtt encap="vlan-1201" instrImedcy="immediate" mode="native"
tDn="topology/pod-1/paths-201/pathep-[eth1/16]"/>
<fvSubnet ip="172.16.2.1/24" scope="private,shared"/>
<fvRsDomAtt instrImedcy="lazy" resImedcy="lazy" tDn="uni/phys-
PhysDomainforCisco"/>
<fvRsBd tnFvBDName="CiscoBD2"/>
</fvAEPg>
<fvAEPg matchT="AtleastOne" name="App">
<fvRsPathAtt encap="vlan-1202" instrImedcy="immediate" mode="native"
tDn="topology/pod-1/paths-202/pathep-[eth1/2]"/>
<fvSubnet ip="172.16.1.1/24" scope="private,shared"/>
<fvRsDomAtt instrImedcy="lazy" resImedcy="lazy" tDn="uni/phys-
PhysDomainforCisco"/>
<fvRsBd tnFvBDName="CiscoBD"/>
<fvRsProv matchT="AtleastOne" tnVzBrCPName="ICMP"/>
</fvAEPg>
</fvAp>
</fvTenant>
```

テナント ポリシーの例

テナント ポリシー例の概要

この付録のテナント ポリシー例の説明では、XML 用語（http://en.wikipedia.org/wiki/XML#Key_terminology）を使用します。この例では、基本的な APIC ポリシー モデル構造が XML コードにどのようにレンダリングされるかを示します。次の図は、テナント ポリシー例の概要について説明します。

図 *3 :* テナント *Solar* に含まれる *EPG* とコントラクト



この図では、webCtrct および EPG ラベルと呼ばれるコントラクトに従って、グリーンラベルの EPG:web1 が http と https の両方を使用してグリーンラベルの EPG:app と通信でき、レッドラベルの EPG:web2 は https のみを使用してレッドラベルの EPG:db と通信できます。

テナント ポリシー例の XML コード

```
<polUni>
  <fvTenant name="solar">

      <vzFilter name="Http">
          <vzEntry name="e1"

              etherT="ipv4"
              prot="tcp"
              dFromPort="80"
              dToPort="80"/>
      </vzFilter>

      <vzFilter name="Https">
          <vzEntry name="e1"
              etherT="ipv4"
              prot="tcp"
              dFromPort="443"
              dToPort="443"/>
      </vzFilter>

      <vzBrCP name="webCtrct">
          <vzSubj name="http" revFltPorts="true" provmatchT="All">
              <vzRsSubjFiltAtt tnVzFilterName="Http"/>
              <vzRsSubjGraphAtt graphName="G1" termNodeName="TProv"/>
              <vzProvSubjLbl name="openProv"/>
              <vzConsSubjLbl name="openCons"/>
          </vzSubj>
          <vzSubj name="https" revFltPorts="true" provmatchT="All">
              <vzProvSubjLbl name="secureProv"/>
              <vzConsSubjLbl name="secureCons"/>
              < vzRsSubjFiltAtt tnVzFilterName="Https"/>
              <vzRsOutTermGraphAtt graphName="G2" termNodeName="TProv"/>
          </vzSubj>
```

```
        </vzBrCP>

        <fvCtx name="solarctx1"/>

        <fvBD name="solarBD1">
            <fvRsCtx tnFvCtxName="solarctx1" />
            <fvSubnet ip="11.22.22.20/24">
                <fvRsBDSubnetToProfile
                  tnL3extOutName="rout1"
                  tnRtctrlProfileName="profExport"/>
            </fvSubnet>
            <fvSubnet ip="11.22.22.211/24">
                <fvRsBDSubnetToProfile
                  tnL3extOutName="rout1"
                  tnRtctrlProfileName="profExport"/>
            </fvSubnet>
        </fvBD>

        <fvAp name="sap">
            <fvAEPg name="web1">
                <fvRsBd tnFvBDName="solarBD1" />
                <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
                <fvRsProv tnVzBrCPName="webCtrct" matchT="All">
                    <vzProvSubjLbl name="openProv"/>
                    <vzProvSubjLbl name="secureProv"/>
                    <vzProvLbl name="green"/>
                </fvRsProv>
            </fvAEPg>
            <fvAEPg name="web2">
                <fvRsBd tnFvBDName="solarBD1" />
                <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
                <fvRsProv tnVzBrCPName="webCtrct" matchT="All">
                    <vzProvSubjLbl name="secureProv"/>
                    <vzProvLbl name="red"/>
                </fvRsProv>
            </fvAEPg>
            <fvAEPg name="app">
                <fvRsBd tnFvBDName="solarBD1" />
                <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
                <fvRsCons tnVzBrCPName="webCtrct">
                    <vzConsSubjLbl name="openCons"/>
                    <vzConsSubjLbl name="secureCons"/>
                    <vzConsLbl name="green"/>
                </fvRsCons>
            </fvAEPg>
            <fvAEPg name="db">
                <fvRsBd tnFvBDName="solarBD1" />
                <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
                <fvRsCons tnVzBrCPName="webCtrct">
                    <vzConsSubjLbl name="secureCons"/>
                    <vzConsLbl name="red"/>
                </fvRsCons>
            </fvAEPg>
        </fvAp>
    </fvTenant>
</polUni>
```

テナント ポリシー例の説明

この項には、テナント ポリシー例の詳しい説明が含まれます。

ポリシー ユニバース

ポリシーユニバースには、各テナントのポリシーが定義されているすべてのテナント管理対象オブジェクトが含まれます。

```
<polUni>
```

最初の行のこの開始タグ <polUni> は、ポリシー ユニバース要素の開始を示します。このタグは、ポリシーの最後にある </polUni> と一致します。間にあるのはすべて、ポリシー定義です。

テナント ポリシーの例

タグ <fvTenant> は、テナント要素の開始を識別します。

```
<fvTenant name="solar">
```

このテナントのポリシーはすべてこの要素で定義されます。この例でのテナントの名前はsolarです。テナントの名前はシステム内で一意である必要があります。テナントが含む主要な要素は、フィルタ、コントラクト、外部ネットワーク、ブリッジ ドメイン、およびEPGを含むアプリケーション プロファイルです。

Filters

フィルタ要素は、タグ <vzFilter> から始まり、タグ <vzEntry> で示される要素が含まれます。

次に、「Http」と「Https」フィルタを定義する例を示します。フィルタの最初の属性が名前で、name 属性の値はテナントに一意の文字列です。これらの名前は異なるテナントで再利用できます。これらのフィルタは、この例の後でコントラクト内のサブジェクト要素で使用されます。

```
<vzFilter name="Http">
    <vzEntry name="e1" etherT="ipv4" prot="tcp" dFromPort="80" dToPort="80"/>
</vzFilter>

<vzFilter name="Https">
    <vzEntry name="e1" etherT="ipv4" prot="tcp" dFromPort="443" dToPort="443"/>
</vzFilter>
```

この例では、2 つのフィルタ、Http および Https を定義します。フィルタの最初の属性はその名前で、name 属性の値はテナントに一意の文字列です。つまり、これらの名前は異なるテナントで再利用できます。これらのフィルタは、この例の後でコントラクト内のサブジェクト要素で使用されます。

各フィルタには、各エントリがレイヤ 4 TCP または UDP ポート番号のセットを説明する 1 つ以上のエントリを含めることができます。<vzEntry> 要素の考えられる属性の一部は次のとおりです。

- name
- prot
- dFromPort
- dToPort

- sFromPort
- sToPort
- etherT
- ipFlags
- arpOpc
- tcpRules

この例では、各エントリの name 属性が指定されます。名前はフィルタ内で一意でなければならない ASCII 文字列ですが、他のフィルタで再利用できます。なぜなら、この例では、後で特定のエントリを参照せず、「e1」という単純な名前が与えられるためです。

EtherType 属性 etherT が次です。ipv4 の値が割り当てられ、このフィルタが IPv4 パケット用であることを指定します。この属性には考えられる他の多くの値があります。一般的なものは、ARP、RARP、IPv6 などです。デフォルトは unspecified なので、値を割り当てることが重要です。

EtherType 属性の後は、prot 属性です。この属性は tcp に設定され、このフィルタが TCP トラフィック用であることを示します。代替プロトコル属性には、udp、icmp、および unspecified（デフォルト）があります。

プロトコルの後、宛先の TCP ポート番号は 80 ～ 80 の範囲（正確には TCP port 80）になるように dFromPort および dToPort 属性で割り当てられます。送信元と宛先が異なっている場合、それらはポート番号の範囲を指定します。

この例では、これらの宛先ポート番号は属性 dFromPort および dToPort で指定されます。ただし、コントラクトで使用されている場合は、TCP クライアントからサーバへの宛先ポートのためにリターン トラフィックの送信元ポートとして使用する必要があります。詳細については、この例の後に出てくる属性 revFltPorts を参照してください。

2 番目のフィルタは基本的に同じ機能がありますが、ポート 443 に対するものです。

フィルタは、ターゲットの識別名 tDn によってコントラクト内のサブジェクトによって参照されます。tDn 名は次のように構成されます。

```
uni/tn-<tenant name>/flt-<filter name>
```

たとえば、上記の最初のフィルタの tDn は uni/tn-coke/flt-Http です。2 番目のフィルタには名前 uni/tn-coke/flt-Https があります。いずれの場合も、solar がテナント名から取得されます。

コントラクト

コントラクト要素は、vzBrCP でタグ付けされ、name 属性があります。

```
<vzBrCP name="webCtrct">
   <vzSubj name="http" revFltPorts="true" provmatchT="All">
       <vzRsSubjFiltAtt tnVzFilterName="Http"/>
       <vzRsSubjGraphAtt graphName="G1" termNodeName="TProv"/>
       <vzProvSubjLbl name="openProv"/>
       <vzConsSubjLbl name="openCons"/>
```

```
        </vzSubj>
        <vzSubj name="https" revFltPorts="true" provmatchT="All">
            <vzProvSubjLbl name="secureProv"/>
            <vzConsSubjLbl name="secureCons"/>
            <vzRsFiltAtt tnVzFilterName="Https "/>
            <vzRsOutTermGraphAtt graphName="G2" termNodeName="TProv"/>
        </vzSubj>
    </vzBrCP>
```

コントラクトは EPG 間のポリシー要素です。コントラクトには、コントラクトを作成して消費する EPG 間で適用されるすべてのフィルタが含まれます。コントラクト要素は、`vzBrCP` でタグ付けされ、name 属性があります。コントラクト要素で使用できるその他の属性については、オブジェクト モデルの参照資料を参照してください。この例では、`webCtrct` という名前のコントラクトが 1 つあります。

コントラクトには、各サブジェクトが一連のフィルタを含む複数のサブジェクト要素が含まれます。この例では、2 つのサブジェクト、`http` と `https` があります。

コントラクトは、それを提供または消費する EPG によって後で参照されます。EPG は、以下の方法で名前によってそのコントラクトを参照します。

```
uni/tn-[tenant-name]/brc-[contract-name]
```

`tenant-name` はテナントの名前で、この例では「solar」となります。contract-name はコントラクトの名前です。この例では、コントラクトの `tDn` 名は `uni/tn-solar/brc-webCtrct` です。

サブジェクト

サブジェクト要素は、タグ `vzSubj` から始まり、3 つの属性、`name`、`revFltPorts` および `matchT` を持ちます。`name` は、単にサブジェクトの ASCII 名です。

`revFltPorts` は、このサブジェクトのフィルタ内のレイヤ 4 送信元および宛先ポートをフィルタの説明に示すとおりに転送方向（つまり、コンシューマからプロデューサ EPG の方向）に使用する必要があり、逆方向には逆の方法で使用する必要があることを示すフラグです。この例では、「http」サブジェクトには、TCP宛先ポート80を定義し、送信元ポートを指定していない「Http」フィルタが含まれます。`revFltPorts` フラグが true に設定されているため、ポリシーは、TCP **宛先ポート 80** およびコンシューマからプロデューサへのトラフィック用の送信元ポートであり、また、TCP宛先ポートおよびプロデューサからコンシューマへのトラフィック用の**送信元ポート 80** になります。コンシューマがプロデューサへの TCP 接続を開始することを前提としています（コンシューマがクライアントで、プロデューサがサーバ）。

指定しない場合、`revFltPrts` 属性のデフォルト値は `false` です。

ラベル

一致タイプ属性、`provmatchT`（プロバイダー一致の場合）および `consmatchT`（コンシューマー一致の場合）は、サブジェクトが所定のコンシューマとプロデューサのペアに対し適用されるかを判断するためにサブジェクトラベルがどのように比較されるかを決定します。次の一致タイプの値が使用可能です。

- すべて
- `AtLeastOne`（デフォルト）

- None
- ExactlyOne

サブジェクトがプロデューサとコンシューマ EPG 間のトラフィックに適用されるかどうかを決定する場合、一致属性は、これらの EPG で定義されている（または定義されていない）サブジェクト ラベルがサブジェクト内のラベルとどのように比較されるべきかを決定します。一致属性の値が All に設定されると、それはプロバイダー サブジェクト ラベル vzProvSubjLbl がサブジェクト内で定義されたすべての vzProvSubjLbl ラベルと一致するプロバイダーにのみ適用されます。2 つのラベルが定義されている場合、両方ともプロバイダー内にある必要があります。プロバイダー EPG に 10 個のラベルがある場合、サブジェクト内のすべてのプロバイダー ラベルが存在する限り、一致が確認されます。同様の基準が vzConsSubjLbl を使用するコンシューマに使用されます。matchT 属性値が AtLeastOne の場合、ラベルの 1 つだけが一致する必要があります。matchT 属性が None の場合、サブジェクト内のプロバイダー ラベルがプロバイダー EPG のプロバイダー ラベルと一致しない場合にのみ一致が発生します。コンシューマの場合も同様です。

プロデューサまたはコンシューマ EPG にサブジェクト ラベルがなく、サブジェクトがラベルを持たない場合、一致は All、AtLeastOne、および None の場合に発生します（ラベルを使用しない場合は、サブジェクトが使用され matchT 属性は問題になりません）。

この例には示されていないサブジェクトのオプション属性は prio で、フィルタに一致するトラフィックのプライオリティが指定されます。考えられる値は、gold、silver、bronze、または unspecified（デフォルト）です。

この例では、サブジェクト要素にフィルタ要素、サブジェクト ラベル要素およびグラフ要素への参照が含まれます。<vzRsSubjFiltAtt tDn="uni/tn-coke/flt-Http"/> は事前に定義されたフィルタへの参照です。この要素は vzRsSubjFiltAtt タグによって識別されます。

<vzRsSubjGraphAtt graphName="G1" termNodeName="TProv"/> は端末接続を定義します。

<vzProvSubjLbl name="openProv"/> は「openProv」という名前のプロバイダー ラベルを定義します。ラベルは、どのサブジェクトがどの EPG に適用されるかを認定したりフィルタリングするために使用されます。この特定のラベルがプロバイダー ラベルであり、対応するコンシューマ ラベルがタグ vzConsSubjLbl で識別されます。これらのラベルは、現在のコントラクトに関連付けられたプロバイダーまたはコンシューマ EPG の対応するラベルと一致します。前述の matchT 基準に従って一致が発生する場合は、特定のサブジェクトが EPG に適用されます。一致が発生しない場合、サブジェクトは無視されます。

複数のプロバイダーおよびコンシューマのサブジェクト ラベルをサブジェクトに追加して、より複雑な一致基準を可能にすることができます。この例では、各サブジェクトに各タイプのラベルが 1 個だけあります。ただし、最初のサブジェクトのラベルは 2 番目のサブジェクトのラベルとは異なり、これら 2 つのサブジェクトを対応する EPG のラベルに応じて、それぞれ処理できます。サブジェクト要素内の要素の順序は重要ではありません。

VRF

Virtual Routing and Forwarding（VRF）（コンテキストまたはプライベート ネットワークとも呼ばれる）は、fvCtx タグにより識別され、名前属性を含みます。

テナントには、複数の VRF を含めることができます。この例では、テナントは「solartx1」という名前の VRF を 1 個使用します。名前は、テナント内で一意である必要があります。

VRFは、レイヤ 3 のアドレス ドメインを定義します。レイヤ 3 ドメイン内のすべてのエンドポイントが一意の IPv4 または IPv6 アドレスを持っている必要があります。なぜなら、ポリシーで許可されている場合にこれらのデバイス間でパケットを直接転送できるためです。

VRF が一意の IP アドレス空間を定義する一方で、対応するサブネットがブリッジ ドメイン内で定義されます。各ブリッジ ドメインは VRF に関連付けられます。

ブリッジ ドメイン

ブリッジ ドメインの要素は `fvBD` タグで識別され、name 属性があります。

```
<fvBD name="solarBD1">
    <fvRsCtx tnFvCtxName="solarctx1" />
    <fvSubnet ip="11.22.22.20/24">
        <fvRsBDSubnetToProfile
             tnL3extOutName="rout1"
             tnRtctrlProfileName="profExport" />
    </fvSubnet>
    <fvSubnet ip="11.22.23.211/24">
        <fvRsBDSubnetToProfile
             tnL3extOutName="rout1"
             tnRtctrlProfileName="profExport"/>
    </fvSubnet>
</fvBD>
```

ブリッジ ドメイン要素内でサブネットが定義され、リファレンスが対応する Virtual Routing and Forwarding（VRF）インスタンスに作成されます（コンテキストまたはプライベート ネットワークとも呼ばれる）。各ブリッジ ドメインは VRF にリンクされ、少なくとも 1 個のサブネットを設定する必要があります。

この例では、「solarBD1」という名前のブリッジ ドメインを 1 個使用します。この例では、「solarctx1」という VRF が、`fvRsCtx` とタグ付けされた要素を使用して参照され、`tnFvCtxName` 属性に値「solarctx1」が与えられます。この名前は、上記で定義した VRF から取得されます。

サブネットがブリッジ ドメイン内に含まれ、ブリッジ ドメインは複数のサブネットを含むことができます。この例では、2 個のサブネットを定義します。ブリッジ ドメイン内で使用されるすべてのアドレスは、サブネットで定義されるアドレス範囲のいずれかに分類される必要があります。ただし、サブネットは、使用されることがないであろう多数のアドレスを含む大規模なサブネットであるスーパーネットにすることもできます。現在および将来のアドレスすべてに対応する大規模なサブネットを 1 個指定すると、ブリッジ ドメインの仕様を簡素化できます。ただし、異なるサブネットがブリッジ ドメイン内で重複したり、または同じ VRF に関連付けられている他のブリッジ ドメインで定義されたサブネットと重複してはなりません。サブネットは、他の VRF に関連付けられている他のサブネットと重複できます。

前述のサブネットは、11.22.22.xx/24 と 11.22.23.xx/24 です。ただし、24 だけが使用されることをマスクが示していても、アドレスの完全な 32 ビットが与えられます。それは、この IP 属性がそのサブネットに対するルータの完全な IP アドレスも示しているためです。最初のケースでは、ルータの IP アドレス（デフォルト ゲートウェイ）は 11.22.22.20 で、2 番目のサブネットでは、11.22.23.211 です。

エントリ 11.22.22.20/24 は以下に相当しますが、コンパクト形式です。

- サブネット：11.22.22.00
- サブネット マスク：255.255.255.0
- デフォルト ゲートウェイ：11.22.22.20

アプリケーション プロファイル

アプリケーション プロファイルの開始はタグ `fvAp` で示され、name 属性があります。

```
<fvAp name="sap">
```

この例では、アプリケーション ネットワーク プロファイルが 1 つあり、「sap」という名前になっています。

アプリケーションプロファイルは、EPGを保持するコンテナです。EPGは同じアプリケーション プロファイル内の他の EPG および他のアプリケーション プロファイル内の EPG と通信できます。アプリケーション プロファイルは、互いに論理的に関連する複数の EPG を保持するために使用される簡易で便利なコンテナです。それらは、「sap」などの提供するアプリケーション、「インフラストラクチャ」などの提供する機能、「DMZ」などのデータセンターの構造内のそれらが存在する場所、または管理者が使用することを選択した組織化の原則によって組織化できます。

アプリケーションプロファイルに含まれるプライマリオブジェクトは、エンドポイント グループ（EPG）です。この例では、「sap」アプリケーションプロファイルには4個のEPG、web1、web2、app および db が含まれます。

エンドポイントおよびエンドポイント グループ（EPG）

EPG は、タグ `fvAEPg` で始まり、name 属性があります。

```
<fvAEPg name="web1">
    <fvRsBd tnFvBDName="solarBD1" />
    <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
    <fvRsProv tnVzBrCPName="webCtrct" matchT ="All">
        <vzProvSubjLbl name="openProv"/>
        <vzProvSubjLbl name="secureProv"/>
        <vzProvLbl name="green"/>
    </fvRsProv>
</fvAEPg>
```

EPG は、ポリシー モデルの最も重要な基本オブジェクトです。これは、ポリシーの観点から同じ方法で処理されるエンドポイントの集合を表します。それらのエンドポイントは個別に設定および管理されるのではなく、EPG 内に配置され、集合またはグループとして管理されます。

EPG オブジェクトは、どのようなポリシーが適用されるのか、また他のどの EPG がこの EPG と通信できるかを規定するラベルが定義されている場所です。また、EPG内のエンドポイントが関連付けられるブリッジ ドメイン、およびそれらが関連付けられる Virtual Machine Manager（VMM）のドメインへの参照が含まれています。VMMにより、2台のVMサーバ間の仮想マシンのモビリティがアプリケーションのダウンタイムなしで即座に可能になります。

この例の最初の EPG は「web1」という名前です。EPG 内の fvRsBd 要素は、関連付けられるブリッジ ドメインを定義します。ブリッジ ドメインは tnFxBDName 属性の値によって識別されます。この EPG は、前述の「ブリッジ ドメイン」の項で名前を付けられた「solarBD1」というブリッジ ドメインに関連付けられます。ブリッジ ドメインへのバインディングは、デフォルト ゲートウェイ アドレスがこの EPG のエンドポイントのためにどうあるべきかを理解するためにシステムによって使用されます。エンドポイントが同じサブネットにあるということや、ブリッジングを介してのみ通信できるという意味ではありません。エンドポイントのパケットがブリッジングまたはルーティングされるかどうかは、送信元エンドポイントがパケットをデフォルト ゲートウェイまたは要求される最後の宛先に送信するかどうかで決定されます。デフォルト ゲートウェイにパケットを送信すると、パケットはルーティングされます。

この EPG で使用される VMM ドメインは fvRsDomAtt タグによって識別されます。この要素は、他の場所で定義された VMM ドメイン オブジェクトを参照します。VMM ドメイン オブジェクトは、その tDn name 属性によって識別されます。この例では、「uni/vmmp-VMware/dom-mininet」と呼ばれる VMM ドメイン 1 個のみを示します。

「web1」EPG の次の要素は、この EPG が提供するコントラクトを定義し、fvRsProv タグによって識別されます。「web1」が複数のコントラクトを提供すると、fvRsProv 要素が複数あります。同様に、それが 1 つ以上のコントラクトを消費すると、fvRsCons 要素があります。

fvRsProv 要素には、提供されているコントラクトの名前である必須属性があります。「web1」は、tDn="uni/tn-coke/brc-webCtrct" と呼ばれる以前に定義されたコントラクト「webCtrct」を提供しています。

次の属性は、matchT 属性です。その属性には、それがサブジェクト ラベルのコントラクト内にあったので、プロバイダーまたはコンシューマのラベルと一致するための同じセマンティックがあります（All、AtLeastOne または None の値を取ることができます）。この条件は、対応するコンシューマ ラベルと比較されるときにプロバイダーのラベルに適用されます。ラベルの一致は、コンシューマとプロバイダーがコントラクトで許可された場合に通信できることを意味します。つまり、コントラクトが通信を許可する必要があり、コンシューマとプロバイダーのラベルがプロバイダーで指定された一致基準を使用して一致する必要があります。

コンシューマには、対応する一致基準がありません。使用される一致タイプはプロバイダーによって常に定められます。

プロバイダー要素 fvRsProv の中で、管理者は使用するラベルを指定する必要があります。2 種類のラベル、プロバイダー ラベルとプロバイダー サブジェクト ラベルがあります。プロバイダー ラベル vzProvLbl は、前述の matchT 基準を使用する他の EPG 内のコンシューマ ラベルに一致させるために使用されます。プロバイダー サブジェクト ラベル vzProvSubjLbl は、コントラクトで指定されるサブジェクト ラベルに一致させるために使用されます。ラベルの唯一の属性は name 属性です。

「web1」EPG では、2 つのプロバイダー サブジェクト ラベル openProv および secureProv が「webCtrct」コントラクトのサブジェクト「http」および「https」と一致するように指定されます。1 つのプロバイダー ラベル「green」が、「App」EPG 内の同じラベルと一致する All の一致基準で指定されます。

この例の次の EPG「web2」は「web1」と似ていますが、vzProvSubjLbl が 1 つだけあり、ラベル自体は異なっています。

3 番目の EPG は「app」と呼ばれるもので、次のように定義されます。

```
<fvAEPg name="app">
    <fvRsBd tnFvBDName="solarBD1" />
    <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
    <fvRsCons tnVzBrCPName="webCtrct">
        <vzConsSubjLbl name="openCons"/>
        <vzConsSubjLbl name="secureCons"/>
        <vzConsLbl name="green"/>
     </fvRsCons>
</fvAEPg>
```

最初の部分は「web1」EPG とほぼ同じです。主な違いは、この EPG は「webCtrct」のコンシューマで、対応するコンシューマ ラベルとコンシューマ サブジェクト ラベルがあることです。構文はほぼ同じですが、タグで「Prov」が「Cons」に置き換えられます。プロバイダーをコンシューマ ラベルに一致させるための一致タイプがプロバイダーで指定されるため、`FvRsCons` 要素に一致属性はありません。

最後の EPG では、純粋なコンシューマであるという点において「db」は「app」EPG と非常によく似ています。

この例では、EPG は単一コントラクトのコンシューマまたはプロデューサであり、EPG が即座に複数のコントラクトのプロデューサおよび複数のコントラクトのコンシューマになることが一般的です。

最後に

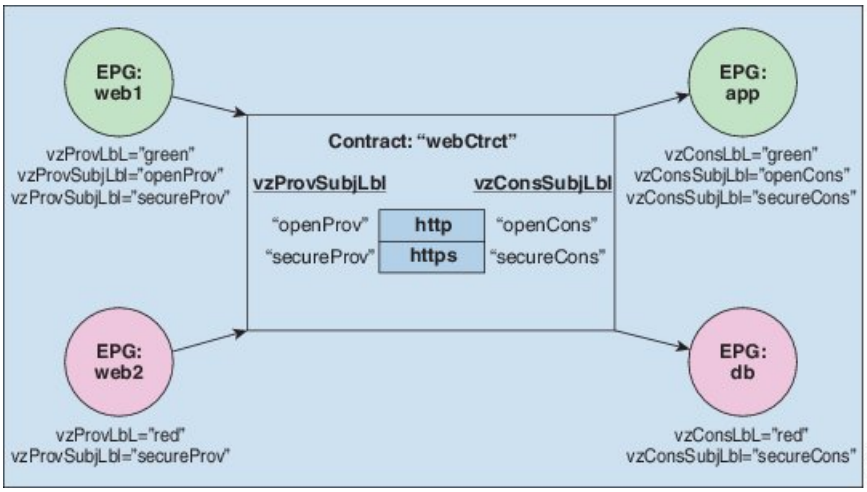
```
        </fvAp>
    </fvTenant>
</polUni>
```

最後の数行でポリシーが完了します。

この例のテナント ポリシーが行うこと

次の図は、コントラクトがエンドポイント グループ（EPG）の通信をどのように管理するかを示します。

図 *4 : EPG/EPG* 通信を決定するラベルとコントラクト



4 つの EPG には、EPG:web1、EPG:web2、EPG:app および EPG:db という名前が付いています。EPG:web1 および EPG:web2 は webCtrct と呼ばれるコントラクトを提供します。EPG:app および EPG:db は、同じコントラクトを消費します。

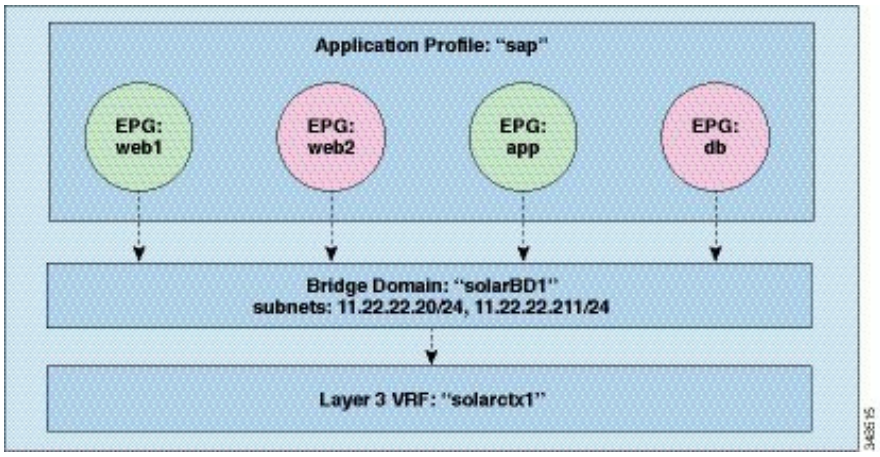
EPG:web1 は EPG:app のみと通信でき、EPG:web2 は EPG:db のみと通信できます。このインタラクションは、プロバイダーおよびコンシューマのラベル「green」と「red」によって制御されます。

EPG:web1 が EPG:app と通信するとき、webCtrct コントラクトが使用されます。EPG:app は、EPG:web1 が提供するコントラクトを消費するので、EPG:web1 への接続を開始できます。

EPG:web1 と EPG:app が通信を行うために使用できるサブジェクトは両方とも http および https です。なぜなら、EPG:web1 にはプロバイダー サブジェクト ラベル「openProv」があり、http サブジェクトにもそれがあるためです。EPG:web1 には、プロバイダー サブジェクト ラベル「secureProv」があり、サブジェクト https も同様です。同様に、EPG:app にはサブジェクト ラベル「openCons」および「secureCons」があり、サブジェクト http および https にもあります。

EPG:web2 が EPG:db と通信するとき、それらは https サブジェクトのみを使用できます。https サブジェクトのみがプロバイダーおよびコンシューマのサブジェクト ラベルを持っているためです。EPG:db は EPG:web2 への TCP 接続を開始できます。なぜなら、EPG:db が EPG:web2 により提供されるコントラクトを消費するからです。

図 *5:* ブリッジ ドメイン、サブネット、およびレイヤ *3 VRF*



この例のポリシーは、EPG、アプリケーション プロファイル、ブリッジ ドメインおよびレイヤ 3 Layer 3 Virtual Routing and Forwarding（VRF）インスタンス間の関係を次のように指定します。EPG の EPG:web1、EPG:web2、EPG:app および EPG:db はすべて、「sap」と呼ばれるアプリケーション プロファイルのメンバーです。

これらの EPG はブリッジ ドメイン「solarBD1」にもリンクされています。solarBD1 には、2 つのサブネット 11.22.22.XX/24 と 11.22.23.XX/24 があります。4 つの EPG 内のエンドポイントは、これら 2 つのサブネット範囲内にある必要があります。これら 2 つのサブネット内のデフォルト ゲートウェイの IP アドレスは 11.22.22.20 と 11.22.23.211 です。solarBD1 ブリッジ ドメインは「solarctx1」レイヤ 3 VRF にリンクされます。

これらのポリシーの詳細はすべて、「solar」というテナントに含まれています。

EPG

AEPまたはインターフェイスポリシーグループを使用したアプリケーション EPG の複数のポートへの導入

APIC の拡張 GUI と REST API を使用して、接続エンティティ プロファイルをアプリケーション EPG に直接関連付けることができます。これにより、単一の構成の接続エンティティ プロファイルに関連付けられたすべてのポートに、関連付けられたアプリケーション EPG を導入します。

APIC REST API または NX-OS スタイルの CLI を使用し、インターフェイス ポリシー グループを介して複数のポートにアプリケーション EPG を導入できます。

REST API を使用した APIC の特定のポートへの EPG の導入

始める前に

EPG を導入するテナントが作成されていること。

手順

特定のポート上に EPG を導入します。

例：

```
<fvTenant name="<tenant_name>" dn="uni/tn-test1" >
    <fvCtx name="<network_name>" pcEnfPref="enforced" knwMcastAct="permit"/>
    <fvBD name="<bridge_domain_name>" unkMcastAct="flood" >
        <fvRsCtx tnFvCtxName="<network_name>"/>
    </fvBD>
    <fvAp name="<application_profile>" >
        <fvAEPg name="<epg_name>" >
           <fvRsPathAtt tDn="topology/pod-1/paths-1017/pathep-[eth1/13]" mode="regular"
 instrImedcy="immediate" encap="vlan-20"/>
        </fvAEPg>
    </fvAp>
</fvTenant>
```

REST API を使用した AEP による複数のインターフェイスへの EPG の導入

AEP のインターフェイス セレクタを使用して、AEPg の複数のパスを設定できます。以下を選択できます。

1. ノードまたはノード グループ
2. インターフェイスまたはインターフェイス グループ

   インターフェイスは、インターフェイス ポリシー グループ（および `infra:AttEntityP`）を使用します。
3. infra:AttEntityP を AEPg に関連付けることで、使用する VLAN を指定する。

   infra:AttEntityP は、VLAN が異なる複数の AEPg に関連付けることができます。

3 のように infra:AttEntityP を AEPg に関連付けた場合、1 で選択したノード上の 2 のインターフェイスに、3 で指定した VLAN を使用して AEPg が導入されます。

この例では、AEPg `uni/tn-Coke/ap-AP/epg-EPG1` が、ノード 101 および 102 のインターフェイス 1/10、1/11、および 1/12 に vlan-102 で導入されます。

始める前に

- ターゲット アプリケーション EPG（AEPg）を作成する。
- 接続エンティティ プロファイル（AEP）による EPG 導入に使用する VLAN の範囲が含まれている VLAN プールを作成する。
- 物理ドメインを作成して VLAN プールおよび AEP にリンクさせる。

手順

選択したノードとインターフェイスに AEPg を導入するには、次の例のような XML を POST 送信します。

例：

```
 <infraInfra dn="uni/infra">
   <infraNodeP name="NodeProfile">
       <infraLeafS name="NodeSelector" type="range">
          <infraNodeBlk name="NodeBlok" from_="101" to_="102"/>
          <infraRsAccPortP tDn="uni/infra/accportprof-InterfaceProfile"/>
        </infraLeafS>
   </<infraNodeP>

   <infraAccPortP name="InterfaceProfile">
      <infraHPortS name="InterfaceSelector" type="range">
          <infraPortBlk name=" InterfaceBlock" fromCard="1" toCard="1" fromPort="10"
toPort="12"/>
          <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-PortGrp" />
      </infraHPortS>
   </infraAccPortP>

   <infraFuncP>
       <infraAccPortGrp name="PortGrp">
           <infraRsAttEntP  tDn="uni/infra/attentp-AttEntityProfile"/>
       </infraAccPortGrp>
   </infraFuncP>

   <infraAttEntityP name="AttEntityProfile" >
       <infraGeneric name="default" >
          <infraRsFuncToEpg tDn="uni/tn-Coke/ap-AP/epg-EPG1" encap="vlan-102"/>
       </infraGeneric>
   </infraAttEntityP>
</infraInfra>
```

REST API を使用した、EPG を特定のポートに導入するための AEP、ドメイン、および VLAN の作成

始める前に

- EPG を導入するテナントがすでに作成されていること。

• EPG は特定のポートに静的に導入されます。

手順

ステップ 1 インターフェイス プロファイル、スイッチ プロファイル、および接続エンティティ プロファイル（AEP）を作成します。

例：

```
   <infraInfra>

     <infraNodeP  name="<switch_profile_name>" dn="uni/infra/nprof-<switch_profile_name>"
 >
           <infraLeafS  name="SwitchSeletor" descr="" type="range">
                <infraNodeBlk name="nodeBlk1" descr="" to_="1019" from_="1019"/>
           </infraLeafS>
          <infraRsAccPortP tDn="uni/infra/accportprof-<interface_profile_name>"/>
      </infraNodeP>

      <infraAccPortP name="<interface_profile_name>"
dn="uni/infra/accportprof-<interface_profile_name>" >
           <infraHPortS  name="portSelector"  type="range">
               <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-<port_group_name>"
 fexId="101"/>
               <infraPortBlk name="block2"  toPort="13" toCard="1" fromPort="11"
fromCard="1"/>
         </infraHPortS>
     </infraAccPortP>

    <infraAccPortGrp  name="<port_group_name>"
dn="uni/infra/funcprof/accportgrp-<port_group_name>" >
          <infraRsAttEntP tDn="uni/infra/attentp-<attach_entity_profile_name>"/>
          <infraRsHIfPol tnFabricHIfPolName="1GHifPol"/>
    </infraAccPortGrp>

    <infraAttEntityP  name="<attach_entity_profile_name>"
dn="uni/infra/attentp-<attach_entity_profile_name>" >
         <infraRsDomP tDn="uni/phys-<physical_domain_name>"/>
    </infraAttEntityP>

<infraInfra>
```

ステップ 2 ドメインを作成する。

例：

```
<physDomP  name="<physical_domain_name>" dn="uni/phys-<physical_domain_name>">
    <infraRsVlanNs tDn="uni/infra/vlanns-[<vlan_pool_name>]-static"/>
</physDomP>
```

ステップ 3 VLAN 範囲を作成します。

例：

```
<fvnsVlanInstP  name="<vlan_pool_name>" dn="uni/infra/vlanns-[<vlan_pool_name>]-static"
  allocMode="static">
    <fvnsEncapBlk name="" descr="" to="vlan-25" from="vlan-10"/>
</fvnsVlanInstP>
```

ステップ 4 ドメインに EPG を関連付けます。

例：

```
<fvTenant  name="<tenant_name>" dn="uni/tn-" >
    <fvAEPg prio="unspecified" name="<epg_name>" matchT="AtleastOne"
dn="uni/tn-test1/ap-AP1/epg-<epg_name>" descr="">
        <fvRsDomAtt tDn="uni/phys-<physical_domain_name>" instrImedcy="immediate"
resImedcy="immediate"/>
    </fvAEPg>
</fvTenant>
```

EPG 内の分離

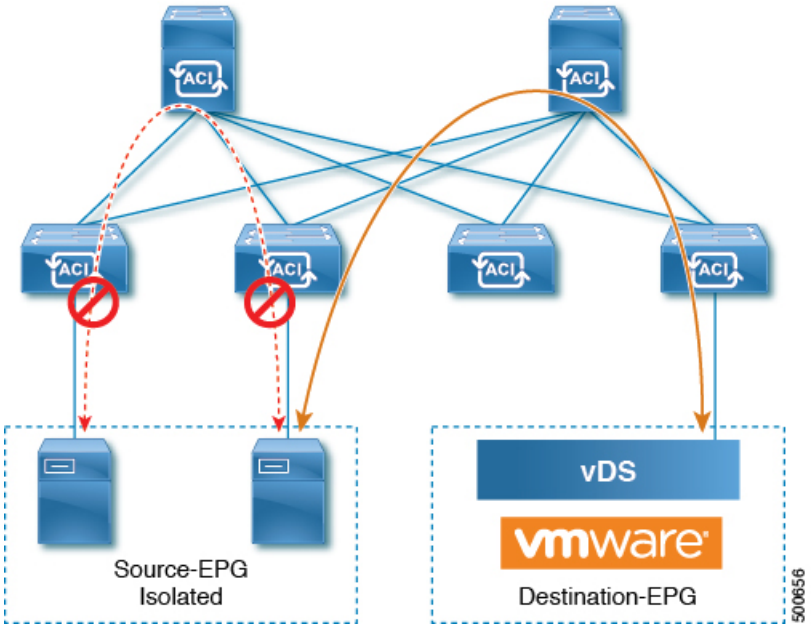
ベア メタル サーバの EPG 内分離

EPG 内エンドポイント分離のポリシーは、ベア メタル サーバなどの直接接続されているエンドポイントに適用できます。

次のような使用例があります。

- バックアップ クライアントは、バックアップ サービスにアクセスするための通信要件は同じですが、相互に通信する必要はありません。
- ロードバランサの背後にあるサーバの通信要件は同じですが、それらのサーバを相互に分離すると、不正アクセスや感染のあるサーバに対して保護されます。

図 *6*: ベア メタル サーバの *EPG* 内分離



ベア メタル の EPG 分離はリーフ スイッチで適用されます。ベア メタル サーバは VLAN カプセル化を使用します。ユニキャスト、マルチキャスト、およびブロードキャストのすべてのトラフィックが、分離が適用された EPG 内でドロップ（拒否）されます。ACI ブリッジ ドメインには、分離された EPG と通常の EPG を混在させることができます。分離された EPG それぞれには、VLAN 間トラフィックを拒否する複数の VLAN を指定できます。

REST API を使用したベア メタル サーバのイントラ EPG 分離の設定

始める前に

EPG が使用するポートは、物理ドメイン内のベア メタル サーバ インターフェイスに関連付けられている必要があります。

手順

ステップ 1 XML API を使用してアプリケーションを展開するには、次の HTTP POST メッセージを送信します。

例：

```
POST https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml
```

ステップ 2 次の XML 構造を POST メッセージの本文に含めます。

例：

```
<fvTenant name="Tenant_BareMetal" >
  <fvAp name="Web">
    <fvAEPg name="IntraEPGDeny" pcEnfPref="enforced">
          <!-- pcEnfPref="enforced" ENABLES ISOLATION-->
          <fvRsBd tnFvBDName="bd" />
          <fvRsDomAtt tDn="uni/phys-Dom1" />
          <!-- PATH ASSOCIATION -->
          <fvRsPathAtt tDn="topology/pod-1/paths-1017/pathep-[eth1/2]" encap="vlan-51"
primaryEncap="vlan-100" instrImedcy='immediate'/>
    </fvAEPg>
 </fvAp>
</fvTenant>
```

VMware VDS または Microsoft vswitch の EPG 内の分離

EPG 内分離は、同じベース EPG または uSeg EPG 内の物理エンドポイント デバイスまたは仮想エンドポイントデバイスが相互に通信しないようにするオプションです。デフォルトでは、同じEPGに含まれるエンドポイントデバイスは互いに通信することができます。しかし、EPG 内のエンドポイント デバイスの別のエンドポイント デバイスからの完全な分離が望ましい状況が存在します。たとえば、同じ EPG 内のエンドポイント VM が複数のテナントに属している場合、またはウイルスが広がるのを防ぐために、EPG内の分離を実行することができます。

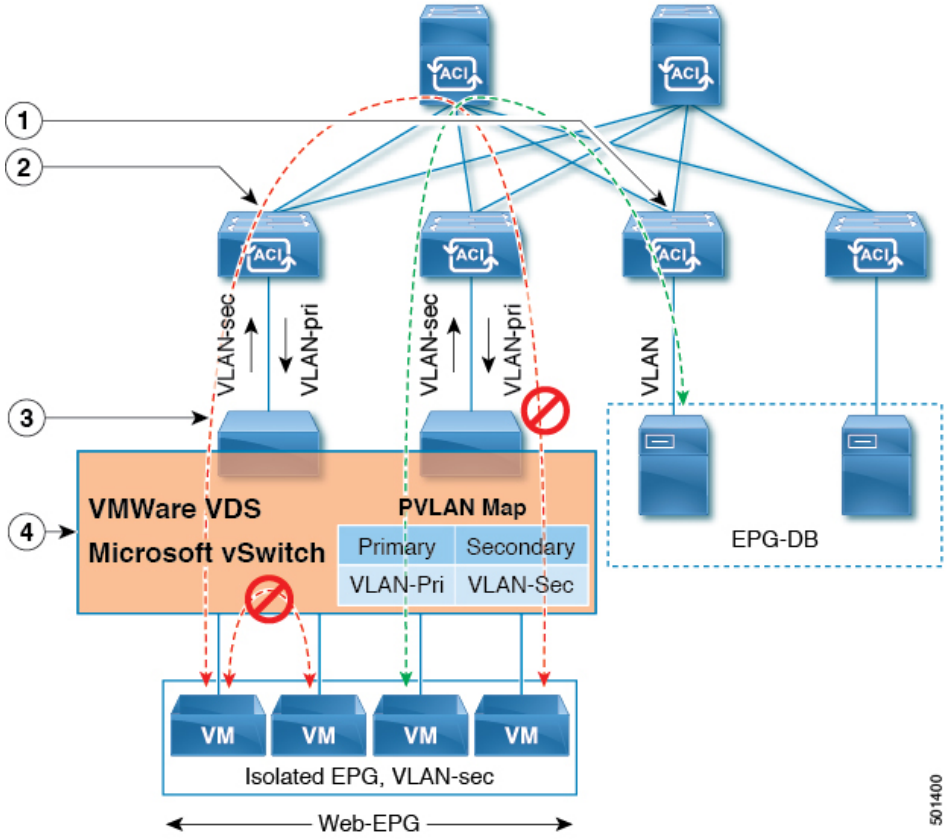
Cisco ACI 仮想マシン マネージャ（VMM）ドメインは、EPG 内分離が有効である EGP ごとに、分離された PVLAN ポート グループを VMware VDS または Microsoft vSwitch に作成します。ファブリック管理者がプライマリ カプセル化を指定するか、または EPG と VMM ドメインの関連付け時にファブリックが動的にプライマリ カプセル化を指定します。ファブリック管理者が VLAN pri 値とVLAN-sec 値を静的に選択すると、VMM ドメインによって VLAN-pri と VLAN-sec がドメイン プール内のスタティック ブロックの一部であることが検証されます。

（注） イントラ EPG 隔離が強制されない場合、設定で指定されていても VLAN-pri 値は無視されます。

VMware VDS または Microsoft vSwitch の VLAN pri と VLAN-sec のペアは、EPG とドメインの関連付け時に VMM 単位で選択されます。EPG 内隔離 EPG に作成されたポート グループは PVLAN に設定されたタイプでタグ付けされた VLAN-sec を使用します。VMware VDS または Microsoft vSwitch とファブリックは、VLAN-pri/VLAN-sec カプセル化を切り替えます。

- Cisco ACI ファブリックから VMware VDS または Microsoft vSwitch への通信では VLAN pri を使用します。
- VMware VDS または Microsoft vSwitch から Cisco ACI ファブリックへの通信は VLAN-sec を使用します。

図 *7 : VMware VDS* または *Microsoft vswitch* の *EPG* 内の分離



この図に関する次の詳細に注意してください。

1. EPG-DB は VLAN トラフィックを Cisco ACI リーフ スイッチに送信します。Cisco ACI の出力リーフ スイッチは、プライマリ VLAN (PVLAN) タグでトラフィックをカプセル化し、それを Web-EPG のエンドポイントに転送します。

2. VMware VDS または Microsoft vSwitch は VLAN-sec を使用して Cisco ACI リーフ スイッチにトラフィックを送信します。Cisco ACI リーフ スイッチは、Web-EPG 内のすべての VLAN-sec 内トラフィックに分離が適用されているため、すべての EPG 内トラフィックをドロップします。

3. Cisco ACI リーフへの VMware VDS または Microsoft vSwitch VLAN-sec アップリンクは分離トランク モードです。Cisco ACI リーフ スイッチは VMware VDS または Microsoft vSwitch へのダウンリンク トラフィックに VLAN-pri を使用します。

4. VMware VDS または Microsoft vSwitch および Cisco ACI リーフ スイッチに PVLAN マップが設定されます。WEB-EPG からのVMトラフィックは VLAN-sec 内でカプセル化されます。VMware VDS または Microsoft vSwitch は PVLAN タグに従ってローカルの WEB 内 EPG VM トラフィックを拒否します。ESXi 内ホストまたは Microsoft Hyper-V ホストのすべての VM トラフィックは、VLAN-Sec を使用して Cisco ACI リーフに送信されます。

関連項目

Cisco AVS 環境での EPG 内分離の設定については、Cisco AVS の EPG 内分離の適用 （27 ページ）を参照してください。

REST API を使用した VMware VDS または Microsoft vswitch の EPG 間分離の設定

手順

ステップ 1 XML API を使用してアプリケーションを展開するには、次の HTTP POST メッセージを送信します。

例：

```
POST https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml
```

ステップ 2 VMware VDS または Microsoft vSwitch 展開については、POST メッセージの本文に次の XML 構造のいずれかが含まれます。

例：

次の例は、 Vmware VDS 用です。

```
<fvTenant name="Tenant_VMM" >
  <fvAp name="Web">
    <fvAEPg name="IntraEPGDeny" pcEnfPref="enforced">
          <!-- pcEnfPref="enforced" ENABLES ISOLATION-->
          <fvRsBd tnFvBDName="bd" />
          <!-- STATIC ENCAP ASSOCIATION TO VMM DOMAIN-->
          <fvRsDomAtt encap="vlan-2001" instrImedcy="lazy" primaryEncap="vlan-2002"
resImedcy="immediate" tDn="uni/vmmp-VMware/dom-DVS1”>
    </fvAEPg>
 </fvAp>
</fvTenant>
```

例：

次の例は、 Microsoft vSwitch 用です。

```
<fvTenant name="Tenant_VMM" >
  <fvAp name="Web">
    <fvAEPg name="IntraEPGDeny" pcEnfPref="enforced">
          <!-- pcEnfPref="enforced" ENABLES ISOLATION-->
          <fvRsBd tnFvBDName="bd" />
         <!-- STATIC ENCAP ASSOCIATION TO VMM DOMAIN-->
       <fvRsDomAtt tDn="uni/vmmp-Microsoft/dom-domain1”>
 <fvRsDomAtt encap="vlan-2004" instrImedcy="lazy" primaryEncap="vlan-2003"
 resImedcy="immediate" tDn="uni/vmmp-Microsoft/dom-domain2”>
   </fvAEPg>
 </fvAp>
</fvTenant>
```

Cisco AVS の EPG 内分離の適用

デフォルトでは、EPGに属するエンドポイントは契約が設定されていなくても相互に通信できます。ただし、相互に、EPG 内のエンドポイントを特定できます。EPG 内でエンドポイント分離を適用することで、ウィルスやその他の問題がある VM が EPG 内の他の VM に影響を与えないようにすることができる場合もあります。

アプリケーション内のすべてのエンドポイントに分離を設定することも、いずれにも設定しないこともできます。一部のエンドポイントに分離を設定し、他のエンドポイントに設定しない方法は使用できません。

EPG 内のエンドポイントを分離しても、エンドポイントが別の EPG 内のエンドポイントと通信できるようにするコントラクトには影響しません。

EPG が VLAN モードの Cisco AVS ドメインと関連付けられている場合は、EPG 内のエンドポイントの分離によって障害が発生します。

（注） 内部 EPG で、Cisco AVS microsegment（uSeg）EPG の分離は現在サポートされていません。2 個の EPG 間に存在するすべてのコントラクトに関係なく、内部 EPG 分離いずれかが適用された場合、別の uSeg Epg 内に存在する 2 つのエンドポイント間の通信が可能です。

REST API を使用した Cisco AVS の EPG 内分離の設定

始める前に

Cisco AVS が VXLAN モードであることを確認します。

手順

ステップ 1 XML API を使用してアプリケーションを展開するには、次の HTTP POST メッセージを送信します。

例：

```
   POST
https://192.0.20.123/api/mo/uni/tn-ExampleCorp.xml
```

ステップ 2 VMM の導入では、POST メッセージの本文に次の例に示す XML 構造を含めます。

例：

```
Example:
<fvTenant name="Tenant_VMM" >
  <fvAp name="Web">
    <fvAEPg name="IntraEPGDeny" pcEnfPref="enforced">
      <!-- pcEnfPref="enforced" ENABLES ISOLATION-->
      <fvRsBd tnFvBDName="bd" />
      <fvRsDomAtt encap="vlan-2001" tDn="uni/vmmp-VMware/dom-DVS1”>
    </fvAEPg>
```

```
  </fvAp>
</fvTenant>
```

次のタスク

統計情報を選択して表示すると、エンドポイントが関与する問題の診断に役立ちます。「*Cisco AVS* 設定ガイド」または「*Cisco APIC* レイヤ *2* 設定ガイド」の「統計情報を選択して分離されたエンドポイントを表示する」および「分離されたエンドポイントの統計情報を表示する」セクションを参照してください。

マイクロセグメンテーション

ベア メタルでのネットワークベースの属性によるマイクロセグメンテーションの使用

Cisco APIC を使用して Cisco ACI でのマイクロセグメンテーションを設定し、ネットワークベースの属性、MAC アドレス、または 1 つ以上の IP アドレスを使用した新しい属性ベースの EPG を作成できます。ネットワークベースの属性を使用して Cisco ACI でのマイクロセグメンテーションを設定し、単一のベース EPG または複数の EPG 内で VM または物理エンドポイントを分離できます。

IP ベースの属性の使用

IP ベースのフィルタを使用して、単一のマイクロセグメントで単一 IP アドレス、サブネット、または多様な非連続 IP アドレスを分離できます。ファイアウォールの使用と同様に、セキュリティ ゾーンを作成するための迅速かつ簡単な方法として、IP アドレスに基づいて物理エンドポイントを分離できます。

MAC ベースの属性の使用

MAC ベースのフィルタを使用して、単一 MAC アドレスまたは複数の MAC アドレスを分離できます。不適切なトラフィックをネットワークに送信するサーバがある場合はこの方法を推奨します。MAC ベースのフィルタを使用してマイクロセグメントを作成することで、このサーバを分離できます。

REST API を使用した共有リソースとしての IP ベースのマイクロセグメント EPG の設定

VRF および現在のファブリック外のクライアントがアクセス可能な共有サービスとして、32 ビット マスクの IP アドレスを持つマイクロセグメント EPG を設定できます。

手順

共有サブネットを持つ IP アドレス属性のマイクロセグメント EPG（epg3）を設定するには、IP アドレスと 32 ビット マスクを使用して、次の例のような XML を POST 送信します。IP 属性の **usefvSubnet** は「yes」に設定します。

例：

```
<fvAEPg descr="" dn="uni/tn-t0/ap-a0/epg-epg3" fwdCtrl=""
          isAttrBasedEPg="yes" matchT="AtleastOne" name="epg3" pcEnfPref="unenforced"
prefGrMemb="exclude"prio="unspecified">
        <fvRsCons prio="unspecified" tnVzBrCPName="ip-epg"/>
        <fvRsNodeAtt descr="" encap="unknown" instrImedcy="immediate" mode="regular"
tDn="topology/pod-2/node-106"/>
        <fvSubnet ctrl="" descr="" ip="56.4.0.2/32" name="" preferred="no"
           scope="public,shared" virtual="no"/>
        <fvRsDomAtt classPref="encap" delimiter="" encap="unknown" encapMode="auto"
instrImedcy="immediate"
           primaryEncap="unknown" resImedcy="immediate" tDn="uni/phys-vpc"/>
        <fvRsCustQosPol tnQosCustomPolName=""/>
        <fvRsBd tnFvBDName="b2"/>
        <fvCrtrn descr="" match="any" name="default" ownerKey="" ownerTag="" prec="0">
           <fvIpAttr descr="" ip="1.1.1.3" name="ipv4" ownerKey="" ownerTag=""
usefvSubnet="yes”/>
        </fvCrtrn>
        <fvRsProv matchT="AtleastOne" prio="unspecified" tnVzBrCPName="ip-epg"/>
        <fvRsProv matchT="AtleastOne" prio="unspecified" tnVzBrCPName="shared-svc"/>
</fvAEPg>
```

REST API を使用したベアメタル環境でのネットワークベースのマイクロセグメント EPG の設定

ここでは、REST API を使用してベアメタル環境の Cisco ACI でネットワーク属性のマイクロセグメンテーションを設定する方法について説明します。

手順

ステップ 1 Cisco APIC にログインします。

ステップ 2 https://*apic-ip-address*/api/node/mo/.xml にポリシーをポストします。

例：

A：次の例では、IP ベースの属性を使用して 41-subnet という名前のマイクロセグメントを設定します。

```
<polUni>
  <fvTenant dn="uni/tn-User-T1" name="User-T1">
      <fvAp dn="uni/tn-User-T1/ap-Base-EPG" name="Base-EPG">
            <fvAEPg dn="uni/tn-User-T1/ap-Base-EPG/epg-41-subnet" name="41-subnet"
pcEnfPref="enforced” isAttrBasedEPg="yes" >
               <fvRsBd tnFvBDName="BD1" />
               <fvCrtrn name="Security1">
```

```
                <fvIpAttr name="41-filter" ip="12.41.0.0/16"/>
              </fvCrtrn>
            </fvAEPg>
          </fvAp>
    </fvTenant>
</polUni>
```

例：

次の例は、例 A のベース EPG です。

```
<polUni>
  <fvTenant dn="uni/tn-User-T1" name="User-T1">
      <fvAp dn="uni/tn-User-T1/ap-Base-EPG" name="Base-EPG">
          <fvAEPg dn="uni/tn-User-T1/ap-Base-EPG/baseEPG” name=“baseEPG”
pcEnfPref="enforced” >
                <fvRsBd tnFvBDName="BD1" />
            </fvAEPg>
          </fvAp>
    </fvTenant>
</polUni>
```

例：

B：次の例では、MAC ベースの属性を使用して useg-epg という名前のマイクロセグメントを設定します。

```
<polUni>
  <fvTenant name="User-T1">
        <fvAp name="customer">
            <fvAEPg name="useg-epg" isAttrBasedEPg="true">
              <fvRsBd tnFvBDName="BD1"/>
              <fvRsDomAtt  instrImedcy="immediate" resImedcy="immediate"
tDn="uni/phys-phys" />
            <fvRsNodeAtt tDn="topology/pod-1/node-101" instrImedcy="immediate" />
            <fvCrtrn name="default">
                <fvMacAttr name="mac" mac="00:11:22:33:44:55" />
              </fvCrtrn>
            </fvAEPg>
        </fvAp>
  </fvTenant>
</polUni>
```

仮想スイッチでのマイクロセグメンテーションの設定

Cisco Application Centric Infrastructure（ACI）マイクロセグメンテーションは、さまざまなネットワークベースまたは仮想マシン（VM）ベース属性に基づき、エンドポイント グループ（EPG）と呼ばれるロジカル セキュリティ ゾーンにエンドポイントを自動的に割り当てることができます。このセクションでは、仮想スイッチのマイクロセグメント（uSeg）EPGを設定する方法について説明します。

Cisco ACI マイクロセグメンテーションでは、次に接続されている仮想エンドポイントのサポートを提供します。

- VMware vSphere Distributed Switch（vDS）
- Cisco Application Virtual Switch（AVS）

- Microsoft vSwitch

マイクロセグメンテーションと Cisco ACI の動作、前提条件、ガイドライン、およびシナリオについての詳細は「*Cisco ACI 仮想化ガイド*」を参照してください。

REST API を使用した Cisco ACI でのマイクロセグメンテーションの設定

This section describes how to configure Microsegmentation with Cisco ACI for Cisco ACI Virtual Edge, Cisco AVS, VMware VDS, or Microsoft vSwitch using the REST API.

手順

ステップ 1 Cisco APIC にログインします。

ステップ 2 `Https://apic-ip-address/api/node/mo/.xml.apic-ip-address/api/node/mo/.xml.` にポリシーをポストします。

例：

この例では、すべての属性と EPG 一致設定 1 と一致する状態で、「vm」を含む属性 VM 名と「CentOS」および「Linux」の値を含むオペレーティング システム属性を持つ uSeg EPG を設定します。

```
<fvAEPg name="Security" isAttrBasedEPg="yes" pcEnfPref="unenforced" status="">
      <fvRsBd tnFvBDName="BD1" />
      <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet”/>
       <fvCrtrn name="default" match=“all” prec=“1”>
                   <fvVmAttr name="foo" type="vm-name" operator="contains" value="vm"/>
                <fvSCrtrn name="sub-def" match=“any”>
                         <fvVmAttr name="foo1" type="guest-os" operator="contains"
value="CentOS"/>
                         <fvVmAttr name="foo2" type="guest-os" operator="contains"
value="Linux"/>
                    </fvSCrtrn>
       </fvCrtrn>
 </fvAEPg>
```

例：

この例では、アプリケーション EPG のマイクロセグメンテーションが有効になっています。

```
<polUni>
  <fvTenant dn="uni/tn-User-T1" name="User-T1">
      <fvAp dn="uni/tn-User-T1/ap-Application-EPG" name="Application-EPG">
          <fvAEPg dn="uni/tn-User-T1/ap-Application-EPG/applicationEPG”
name=“applicationEPG” pcEnfPref="enforced” >
               <fvRsBd tnFvBDName="BD1" />
               <fvRsDomAtt tDn="uni/vmmp-VMware/dom-cli-vmm1" classPref=“useg”/>
           </fvAEPg>
        </fvAp>
   </fvTenant>
</polUni>
```

上記の例では、文字列 `<fvRsDomAtt tDn="uni/vmmp-VMware/dom-cli-vmm1" classPref=“useg”/>` が、Cisco ACI Virtual Edge, Cisco AVS、Microsoft vSwitchではなく、VMware VDS のみに関連します。

例：

この例では、uSeg EPG を Cisco ACI Virtual Edge VNM ドメインに接続してスイッチング モードを追加します。

```
<fvRsDomAtt resImedcy="immediate" instrImedcy="immediate" switchingMode="AVE"
encapMode="auto" tDn="uni/vmmp-VMware/dom-AVE-CISCO" primaryEncapInner=""
secondaryEncapInner=""/>
```

Application Profiles

Three-Tier アプリケーションの展開

フィルタは、フィルタを含むコントラクトにより許可または拒否されるデータ プロトコルを指定します。コントラクトには、複数のサブジェクトを含めることができます。サブジェクトは、単方向または双方向のフィルタを実現するために使用できます。単方向フィルタは、コンシューマからプロバイダー（IN）のフィルタまたはプロバイダーからコンシューマ（OUT）のフィルタのどちらか一方向に使用されるフィルタです。双方向フィルタは、両方の方向で使用される同一フィルタです。これは、再帰的ではありません。

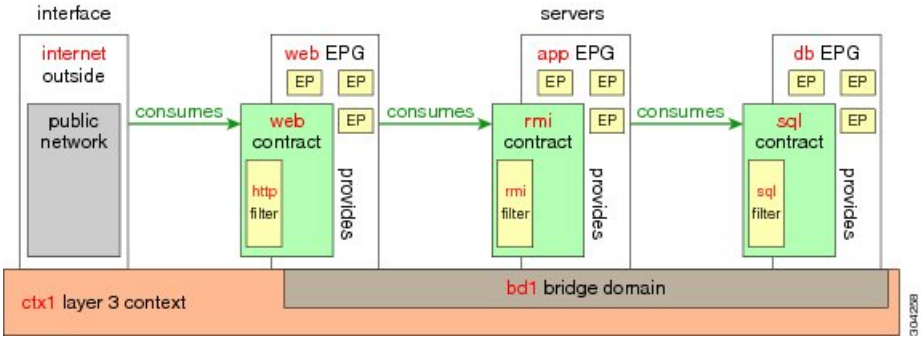
コントラクトは、エンドポイント グループ間 (EPG 間) の通信をイネーブルにするポリシーです。これらのポリシーは、アプリケーション層間の通信を指定するルールです。コントラクトが EPG に付属していない場合、EPG 間の通信はデフォルトでディセーブルになります。EPG 内の通信は常に許可されているので、EPG 内の通信には契約は必要ありません。

アプリケーション プロファイルでは、APIC がその後ネットワークおよびデータセンターのインフラストラクチャで自動的にレンダリングするアプリケーション要件をモデル化することができます。アプリケーションプロファイルでは、管理者がインフラストラクチャの構成要素ではなくアプリケーションの観点から、リソースプールにアプローチすることができます。アプリケーション プロファイルは、互いに論理的に関連する EPG を保持するコンテナです。EPG は同じアプリケーション プロファイル内の他の EPG および他のアプリケーション プロファイル内の EPG と通信できます。

アプリケーション ポリシーを展開するには、必要なアプリケーション プロファイル、フィルタ、および契約を作成する必要があります。通常、APIC ファブリックは、テナント ネットワーク内の Three-Tier アプリケーションをホストします。この例では、アプリケーションは 3 台のサーバ（Web サーバ、アプリケーション サーバ、およびデータベース サーバ）を使用して実行されます。Three-Tier アプリケーションの例については、次の図を参照してください。

Web サーバには HTTP フィルタがあり、アプリケーション サーバには Remote Method Invocation（RMI）フィルタがあり、データベース サーバには Structured Query Language（SQL）フィルタがあります。アプリケーション サーバは、SQL コントラクトを消費してデータベース サーバと通信します。Web サーバは、RMI コントラクトを消費して、アプリケーション サーバと通信します。トラフィックは Web サーバから入り、アプリケーション サーバと通信します。アプリケーション サーバはその後、データベース サーバと通信し、トラフィックは外部に通信することもできます。

図 *8 : Three-Tier* アプリケーションの図



http 用のフィルタを作成するパラメータ

この例での http 用のフィルタを作成するパラメータは次のとおりです。

パラメータ名	**http** のフィルタ
名前	http
エントリの数	2
エントリ名	Dport-80 Dport-443
Ethertype	IP
プロトコル	tcp tcp
宛先ポート	http https

rmi および sql 用のフィルタを作成するパラメータ

この例での rmi および sql 用のフィルタを作成するパラメータは次のとおりです。

パラメータ名	**rmi** のフィルタ	**sql** のフィルタ
名前	rmi	sql
エントリの数	1	1
エントリ名	Dport-1099	Dport-1521
Ethertype	IP	IP
プロトコル	tcp	tcp

パラメータ名	rmi のフィルタ	sql のフィルタ
宛先ポート	1099	1521

REST API を使用したアプリケーション プロファイルの展開

EPG が使用するポートは、VM マネージャ (VMM) ドメインまたは EPG に関連付けられた物理ドメインのいずれか 1 つに属している必要があります。

手順

ステップ 1 XML API を使用してアプリケーションを展開するには、次の HTTP POST メッセージを送信します。

例：

```
POST https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml
```

ステップ 2 次の XML 構造を POST メッセージの本文に含めます。

例：

```
<fvTenant name="ExampleCorp">

    <fvAp name="OnlineStore">
        <fvAEPg name="web">
            <fvRsBd tnFvBDName="bd1"/>
            <fvRsCons tnVzBrCPName="rmi"/>
            <fvRsProv tnVzBrCPName="web"/>
            <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"delimiter=@/>
        </fvAEPg>

        <fvAEPg name="db">
            <fvRsBd tnFvBDName="bd1"/>
            <fvRsProv tnVzBrCPName="sql"/>
            <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"/>
        </fvAEPg>

        <fvAEPg name="app">
            <fvRsBd tnFvBDName="bd1"/>
            <fvRsProv tnVzBrCPName="rmi"/>
            <fvRsCons tnVzBrCPName="sql"/>
            <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"/>
        </fvAEPg>
    </fvAp>

<vzFilter name="http" >
<vzEntry dFromPort="80" name="DPort-80" prot="tcp" etherT="ip"/>
<vzEntry dFromPort="443" name="DPort-443" prot="tcp" etherT="ip"/>
</vzFilter>
<vzFilter name="rmi" >
<vzEntry dFromPort="1099" name="DPort-1099" prot="tcp" etherT="ip"/>
</vzFilter>
<vzFilter name="sql">
<vzEntry dFromPort="1521" name="DPort-1521" prot="tcp" etherT="ip"/>
</vzFilter>
    <vzBrCP name="web">
```

```
        <vzSubj name="web">
            <vzRsSubjFiltAtt tnVzFilterName="http"/>
        </vzSubj>
    </vzBrCP>

    <vzBrCP name="rmi">
        <vzSubj name="rmi">
            <vzRsSubjFiltAtt tnVzFilterName="rmi"/>
        </vzSubj>
    </vzBrCP>

    <vzBrCP name="sql">
        <vzSubj name="sql">
           <vzRsSubjFiltAtt tnVzFilterName="sql"/>
        </vzSubj>
    </vzBrCP>
</fvTenant>
```

文字列内の **fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"delimiter=@/**, **delimiter=@** はオプションです。デリミタを入力しなかった場合、システムはデフォルトのデリミタである | を使用します。

XML 構造の最初の行は、ExampleCorp という名前のテナントを変更するかまたは必要に応じて作成します。

```
<fvTenant name="ExampleCorp">
```

次の行は、OnlineStore という名前のアプリケーション ネットワーク プロファイルを作成します。

```
<fvAp name="OnlineStore">
```

アプリケーション ネットワーク プロファイル内の要素は、3 つのエンドポイント グループを作成します（3 台のサーバそれぞれに 1 つずつ）。以降の行は、web という名前のエンドポイント グループを作成し、既存の bd1 という名前のブリッジ ドメインに関連付けます。このエンドポイント グループは、rmi という名前のバイナリ契約で許可されたトラフィックのコンシューマまたは宛先であり、webという名前のバイナリ契約で許可されたトラフィックのプロバイダーまたは送信元です。エンドポイント グループは、datacenter という名前の VMM ドメインに関連付けられます。

```
<fvAEPg name="web">
    <fvRsBd tnFvBDName="bd1"/>
    <fvRsCons tnVzBrCPName="rmi"/>
    <fvRsProv tnVzBrCPName="web"/>
    <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"/>
</fvAEPg>
```

残りの 2 つ、アプリケーション サーバとデータベース サーバのためのエンドポイント グループも同様の方法で作成されます。

以降の行は、HTTP（ポート 80）および HTTPS（ポート 443）という種類の TCP トラフィックを指定する、http という名前のトラフィック フィルタを定義します。

```
<vzFilter name="http" >
<vzEntry dFromPort="80" name="DPort-80" prot="tcp" etherT="ip"/>
<vzEntry dFromPort="443" name="DPort-443" prot="tcp" etherT="ip"/>
</vzFilter>
```

残りの 2 つ、アプリケーション データおよびデータベース（sql）データのためのフィルタも同様の方法で作成されます。

次の行は、http という名前のフィルタを組み込む web という名前のバイナリ契約を作成します。

```
<vzBrCP name="web">
    <vzSubj name="web">
        <vzRsSubjFiltAtt tnVzFilterName="http"/>
    </vzSubj>
</vzBrCP>
```

残りの2つの契約は、rmi および sql のデータ プロトコルに対し、同様の方法で作成されます。

最後の行で構造を閉じます。

```
</fvTenant>
```

契約、タブー契約は、および優先グループ

セキュリティ ポリシーの適用

トラフィックは前面パネルのインターフェイスからリーフ スイッチに入り、パケットは送信元 EPG の EPG でマーキングされます。リーフ スイッチはその後、テナント エリア内のパケットの宛先 IP アドレスでフォワーディング ルックアップを実行します。ヒットすると、次のシナリオのいずれかが発生する可能性があります。

1. ユニキャスト（/32）ヒットでは、宛先エンドポイントの EPG と宛先エンドポイントが存在するローカル インターフェイスまたはリモート リーフ スイッチの VTEP IP アドレスが提供されます。
2. サブネット プレフィクス（/32 以外）のユニキャスト ヒットでは、宛先サブネット プレフィクスの EPG と宛先サブネット プレフィクスが存在するローカル インターフェイスまたはリモート リーフ スイッチの VTEP IP アドレスが提供されます。
3. マルチキャスト ヒットでは、ファブリック全体の VXLAN カプセル化とマルチキャスト グループの EPG で使用するローカル レシーバのローカル インターフェイスと外側の宛先 IP アドレスが提供されます。

（注） マルチキャストと外部ルータのサブネットは、入力リーフ スイッチでのヒットを常にもたらします。セキュリティ ポリシーの適用は、宛先 EPG が入力リーフ スイッチによって認識されるとすぐに発生します。

転送テーブルの誤りにより、パケットがスパイン スイッチの転送プロキシに送信されます。転送プロキシはその後、転送テーブル検索を実行します。これが誤りである場合、パケットはドロップされます。これがヒットの場合、パケットは宛先エンドポイントを含む出力リーフ スイッチに送信されます。出力リーフ スイッチが宛先の EPG を認識するため、セキュリティ ポリシーの適用が実行されます。出力リーフ スイッチは、パケット送信元の EPG を認識する必要があります。ファブリック ヘッダーは、入力リーフ スイッチから出力リーフ スイッチに EPG を伝送するため、このプロセスをイネーブルにします。スパイン スイッチは、転送プロキシ機能を実行するときに、パケット内の元の EPG を保存します。

出力リーフ スイッチでは、送信元 IP アドレス、送信元 VTEP、および送信元 EPG 情報は、学習によってローカルの転送テーブルに保存されます。ほとんどのフローが双方向であるため、応答パケットがフローの両側で転送テーブルに入力し、トラフィックが両方向で入力フィルタリングされます。

契約とタブー契約

セキュリティ ポリシー仕様を含むコントラクト

ACI セキュリティ モデルでは、コントラクトに EPG 間の通信を管理するポリシーが含まれます。コントラクトは通信内容を指定し、EPG は通信の送信元と宛先を指定します。コントラクトは次のように EPG をリンクします。

EPG 1 ---------- コントラクト ---------- EPG 2

コントラクトで許可されていれば、EPG 1 のエンドポイントは EPG 2 のエンドポイントと通信でき、またその逆も可能です。このポリシーの構造には非常に柔軟性があります。たとえば、EPG 1 と EPG 2 間には多くのコントラクトが存在でき、1 つのコントラクトを使用する EPG が 3 つ以上存在でき、コントラクトは複数の EPG のセットで再利用できます。

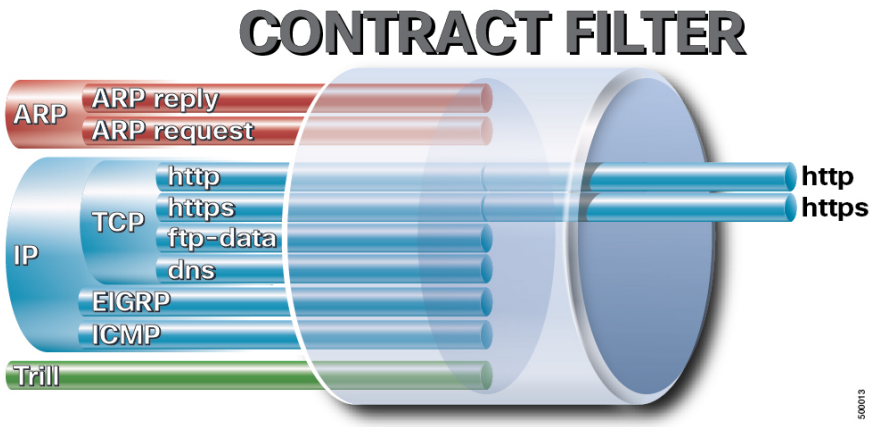
また EPG とコントラクトの関係には方向性があります。EPG はコントラクトを提供または消費できます。コントラクトを提供する EPG は通常、一連のクライアント デバイスにサービスを提供する一連のエンドポイントです。そのサービスによって使用されるプロトコルはコントラクトで定義されます。コントラクトを消費する EPG は通常、そのサービスのクライアントである一連のエンドポイントです。クライアント エンドポイント (コンシューマ) がサーバ エンドポイント (プロバイダー) に接続しようとすると、コントラクトはその接続が許可されるかどうかを確認します。特に指定のない限り、そのコントラクトは、サーバがクライアントへの接続を開始することを許可しません。ただし、EPG 間の別のコントラクトが、その方向の接続を簡単に許可する場合があります。

この提供/消費の関係は通常、EPG とコントラクト間を矢印を使って図で表されます。次に示す矢印の方向に注目してください。

EPG 1 <------- 消費 -------- コントラクト <------- 提供 -------- EPG 2

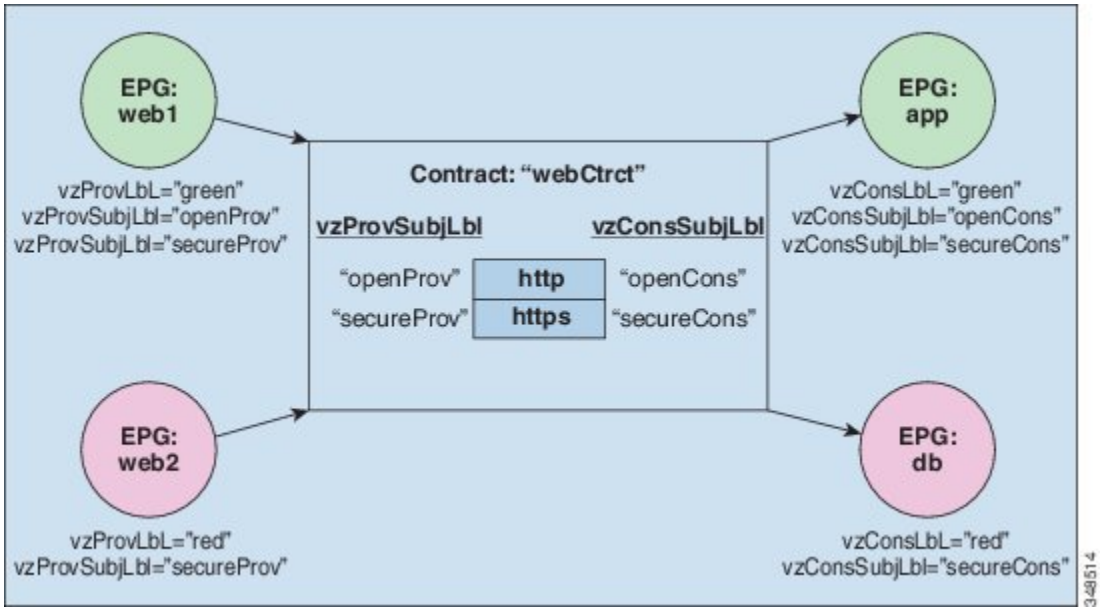
コントラクトは階層的に構築されます。1 つ以上のサブジェクトで構成され、各サブジェクトには 1 つ以上のフィルタが含まれ、各フィルタは 1 つ以上のプロトコルを定義できます。

図 *9:* コントラクト フィルタ



次の図は、コントラクトが EPG の通信をどのように管理するかを示します。

図 *10 : EPG/EPG* 通信を決定するコントラクト



たとえば、TCP ポート 80 とポート 8080 を指定する HTTP と呼ばれるフィルタと、TCP ポート 443 を指定する HTTPS と呼ばれる別のフィルタを定義できます。その後、2 セットのサブジェクトを持つ webCtrct と呼ばれるコントラクトを作成できます。openProv と openCons are は HTTP フィルタが含まれるサブジェクトです。secureProv と secureCons は HTTPS フィルタが含まれるサブジェクトです。この webCtrct コントラクトは、Web サービスを提供する EPG とそのサービスを消費するエンドポイントを含む EPG 間のセキュアな Web トラフィックと非セキュアな Web トラフィックの両方を可能にするために使用できます。

これらの同じ構造は、仮想マシンのハイパーバイザを管理するポリシーにも適用されます。EPG が Virtual Machine Manager（VMM）のドメイン内に配置されると、APIC は EPG に関連付けられたすべてのポリシーを VMM ドメインに接続するインターフェイスを持つリーフ スイッチにダウンロードします。VMM ドメインの完全な説明については、『*Application Centric Infrastructure Fundamentals*』の「*Virtual Machine Manager Domains*」の章を参照してください。このポリシーが作成されると、APIC は EPG のエンドポイントへの接続を可能にするスイッチを指定する VMM ドメインにそれをプッシュ（あらかじめ入力）します。VMM ドメインは、EPG 内のエンドポイントが接続できるスイッチとポートのセットを定義します。エンドポイントがオンラインになると、適切な EPG に関連付けられます。パケットが送信されると、送信元 EPG および宛先 EPG がパケットから取得され、対応するコントラクトで定義されたポリシーでパケットが許可されたかどうかが確認されます。許可された場合は、パケットが転送されます。許可されない場合は、パケットはドロップされます。

コントラクトは 1 つ以上のサブジェクトで構成されます。各サブジェクトには 1 つ以上のフィルタが含まれます。各フィルタには 1 つ以上のエントリが含まれます。各エントリは、アクセス コントロール リスト (ACL) の 1 行に相当し、エンドポイント グループ内のエンドポイントが接続されているリーフ スイッチで適用されます。

詳細には、コントラクトは次の項目で構成されます。

- 名前：テナントによって消費されるすべてのコントラクト (common テナントまたはテナント自体で作成されたコントラクトを含む) にそれぞれ異なる名前が必要です。
- サブジェクト：特定のアプリケーションまたはサービス用のフィルタのグループ。
- フィルタ：レイヤ 2 ～ レイヤ 4 の属性 (イーサネット タイプ、プロトコル タイプ、TCP フラグ、ポートなど) に基づいてトラフィックを分類するために使用します。
- アクション：フィルタリングされたトラフィックで実行されるアクション。次のアクションがサポートされます。
  - トラフィックの許可 (通常のコントラクトのみ)
  - トラフィックのマーク (DSCP/CoS) (通常のコントラクトのみ)
  - トラフィックのリダイレクト (サービス グラフによる通常のコントラクトのみ)
  - トラフィックのコピー (サービス グラフまたは SPAN による通常のコントラクトのみ)
  - トラフィックのブロック（禁止コントラクトのみ）

    Cisco APIC リリース 3.2(x) および名前が EX または FX で終わるスイッチでは、標準コントラクトで代わりに件名 [拒否] アクションまたは [コントラクトまたは件名の除外] を使用して、指定のパターンを持つトラフィックをブロックできます。
  - トラフィックのログ（禁止コントラクトと通常のコントラクト）
- エイリアス：(任意) 変更可能なオブジェクト名。オブジェクト名は作成後に変更できませんが、エイリアスは変更できるプロパティです。

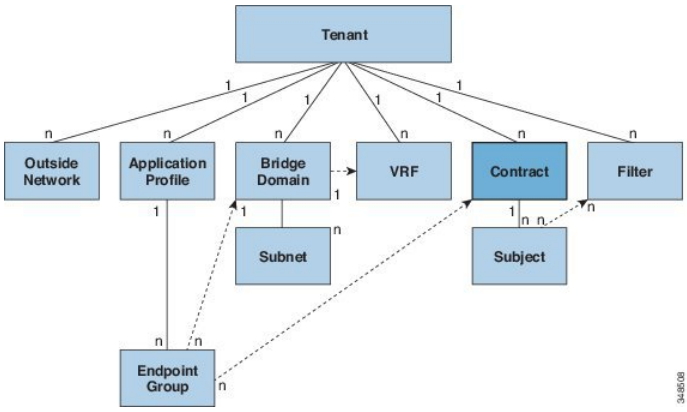
このように、コントラクトによって許可や拒否よりも複雑なアクションが可能になります。コントラクトは、所定のサブジェクトに一致するトラフィックをサービスにリダイレクトした

り、コピーしたり、その QoS レベルを変更したりできることを指定可能です。具象モデルでアクセス ポリシーをあらかじめ入力すると、APIC がオフラインまたはアクセスできない場合でも、エンドポイントは移動でき、新しいエンドポイントをオンラインにでき、通信を行うことができます。APIC は、ネットワークの単一の障害発生時点から除外されます。ACI ファブリックにパケットが入力されると同時に、セキュリティ ポリシーがスイッチで実行している具象モデルによって適用されます。

コントラクト

EPG に加えて、コントラクト（vzBrCP）はポリシー モデルの主要オブジェクトです。EPG は唯一、コントラクトのルールに従って他の EPG と通信できます。次の図は、管理情報ツリー（MIT）内のコントラクトの場所とテナントの他のオブジェクトとの関係を示します。

図 *11* : コントラクト



管理者は、コントラクトを使用して許可されるプロトコルやポートを含む EPG 間を通過できるトラフィックのタイプを選択します。コントラクトがなければ、EPG間通信はデフォルトでディセーブルになります。EPG 内の通信に必要なコントラクトはありません。EPG 内の通信は常に暗黙的に許可されています。

VRF で Epg 間の通信の高い制御を有効にする優先契約グループを設定することもできます。VRF で Epg のほとんどは、オープンな通信である必要がありますが、いくつかの他の Epg との通信が制限のみする必要がありますと、は、通信を正確に制御するフィルタと契約優先グループと契約の組み合わせを設定できます。

コントラクトは、次のタイプのエンドポイント グループの通信を管理します。

- ACI ファブリック アプリケーション EPG （fvAEPg）間、テナント内およびテナント間の両方

（注） 共有サービス モードの場合、コントラクトはテナント間通信に必要です。A contract is used to specify static routes across VRFs, even though the tenant VRF does not enforce a policy.

- ACI ファブリック アプリケーション EPG とレイヤ 2 外部外側ネットワークのインスタンス EPG （l2extInstP）間
- ACI ファブリック アプリケーション EPG とレイヤ 3 外部外側ネットワークのインスタンス EPG （l3extInstP）間
- ACI ファブリック アウトオブバンド（mgmtOoB）またはインバンド（mgmtInB）管理 EPG 間

コントラクトは、プロバイダー、コンシューマ、またはその両方とラベル付された EPG 間の通信を制御します。EPG プロバイダーは、コンシューマ EPG が従う必要のあるコントラクトを公開します。EPG とコントラクトの関係は、プロバイダーまたはコンシューマです。EPG がコントラクトを提供すると、通信が提供されたコントラクトに準拠している限り、その EPG との通信は他の EPG から開始できます。EPG がコントラクトを使用すると、その EPG のエンドポイントは、コントラクトを指定した EPG のエンドポイントと通信を開始できます。

（注） 1 つの EPG で同じコントラクトを指定および使用できます。EPG は複数のコントラクトを同時に提供および消費することもできます。

REST API を使用したコントラクトの設定

手順

次の例のように、XML POST 要求を使用してコントラクトを設定します。

例：

```
<vzBrCP name="webCtrct">
          <vzSubj name="http" revFltPorts="true" provmatchT="All">
              <vzRsSubjFiltAtt tnVzFilterName="Http"/>
              <vzRsSubjGraphAtt graphName="G1" termNodeName="TProv"/>
              <vzProvSubjLbl name="openProv"/>
              <vzConsSubjLbl name="openCons"/>
          </vzSubj>
          <vzSubj name="https" revFltPorts="true" provmatchT="All">
              <vzProvSubjLbl name="secureProv"/>
              <vzConsSubjLbl name="secureCons"/>
              < vzRsSubjFiltAtt tnVzFilterName="Https"/>
              <vzRsOutTermGraphAtt graphName="G2" termNodeName="TProv"/>
          </vzSubj>
      </vzBrCP>
```

REST API を使用した禁止コントラクトの設定

始める前に

次のオブジェクトを作成する必要があります。

- これで関連付けられるテナント **Taboo 契約**
- テナントのアプリケーション プロファイル
- テナントの最低 1 個の EPG

手順

REST API を使用してタブー契約を作成するには、次の例ではよう XML を使用します。

例：

```
<vzTaboo ownerTag="" ownerKey="" name="VRF64_Taboo_Contract"
dn="uni/tn-Tenant64/taboo-VRF64_Taboo_Contract" descr=""><vzTSubj
name="EPG_subject" descr=""><vzRsDenyRule tnVzFilterName="default"
directives="log"/>
</vzTSubj>
</vzTaboo>
```

契約とサブジェクトの例外

コントラクトまたはコントラクトの件名の例外の設定

Cisco APIC リリース 3.2(1) では、EPG 間のコントラクトが拡張され、コントラクトに参加しているコントラクトプロバイダまたはコンシューマのサブネットを拒否できます。インター EPG コントラクトおよびい内部 EPG コントラクトは、この機能でサポートされます。

プロバイダ EPG の件名を有効にして、件名またがコントラクトの例外で一致基準が設定されているものを除くすべてのコンシューマ EPG との通信が可能になります。たとえば、サブセットを除く、テナントのすべての EPG にサービスを提供するために EPG を有効にする場合、これら EPG を除外できます。これを設定するには、コントラクトまたはそのコントラクトの件名のいずれかで例外を作成します。サブセットがコントラクトの提供または消費のアクセスを拒否します。

ラベル、カウンタ、許可および拒否ログは、コントラクトおよび件名の例外でサポートされています。

コントラクトのすべての件名に例外を適用するには、コントラクトに例外を追加します。コントラクトの単一の件名にのみ例外を適用する場合、件名に例外を追加します。

件名にフィルタを追加する場合、フィルタのアクションを設定できます（フィルタ条件に一致するオブジェクトを許可または拒否する）。また、**[拒否]** フィルタについては、フィルタの優

先順位を設定することができます。**[許可]**フィルタは常にデフォルトの優先順位があります。自動拒否の件名-フィルタ関係をマーキングすると、件名に一致している場合、各 EPG のペアに適用されます。コントラクトと件名には、複数の件名-フィルタ関係を含むことができます。これは、フィルタに一致するオブジェクトを許可または拒否するように独自に設定できます。

例外タイプ

コントラクトと件名の例外は次のタイプに基づき、* ワイルドカードなどの正規表現を含むことができます。

例外の条件は、[コンシューマ正規表現] および [プロバイダ正規表現]のフィールドで定義されているように、これらのオブジェクトを除外します。	例	説明
テナント	<vzException consRegex= “common” field= “Tenant” name= “excep03” provRegex= “t1” />	この例では、common テナントを使用して、 EPG が t1 テナントにより提供されるコントラクトを消費しないように除外します。
VRF	<vzException consRegex= “ctx1” field= “Ctx” name= “excep05” provRegex= “ctx1” />	この例では、ctx1 のメンバーが同じ VRF から提供されるサービスを使用しないように除外します。
EPG	<vzException consRegex= “EPgPa*” field= “EPg” name= “excep03” provRegex= “EPg03” />	この例では、名前が EPGPa から始まる複数の EPG が存在すると仮定し、EPg03 により提供されているコントラクトのコンシューマとしてすべて拒否される必要があります。
Dn	<vzException consRegex= “uni/tn-t36/ap-customer/epg-epg193” field= “Dn” name=“excep04” provRegex= “uni/tn-t36/ap-customer/epg-epg200” />	この例では、epg193 が epg200 により提供されたコントラクトを消費しないように除外します。
タグ	<vzException consRegex= “red” field= “Tag” name= “excep01” provRegex= “green” />	例では、red タグでマークされているオブジェクトが消費することと、green タグでマークされているオブジェクトがコントラクトに参加しないように除外します。

REST API を使用した契約またはサブジェクトの例外の設定

このタスクでは、ほとんどの EPG の通信を許可するコントラクトを設定しますが、それらのサブネットへのアクセスを拒否します。契約またはサブジェクトには、複数の例外を追加することができます。

始める前に

テナント、VRF、アプリケーション プロファイル、EPG を設定して、コントラクトを提供し消費します。

手順

ステップ 1 次の例のような XML を POST 送信することによりフィルタを作成します:

例：

```
<vzFilter name='http-filter'>
    <vzEntry name='http-e' etherT='ip' prot='tcp'/>
    <vzEntry name='https-e' etherT='ip' prot='tcp'/>
</vzFilter>
```

ステップ 2 次の例のような XML を POST 送信することにより、サブジェクトを利用できないように `EPg01` を例外とし、提供できないように `EPg03` を例外とします:

vzException MO は、vzBrCP または vzSubj MO に含めることができます。

例：

```
<vzBrCP name="httpCtrct" scope="context">
    <vzSubj name="subj1"
        <vzException consRegex="EPg01" field="EPg" name="excep01" provRegex=EPg03"/>
    <vzSubj/>
         <vzRsSubjFiltAtt tnVzFilterName="http-filter" Action="deny"/>
         <vzRsSubjFiltAtt tnVzFilterName="https-filter" Action="permit"/>
     </vzSubj>
</vzBrCP>
```

REST API を使用した EPG のコントラクト継承の設定

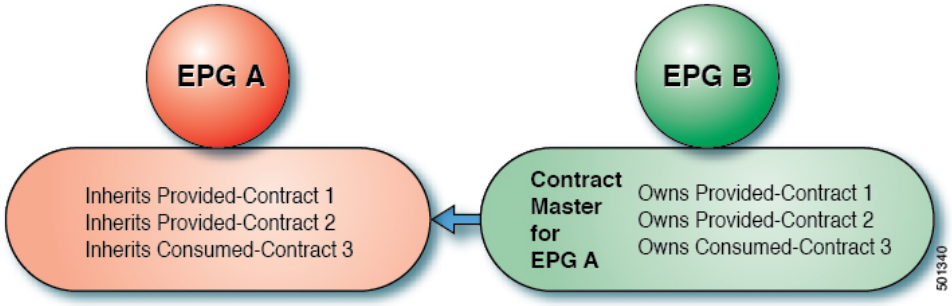
コントラクト継承について

関連する契約を新しい EPG に統合するため、EPG を有効にして同じテナントの別の EPG に直接関連する契約すべて（提供済み/消費済み）を継承できます。コントラクトの継承は、アプリケーション EPG、マイクロセグメント EPG、L2Out EPG、および L3Out EPG に設定できます。

リリース 3.x では、EPG 間の提供済み/消費済みの両方の契約に、契約を継承する設定も可能です。EPG 間契約が、モデル名や後発のモデルの最後に EX または EX が付く、Cisco Nexus 9000 シリーズ スイッチでサポートされています。

EPG を有効にし、APIC GUI、NX-OS スタイル CLI、REST API を使用して、別の EPG に直接関連する契約すべてを継承できます。

図 *12*：コントラクトの継承



上の図で、EPG A は EPG B から（EPG A の契約マスター）提供済みの契約 1 および 2、消費済みの契約 3 を継承するように設定されています。

コントラクト継承を設定する際は、次のガイドラインに従ってください。

- コントラクト継承は、アプリケーション EPG、マイクロセグメント（uSeg）EPG、外部 L2Out EPG、および外部 L3Out EPG 用に設定できます。コントラクト関係は同じタイプの EPG 間で確立する必要があります。
- 関係が確立されると、提供するコントラクトと消費するコントラクトの両方がコントラクト マスターから継承されます。
- コントラクト マスターとコントラクトを継承する EPG は同じテナント内にある必要があります。
- マスター契約への変更は、すべての継承に伝播されます。新しい契約がマスターに追加される場合、継承先にも追加されます。
- EPG は、複数のコントラクト マスターからコントラクトを継承することができます。
- コントラクト継承は単一のレベルでのみサポートされ（連結できない）、コントラクトマスターがコントラクトを継承することはできません。
- コントラクト サブジェクト ラベルおよび EPG ラベルの継承がサポートされています。EPG A が EPG B から契約を継承する場合、EPG A と EPG B で異なるサブジェクト ラベルが設定されていると、APIC は EPG B で設定されているサブジェクト ラベルのみを使用し、どちらの EPG からもラベルを収集しません。
- EPG が契約に直接関連付けられている、または契約を継承しているかどうかに関わらず、TCAM 内のエントリが消費されます。したがって契約スケール ガイドラインが引き続き適用されます。詳細については、お使いのリリースの「検証されたスケーラビリティ ガイド」を参照してください。
- vzAny セキュリティ コントラクトとタブー コントラクはサポートされません。

契約の継承設定および継承済みおよびスタンドアロン契約を表示することに関する詳細は、「*Cisco APIC* の基本設定ガイドを参照してください。

REST API を使用したアプリケーション EPG のコントラクト継承の設定

始める前に

EPG が使用するテナントとアプリケーション プロファイルを設定します。

EPG コントラクト マスターとして機能するようにアプリケーション EPG を設定します。

共有するコントラクトを設定し、コントラクト マスターに関連付けます。

手順

REST API を使用してコントラクト継承を設定するには、コントラクトを継承する EPG に転送される URL を指定して、次の XML および JSON の例のような XML を POST 送信します。

例：

XML の例

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- /api/node/mo/uni/tn-coke/ap-AP/epg-EPg_B.xml -->
<polUni>


  <fvEPg>

    <fvRsSecInherited tDn="uni/tn-coke/ap-AP/epg-EPg_B/>
  </fvEPg>
</polUni>
```

JSON の例

```
https://192.168.200.10/api/node/mo/uni/tn-coke/ap-AP/epg-EPg_B.json
fvAEPg":{"attributes":{
   "dn":"uni/tn-coke/ap-AP/epg-EPg_B","name":"EPg_C",
   "rn":"epg-EPg_C",
   "status":"created"},
   "children":[{
     "fvRsBd":{"attributes":{
        "tnFvBDName":"default",
        "status":"created,modified"},
        "children":[]}},{
     "fvRsSecInherited":{"attributes":{
        "tDn":"uni/tn-coke/ap-AP/epg-EPg_B",
        "status":"created"},
        "children":[]}}]}}
```

REST API を使用した uSeg EPG のコントラクト継承の設定

始める前に

EPG が使用するテナントとアプリケーション プロファイルを設定します。

EPG コントラクト マスターとして機能するようにアプリケーション EPG を設定します。

共有するコントラクトを設定し、コントラクト マスターに関連付けます。

手順

REST API を使用して uSeg のコントラクト継承を設定するには、次の例のような XML を POST 送信します。

例：

```
<polUni>
    <fvTenant name="Tn1" >
        <fvAEPg descr="" dn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_120" fwdCtrl=""
isAttrBasedEPg="yes" matchT="AtleastOne" name="uSeg1_301_120" pcEnfPref="unenforced"
prefGrMemb="exclude" prio="unspecified">
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_100" />
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_110" />
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_50" />
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_60" />
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_30" />
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_10" />
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_40" />
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_70" />
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_90" />
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_20" />
            <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_80" />
           <fvRsNodeAtt descr="" encap="unknown" instrImedcy="immediate" mode="regular"
 tDn="topology/pod-1/node-108" />
           <fvRsNodeAtt descr="" encap="unknown" instrImedcy="immediate" mode="regular"
 tDn="topology/pod-1/node-109" />
           <fvRsDomAtt classPref="encap" delimiter="" encap="vlan-301" encapMode="auto"
 instrImedcy="immediate" netflowPref="disabled" primaryEncap="unknown"
resImedcy="immediate" tDn="uni/phys-PhysDom1" />
            <fvRsCustQosPol tnQosCustomPolName="" />
            <fvRsBd tnFvBDName="T1BD21" />
            <fvCrtrn descr="" match="any" name="default" nameAlias="" ownerKey=""
ownerTag="" prec="0">
                <fvIpAttr descr="" ip="192.14.1.120" name="0" nameAlias="" ownerKey=""
ownerTag="" usefvSubnet="no" />
            </fvCrtrn>
        </fvAEPg>
    </fvTenant>
</polUni>
```

次のタスク

REST API を使用した L2Out EPG のコントラクト継承の設定

始める前に

EPG が使用するテナントとアプリケーション プロファイルを設定します。

L2Out コントラクト マスターとして機能するように L2Out EPG を設定します。

共有するコントラクトを設定し、コントラクト マスターに関連付けます。

手順

REST API を使用して L2Out EPG のコントラクト継承を設定するには、次の例のような XML を POST 送信します。

例：

```
<polUni>
  <fvTenant name="Tn1" >
     <l2extOut name="l2out1">
        <l2extRsEBd encap="vlan-51" tnFvBDName="T1BD1" />
        <l2extRsL2DomAtt tDn="uni/l2dom-l2Dom1" />
        <l2extLNodeP name="default" >
           <l2extLIfP name="default" >
             <l2extRsPathL2OutAtt tDn="topology/pod-1/protpaths-108-109/pathep-[VPC83]"
 />
           </l2extLIfP>
        </l2extLNodeP>
        <l2extInstP matchT="AtleastOne" name="l2Ext1">
           <fvSubnet ctrl="nd" ip="192.13.1.10/24" preferred="no" scope="public,shared"
 virtual="no" />
           <fvRsProv tnVzBrCPName="T1ctr_tcp" />
        </l2extInstP>
     </l2extOut>

     <l2extOut name="l2out2">
        <l2extRsEBd encap="vlan-53" tnFvBDName="T1BD3" />
        <l2extRsL2DomAtt tDn="uni/l2dom-l2Dom1" />
        <l2extLNodeP name="default" >
           <l2extLIfP name="default" >
             <l2extRsPathL2OutAtt tDn="topology/pod-1/protpaths-108-109/pathep-[VPC84]"
 />
           </l2extLIfP>
        </l2extLNodeP>
        <l2extInstP matchT="AtleastOne" name="l2Ext3" prefGrMemb="exclude">
           <fvSubnet ctrl="nd" ip="192.13.2.10/24" preferred="no" scope="public,shared"
 virtual="no" />
           <fvRsSecInherited tDn="uni/tn-Tn1/l2out-l2out1/instP-l2Ext1" />
        </l2extInstP>
     </l2extOut>

  </fvTenant>
</polUni>
```

REST API を使用した L3Out EPG のコントラクト継承の設定

始める前に

EPG が使用するテナントとアプリケーション プロファイルを設定します。

L3Out コントラクト マスターとして機能するように L3Out EPG を設定します。

共有するコントラクトを設定し、コントラクト マスターに関連付けます。

手順

REST API を使用して L3Out EPG のコントラクト継承を設定するには、次の例のような XML を POST 送信します。

例：

```
<polUni>
  <fvTenant name="Tn6" >

  <!— L3out creation -->
    <ospfIfPol deadIntvl="40" helloIntvl="10" name="ospf1" pfxSuppress="inherit" prio="1"
 rexmitIntvl="5" xmitDelay="1" />
     <l3extOut enforceRtctrl="export" name="T6L3out821">
        <ospfExtP areaCost="1" areaCtrl="redistribute,summary" areaId="0.0.0.1"
areaType="regular" />
        <l3extRsL3DomAtt tDn="uni/l3dom-L3Dom1" />
        <l3extRsEctx tnFvCtxName="T6ctx21" />
        <l3extLNodeP name="l3out_vpc82_prof" >
           <l3extRsNodeL3OutAtt rtrId="1.1.1.8" rtrIdLoopBack="yes"
tDn="topology/pod-1/node-108">
              <l3extInfraNodeP fabricExtCtrlPeering="no" />
           </l3extRsNodeL3OutAtt>
           <l3extRsNodeL3OutAtt rtrId="1.1.1.9" rtrIdLoopBack="yes"
tDn="topology/pod-1/node-109">
              <l3extInfraNodeP fabricExtCtrlPeering="no" />
           </l3extRsNodeL3OutAtt>
           <l3extLIfP name="ospf1" >
              <ospfIfP authKeyId="1" authType="none" >
                 <ospfRsIfPol tnOspfIfPolName="ospf1" />
              </ospfIfP>
              <l3extRsPathL3OutAtt encap="vlan-551" ifInstT="ext-svi" mode="regular"
mtu="1500" tDn="topology/pod-1/protpaths-108-109/pathep-[VPC82]" >
                 <l3extMember addr="192.16.51.1/24" llAddr="0.0.0.0" side="B" />
                 <l3extMember addr="192.16.51.2/24" llAddr="0.0.0.0" side="A" />
              </l3extRsPathL3OutAtt>
              <l3extRsNdIfPol tnNdIfPolName="" />
           </l3extLIfP>
        </l3extLNodeP>

        <l3extInstP matchT="AtleastOne" name="T6l3Ext821">
              <fvRsProv tnVzBrCPName="T6ctr_UDP_TCP2" />
              <fvRsCons tnVzBrCPName="T6ctr_UDP_TCP1" />
           <l3extSubnet ip="192.16.51.0/24"
scope="import-security,shared-rtctrl,shared-security" />
              <l3extSubnet ip="192.16.61.0/24"
scope="import-security,shared-rtctrl,shared-security" />
              <vzConsSubjLbl name="tcp" tag="green" />
           <vzProvSubjLbl name="tcp" tag="green" />
        </l3extInstP>
```

```
        <l3extInstP matchT="AtleastOne" name="T6l3Ext823">
           <fvRsSecInherited tDn="uni/tn-Tn6/out-T6L3out821/instP-T6l3Ext821" />
              <l3extSubnet ip="192.16.63.0/24"
scope="import-security,shared-rtctrl,shared-security" />
        </l3extInstP>
     </l3extOut>

  </fvTenant>
</polUni>
```

優先グループ契約

契約優先グループについて

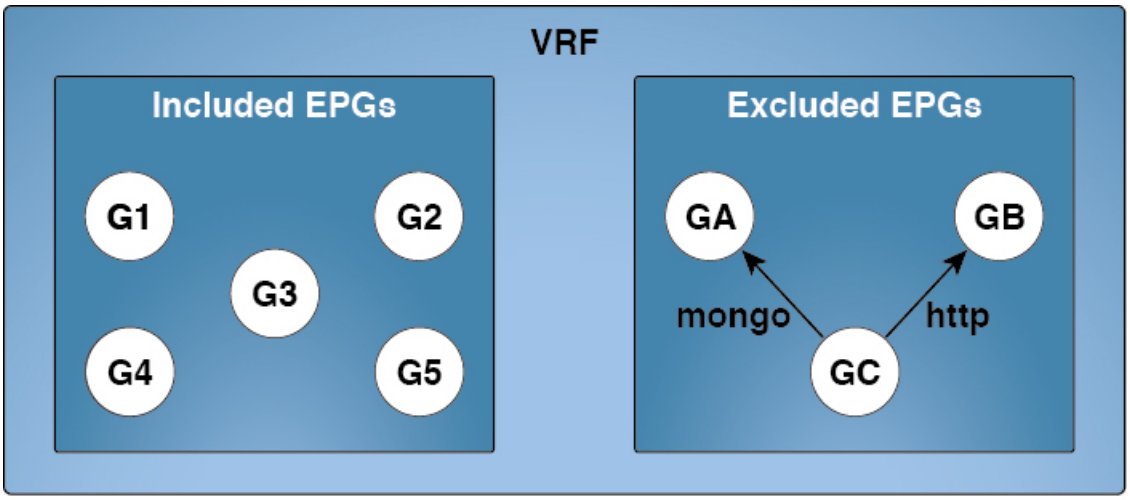
契約優先グループが設定されている VRF で、EPG に利用可能なポリシー適用には 2 種類あります。

- EPG を含む：EPG が契約優先グループのメンバーシップを持っている場合、EPG は契約をせずにお互いに自由に通信できます。これは、source-any-destination-any-permit デフォルト ルールに基づくものです。
- EPG を除外：優先グループのメンバーではない EPG は、相互に通信するために契約が必要です。そうしない場合、デフォルトの source-any-destination-any-deny ルールが適用されます。

契約優先グループ機能では、VRF で EPG 間のより高度な通信の制御が可能です。VRF の EPG のほとんどはオープン通信ですが、一部には他の EPG との制限がある場合、契約優先グループとフィルタ付きの契約の組み合わせを設定し、EPG 内の通信を正確に制御できます。

優先グループから除外されている EPG は、source-any-destination-any-deny デフォルト ルールを上書きする契約がある場合にのみ、他 EPG と通信できます。

図 *13* : 契約優先グループの概要



Source	Destination	Filter	Action
any	any	implicit	permit
GA	any	implicit	deny
any	GA	implicit	deny
GB	any	implicit	deny
any	GB	implicit	deny
GC	any	implicit	deny
any	GC	implicit	deny

Source	Destination	Filter	Action
GC	GA	mongo	permit
GA	GC	mongo	permit
GC	GB	http	permit
GB	GC	http	permit

制限事項

以下の制限が契約優先グループに適用されます。

- L3Out およびアプリケーション EPG が契約優先グループで設定されており、EPG が VPC でのみ展開されているトポロジで、VPC の 1 つのリーフ スイッチのみに L3Out のプレフィックス エントリがあることがわかります。この場合、VPC の他のリーフ スイッチにはエントリがなく、そのためトラフィックをドロップします。

  この問題を回避するには、次のいずれかを行います。

  - VRF の契約グループを無効および再度有効にします。
  - L3Out EPG のプレフィックス エントリを削除し再度作成します。

- また、サービス グラフ契約のプロバイダまたはコンシューマ EPG が契約グループに含まれる場合、シャドウ EPG は契約グループから除外できません。シャドウ EPG は契約グループで許可されますが、シャドウ EPG が展開されているノードで契約グループ ポリシーの展開をトリガしません。ノードに契約グループ ポリシーをダウンロードするには、契約グループ内にダミー EPG を展開します。

REST API を使用した契約優先グループの設定

次の例は、契約優先グループを作成 `vrf64` 、し、VRF で次の 3 つの Epg を作成します。

- `epg ldap` ：優先グループに含まれています
- メール ：優先グループに含まれています
- `radius` ：優先グループから除外

始める前に

VRF で、テナント、Vrf、および、Epg を作成します。

手順

XML の例などと、post を送信することにより契約優先グループを作成します。

例：

```
<polUni>
  <fvTenant name="tenant64">
    <fvCtx name="vrf64"> <vzAny prefGrMemb="enabled"/> </fvCtx>
    <fvBD name="bd64"> <fvRsCtx tnFvCtxName="vrf64"/> </fvBD>
    <fvAp name="app-lldp">
      <fvAEPg name="epg-ldap" prefGrMemb="include">
        <fvRsBd tnFvBDName="bd64"/>
        <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/3]" encap="vlan-113"
instrImedcy="immediate"/>
      </fvAEPg>
      <fvAEPg name="mail" prefGrMemb="include">
        <fvRsBd tnFvBDName="bd64"/>
        <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/4]" encap="vlan-114"
instrImedcy="immediate"/>
      </fvAEPg>
      <fvAEPg name="radius" prefGrMemb="exclude">
        <fvRsBd tnFvBDName="bd64"/>
        <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/5]" encap="vlan-115"
instrImedcy="immediate"/>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

次のタスク

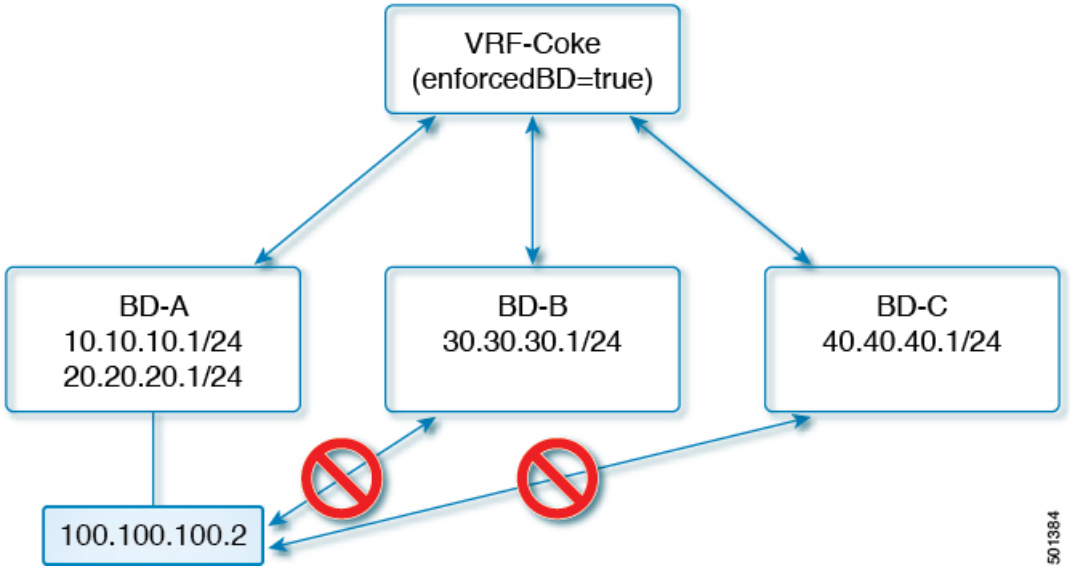
通信を制御するには、契約の作成、 `radius` 他 Epg で EPG。

適用されるブリッジ ドメインの設定

適用されるブリッジ ドメイン（BD）設定では、関連付けられているブリッジ ドメイン内のサブネット ゲートウェイを ping できるサブジェクト エンドポイント グループ（EPG）で、エンドポイントを作成する必要があります。

この設定では、任意のサブネット ゲートウェイに ping を送信できる IP アドレスのグローバル例外リストを作成できます。

図 *14* : 適用されるブリッジ ドメイン



（注）

- 例外の IP アドレスは、すべての VRF 上ですべての BD ゲートウェイの ping を実行できます。
- L3 アウトに設定されているループバック インターフェイスは、対象とするループバック インターフェイスに設定されている IP アドレスへの到達可能性を強制しません。
- eBGP ピア IP アドレスが、L3out インターフェイスのサブネットとは異なるサブネットに存在している場合、許容される例外サブネットにピア サブネットを追加する必要があります。

  そうしないと、送信元 IP アドレスが L3out インターフェイスのサブネットとは異なるサブネットに存在するため、eBGP トラフィックがブロックされます。

REST API を使用した、適用されるブリッジ ドメインの設定

手順

	コマンドまたはアクション	目的

ステップ 1	テナントを作成します。
	例：
	`POST https://apic-ip-address/api/mo/uni.xml`
	`<fvTenant name="ExampleCorp"/>`
		POSTが成功すると、作成したオブジェクトが出力に表示されます。

ステップ 2	VRF およびブリッジ ドメインを作成します。
	例：
	`URL for POST: https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml`
	`<fvTenant name="ExampleCorp">`
	`<fvCtx name="pvn1"/>`
	`<fvBD name="bd1">`
	`<fvRsCtx tnFvCtxName="pvn1" bdEnforceEnable="yes"/>`
	`<fvSubnet ip="10.10.100.1/24"/>`
	`</fvBD>`
	`</fvTenant>`
	例外 IP を追加するには、次の POST 送信を使用します:
	https://apic-ip-address/api/node/mo/uni/infra.xml
	`<bdEnforceExceptionCont>`
	`<bdEnforceExceptIp ip="11.0.1.0/24"/>`
	`</bdEnforceExceptionCont>`
	（注） 外部ルーテッドを設定するときにパブリック サブネットがある場合は、ブリッジ ドメインを外部設定と関連付ける必要があります。
		（注） ゲートウェイ アドレスは、IPv4 または IPv6 アドレスにすることができます。IPv6 ゲートウェイ アドレスの詳細については、関連する KB 記事、「*KB: Creating a Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery*」を参照してください。