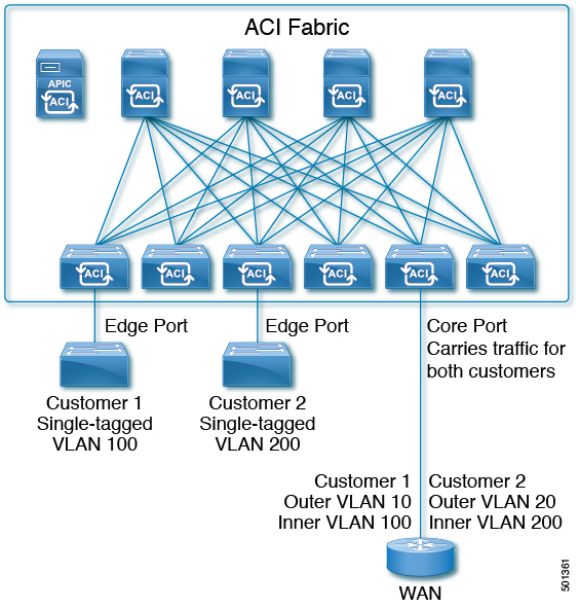
802.1 q トンネリング

この章は、次の内容で構成されています。

- ACI 802.1 q トンネルについて （1 ページ）
- GUI を使用した802.1Q トンネルの設定 （3 ページ）
- NX-OS スタイルの CLI を使用した 802.1Q トンネルの設定 （5 ページ）
- REST API を使用した 802.1Q トンネルの設定 （10 ページ）

ACI 802.1 q トンネルについて

図 *1 : ACI 802.1 q* トンネル



エッジ（トンネル）ポートで 802.1Q トンネルを設定して、Quality of Service（QoS）の優先順位設定とともに、ファブリックのイーサネット フレームの point-to-multi-point トンネリングを有効にできます。Dot1q トンネルは、タグなし、802.1Q タグ付き、802.1ad 二重タグ付きフレームを、ファブリックでそのまま送信します。各トンネルでは、単一の顧客からのトラフィック

を伝送し、単一のブリッジ ドメインに関連付けられています。Cisco Application Centric Infrastructure（ACI）の前面パネル ポートは、Dot1q トンネル の一部とすることができます。レイヤ 2 スイッチングは宛先 MAC（DMAC）に基づいて行われ、通常の MAC ラーニングはトンネルで行われます。エッジ ポート Dot1q トンネルは、スイッチ モデル名の最後に「EX」またはそれ以降のサフィックスが付く、Cisco Nexus 9000 シリーズ スイッチでサポートされます。

同じコア ポートで複数の 802.1Q トンネルを設定することができ、複数の顧客からの二重タグ付きトラフィックを伝送できます。それぞれは、802.1Q トンネルごとに設定されたアクセスのカプセル化で識別されます。802.1Q トンネルでは、MAC アドレス学習を無効にすることもできます。エッジ ポートとコア ポートの両方を、アクセス カプセル化が設定され、MAC アドレス学習が無効にされた 802.1Q トンネルに所属させることができます。エッジ ポートとコア ポートの Dot1q トンネル は、スイッチ モデル名の最後に「FX」またはそれ以降のサフィックスが付く、Cisco Nexus 9000 シリーズ スイッチでサポートされます。

IGMP および MLD パケットは、802.1Q トンネルを介して転送できます。

このドキュメントで使用する用語は、**Cisco Nexus 9000 シリーズ** のドキュメントとは異なっている場合があります。

表 *1 : 802.1Q* トンネルの用語

ACI のドキュメント	Cisco Nexus 9000 シリーズのドキュメント
エッジ ポート	トンネル ポート
コア ポート	トランク ポート

次の注意事項および制約事項が適用されます:

- VTP、CDP、LACP、LLDP、および STP プロトコルのレイヤ 2 トンネリングは、次の制限付きでサポートされます。
  - リンク集約制御プロトコル (LACP) トンネリングは、個々のリーフ インターフェイスを使用する、ポイントツーポイント トンネルでのみ、予想通りに機能します。ポート チャネル（PC）または仮想ポートチャネル（vPC）ではサポートされていません。
  - PC または vPC を持つ CDP および LLDP トンネリングは確定的ではありません。これは、トラフィックの宛先として選択するリンクによって異なります。
  - レイヤ 2 プロトコル トンネリングに VTP を使用するには、CDP をトンネル上で有効にする必要があります。
  - レイヤ 2 プロトコルのトンネリングが有効になっており、Dot1q トンネルのコア ポートにブリッジ ドメインが展開されている場合、STP は 802.1Q トンネル ブリッジ ドメインではサポートされません。
  - Cisco ACI リーフ スイッチは、トンネル ブリッジ ドメインのエンドポイントでフラッシングを行い、ブリッジ ドメインでフラッディングすることにより、STP TCN パケットに反応します。

- 2 個上のインターフェイスを持つ CDP および LLDP トンネリングが、すべてのインターフェイスでパケットをフラッディングします。
- エッジ ポートからコア ポートにトンネリングしているレイヤ 2 プロトコル パケットの宛先 MAC アドレスは、01-00-0c-cd-cd-d0 に書き換えられ、コア ポートからエッジ ポートにトンネリングしているレイヤ 2 プロトコル パケットの宛先 MAC アドレスは、プロトコルに対して標準のデフォルト MAC アドレスに書き換えられます。

- PC または vPC が Dot1q Tunnel 内の唯一のインターフェイスであり、削除してから再設定した場合には、PC/VPC の Dot1q トンネル への関連付けを削除して、再設定してください。
- 製品 ID に EX が含まれるスイッチに導入された 802.1Q トンネルでは、最初の 2 つの VLAN タグの 0x8100 + 0x8100、0x8100 + 0x88a8、0x88a8 + 0x88a8のEthertype の組み合わせはサポートされません。

  トンネルが EX と FX またはそれ以降のスイッチの組み合わせに導入されている場合は、この制限が適用されます。

  製品 ID に FX 以降が含まれるスイッチにのみトンネルが導入されている場合、この制限は適用されません。
- コア ポートについては、二重タグつきフレームのイーサタイプは、0x8100 の後に 0x8100 が続く必要があります。
- 複数のエッジ ポートおよびコア ポートを（リーフ スイッチ上のものであっても）Dot1q トンネル に含めることができます。
- エッジ ポートは 1 つのトンネルの一部にのみ属することが可能ですが、コア ポートは複数の Dot1q トンネルに属することができます。
- 通常の EPG を 802.1Q で使用されるコア ポートに展開できます。
- L3Outs は、Dot1q トンネルで有効になっているインターフェイスではサポートされていません。
- FEX インターフェイスは Dot1q トンネル のメンバーとしてはサポートされていません。
- インターフェイス レベルの統計情報は Dot1q トンネル のインターフェイスでサポートされていますが、トンネル レベルの統計情報はサポートされていません。

GUI を使用した802.1Q トンネルの設定

APIC GUI を使用した 802. 1 Q トンネル インターフェイスの設定

次の手順で、トンネルを使用するインターフェイスを設定します:

始める前に

トンネルを使用するテナントを作成します。

ステップ 1 メニュー バーで、**[Fabric]** > **[Access Policies]** の順にクリックします。

ステップ 2 [ナビゲーション] バーで、**[ポリシー]** > **[インターフェイス]** > **[L2 インターフェイス]** をクリックします。

ステップ 3 **[L2 インターフェイス]** を右クリックし、**[L2 インターフェイス ポリシーの作成]** を選択して、次の操作を実行します。

a) **Name** フィールドに、レイヤ 2 インターフェイス ポリシーの名前を入力します。

b) オプション。ポリシーの説明を追加します。L2インターフェイスポリシーの目的を説明することをお勧めします。

c) **Dot1q トンネル**で、エッジ ポートとして使用するインターフェイスを有効にするインターフェイス ポリシーを作成するために、**QinQ** フィールドで、**edgePort** をクリックします。

d) **Dot1q トンネル**でコア ポートとして使用するインターフェイスを有効にするインターフェイス ポリシーを作成するために、**QinQ** フィールドで、**corePort** をクリックします。

ステップ 4 次の手順で、L2 インターフェイス ポリシーをポリシー グループに適用します。

a) **[ファブリック]** > **[アクセス ポリシー]** > **[インターフェイス]** > **[リーフ インターフェイス]**をクリックして、**[ポリシー グループ]** を展開します。

b) **[リーフ アクセス ポート]**、**[PC インターフェイス]** または **[VPC インターフェイス]** を右クリックし、トンネルに設定しているインターフェイスのタイプに応じて、次のいずれかを選択します。

- **リーフ アクセス ポート ポリシー グループの作成**
- **PC ポリシー グループの作成**
- **VPC ポリシー グループの作成**

c) 表示されるダイアログボックスで、以下のアクションを実行します:

- **Name** フィールドに、ポリシー グループの名前を入力します。

  オプション。ポリシー グループについての説明を追加します。ポリシー グループの目的を説明することをお勧めします。

- **L2 Interface Policy** フィールドで、下向き矢印をクリックし、前に作成した L2 インターフェイス ポリシーを選択します。

- CDP レイヤ 2 トンネリング プロトコルでトンネルを作成する場合は、[CDP Policy] 下向き矢印をクリックし、ポリシー ダイアログボックスでポリシーの名前を追加し、管理状態を無効にして、[Submit] をクリックします。.

- LLDP レイヤ 2 トンネリング プロトコルでトンネルを作成する場合には、[LLDP Policy] 下向き矢印をクリックし、ポリシー ダイアログボックスでポリシーの名前を追加し、送信状態を無効にして [submit] をクリックします。

- [Submit] をクリックします。

ステップ 5 次の手順に従ってリーフ インターフェイス プロファイルを作成します:

a) **[Fabric]** > **[Access Policies]** > **[Interfaces]** > **[Leaf Interfaces]** > **[Profiles]** をクリックします。

b) **Profiles** プロファイルを右クリックし、**Create Leaf Interface Profile** を選択し、次の手順に従います:

- **Name** フィールドに、**Leaf Interface Profile** の名前を入力します。

  オプション。説明を追加します。

- **Interface Selectors** フィールドで、**+** をクリックし、以下の情報を入力します:

  - **[名前]** フィールドに、インターフェイス セレクタの名前を入力します。

    オプション。説明を追加します。

  - **Interface IDs** フィールドに、このトンネルに含められる **Dot1q Tunnel** インターフェイス、または複数のインターフェイスの名前を入力します。

  - **Interface Policy Group** フィールドで、下向き矢印をクリックして、前に作成したインターフェイス ポリシー グループを選択します。

ステップ 6 トンネル設定のポートへのスタティック バインディングを作成するには、**[Tenant]** > **[Networking]** > **[Dot1Q Tunnels]** の順にクリックします。[Dot1Q Tunnels] を展開し、前に作成した **Dot1Q Tunnels <ポリシー名>** をクリックして、次の操作を実行します。

a) [Static Bindings] テーブルを展開して [Create Static Binding] ダイアログボックスを開きます。

b) [Port] フィールドで、ポートの種類を選択します。

c) [Node] フィールドで、ドロップダウンリストからノードを選択します。

d) [Path]フィールドで、ドロップダウンリストからインターフェイス パスを選択し、[Submit] をクリックします。

NX-OS スタイルの CLI を使用した 802.1Q トンネルの設定

NX-OS スタイル CLI を使用した802.1Q トンネルの設定

（注） **Dot1q トンネル** に含まれるインターフェイスのポート、ポート チャネル、仮想ポート チャネルを使用できます。手順の詳細にはポートの設定が含まれます。エッジおよびコア ポート チャネルと仮想ポート チャネルを設定するコマンドについては、下の例を参照してください。

次の手順で、**Dot1q トンネル** を作成し、NX-OS スタイル CLI を使用してトンネルで使用するインターフェイスを設定します。

（注） **Dot1q トンネル**には2 個以上のインターフェイスを含める必要があります。手順を繰り返し（または 2 個のインターフェイスをまとめて設定)、**Dot1q トンネル** で使用する各インターフェイスをマークします。この例で、2 個のインターフェイスは単一の顧客で使用されているエッジ スイッチ ポートとして設定されます。

次の手順を使用して、設定を次の手順を使用して、NX-OS スタイル CLI を使用して **Dot1q トンネル** を設定します。

1. トンネルで使用するインターフェイスを最低 2 個設定します。
2. **Dot1q トンネル** を作成します。
3. トンネルとすべてのインターフェイスを関連付けます。

始める前に

Dot1q トンネル を使用するテナントを設定します。

手順の概要

1. **configure**
2. 次の手順により 802.1Q で使用するための 2 個のインターフェイスを設定します。
3. **leaf** *ID*
4. **interface ethernet** *slot/port*
5. **switchport mode dot1q-tunnel** {**edgePort** | **corePort**}
6. 次の手順で 802.1q トンネルを作成します。
7. **leaf** *ID*
8. **interface ethernet***slot/port*
9. **switchport tenant***tenant-name***dot1q-tunnel** *tunnel-name*
10. トンネルとその他のインターフェイスを関連付けるには、ステップ 7 ～ 10 を繰り返します。

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `apic1# configure`	コンフィギュレーション モードに入ります。
ステップ 2	次の手順により 802.1Q で使用するための 2 個のインターフェイスを設定します。	
ステップ 3	**leaf** *ID* 例： `apic1(config)# leaf 101`	**Dot1q トンネル** のインターフェイスが配置されるリーフを特定します。

	コマンドまたはアクション	目的
ステップ 4	**interface ethernet** *slot/port* 例： `apic1(config-leaf)# interface ethernet 1/13-14`	トンネルのポートとしてマークされるインターフェイスを特定します。

ステップ 5	**switchport mode dot1q-tunnel** {**edgePort** | **corePort**}
	例：
	`apic1(config-leaf-if)# switchport mode dot1q-tunnel edgePort`
	`apic1(config-leaf-if)# exit`
	`apic1(config-leaf)# exit`
	`apic1(config)# exit`
		802.1Q トンネルで使用するインターフェイスをマークして、設定モードをそのままにします。
		この例では、エッジ ポートを使用するためにいくつかのインターフェイス設定を示します。トンネルに複数のインターフェイスを設定するには、手順3 ～ 5 を繰り返します。

ステップ 6	次の手順で 802.1q トンネルを作成します。	
ステップ 7	**leaf** *ID* 例： `apic1(config)# leaf 101`	インターフェイスが配置されているリーフに戻ります。
ステップ 8	**interface ethernet***slot/port* 例： `apic1(config-leaf)# interface ethernet 1/13-14`	トンネルに含まれるインターフェイスに戻ります。

ステップ 9	**switchport tenant***tenant-name***dot1q-tunnel** *tunnel-name*
	例：
	`apic1(config-leaf-if)# switchport  tenant tenant64  dot1q-tunnel  vrf64_edgetunnel`
	`apic1(config-leaf-if)# exit`
		トンネルにインターフェイスに関連付け、設定モードを終了します。

ステップ 10	トンネルとその他のインターフェイスを関連付けるには、ステップ 7 ～ 10 を繰り返します。	

例：NX-OS スタイル CLI でポートを使用する 802.1Q トンネルを設定する

この例では、2 つのポートを **Dot1q トンネル**で使用されるエッジ ポート インターフェイスとしてマークし、 さらに 2 つの ポートをコア ポート インターフェイスで使用されるものととしてマークし、トンネルを作成して、ポートをトンネルに関連付けます。

```
apic1# configure
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/13-14
apic1(config-leaf-if)# switchport mode dot1q-tunnel edgePort
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

```
apic1(config)leaf 102
apic1(config-leaf)# interface ethernet 1/10, 1/21
apic1(config-leaf-if)# switchport mode dot1q-tunnel corePort
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# tenant tenant64
apic1(config-tenant)# dot1q-tunnel vrf64_tunnel
apic1(config-tenant-tunnel)# l2protocol-tunnel cdp
apic1(config-tenant-tunnel)# l2protocol-tunnel lldp
apic1(config-tenant-tunnel)#  access-encap 200
apic1(config-tenant-tunnel)# mac-learning disable
apic1(config-tenant-tunnel)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface ethernet 1/13-14
apic1(config-leaf-if)# switchport tenant tenant64 dot1q-tunnel vrf64_tunnel
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 102
apic1(config-leaf)# interface ethernet 1/10, 1/21
apic1(config-leaf-if)# switchport tenant tenant64 dot1q-tunnel vrf64_tunnel
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

例：NX-OS スタイル CLI でポート チャネルを使用する 802.1Q トンネルを設定する

例では、このエッジポート8021 qインターフェイスとして2つのポートチャネルにマークし、2つ以上のポート チャネルをコア ポート802.1 qインターフェイスとしてマークして、**Dotq トンネル** を作成し、トンネルとポート チャネルを関連付けます。

```
apic1# configure
apic1(config)# tenant tenant64
apic1(config-tenant)# dot1q-tunnel vrf64_tunnel
apic1(config-tenant-tunnel)# l2protocol-tunnel cdp
apic1(config-tenant-tunnel)# l2protocol-tunnel lldp
apic1(config-tenant-tunnel)#  access-encap 200
apic1(config-tenant-tunnel)# mac-learning disable
apic1(config-tenant-tunnel)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 101
apic1(config-leaf)# interface port-channel pc1
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 1/2-3
apic1(config-leaf-if)# channel-group pc1
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface port-channel pc1
apic1(config-leaf-if)# switchport mode dot1q-tunnel edgePort
apic1(config-leaf-if)# switchport tenant tenant64 dot1q-tunnel vrf64_tunnel
apic1(config-tenant-tunnel)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 102
apic1(config-leaf)# interface port-channel pc2
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 1/4-5
```

```
apic1(config-leaf-if)# channel-group pc2
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface port-channel pc2
apic1(config-leaf-if)# switchport mode dot1q-tunnel corePort
apic1(config-leaf-if)# switchport tenant tenant64 dot1q-tunnel vrf64_tunnel
```

例：NX-OS スタイル CLI で仮想ポート チャネルを使用する 802.1Q トンネルを設定する

この例では、2 つの仮想ポート チャネル (vPC) を **Dot1q トンネル**の エッジポート 802.1Q インターフェイスとしてマークし、さらに2つのVPCをトンネルのためのコアポートインターフェイスとしてマークし、トンネルを作成して、仮想ポートチャネルをトンネルに関連付けています。

```
apic1# configure
apic1(config)# vpc domain explicit 1 leaf 101 102
apic1(config)# vpc context leaf 101 102
apic1(config-vpc)# interface vpc vpc1
apic1(config-vpc-if)# switchport mode dot1q-tunnel edgePort
apic1(config-vpc-if)# exit
apic1(config-vpc)# exit
apic1(config)# vpc domain explicit 1 leaf 103 104
apic1(config)# vpc context leaf 103 104
apic1(config-vpc)# interface vpc vpc2
apic1(config-vpc-if)# switchport mode dot1q-tunnel corePort
apic1(config-vpc-if)# exit
apic1(config-vpc)# exit
apic1(config)# tenant tenant64
apic1(config-tenant)# dot1q-tunnel vrf64_tunnel
apic1(config-tenant-tunnel)# l2protocol-tunnel cdp
apic1(config-tenant-tunnel)# l2protocol-tunnel lldp
apic1(config-tenant-tunnel)#  access-encap 200
apic1(config-tenant-tunnel)# mac-learning disable
apic1(config-tenant-tunnel)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 103
apic1(config-leaf)# interface ethernet 1/6
apic1(config-leaf-if)# channel-group vpc1 vpc
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config)# leaf 104
apic1(config-leaf)# interface ethernet 1/6
apic1(config-leaf-if)# channel-group vpc1 vpc
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
apic1(config-vpc)# interface vpc vpc1
apic1(config-vpc-if)# switchport tenant tenant64 dot1q-tunnel vrf64_tunnel
apic1(config-vpc-if)# exit
```

REST API を使用した 802.1Q トンネルの設定

REST API を使用してポートを持つトンネル 802.1 q の設定

作成、 **Dot1q トンネル** 、ポートを使用して、次の例などの手順でのインターフェイスを設定します。

始める前に

Dot1q トンネル を使用するテナントを設定します。

ステップ 1 次の例のように XML で REST API を使用して **Dot1q トンネル** を作成します。

例では、LLDP レイヤ 2 トンネリング プロトコルでトンネルを設定し、アクセス カプセル化 VLAN を追加し、トンネルで MAC ラーニングを無効にします。

例：

```
<fvTnlEPg name="VRF64_dot1q_tunnel" qiqL2ProtTunMask="lldp" accEncap="vlan-10"
fwdCtrl="mac-learn-disable"  >
    <fvRsTnlpathAtt  tDn="topology/pod-1/paths-101/pathep-[eth1/13]"/>
</fvTnlEPg>
```

ステップ 2 次の例のように XML で静的にバインディングするレイヤ 2 インターフェイス ポリシーを設定します。

この例では、エッジ スイッチ ポートにレイヤ 2 インターフェイス ポリシーを設定します。コア スイッチ ポートのポリシーを設定するには、`l2IfPol` MO で `edgePort` の代わりに `corePort` を使用します。

例：

```
<l2IfPol name="VRF64_L2_int_pol" qinq="edgePort"  />
```

ステップ 3 次の例のように、XML でリーフ アクセス ポート ポリシー グループにレイヤ 2 インターフェイス ポリシーを適用します。

例：

```
<infraAccPortGrp name="VRF64_L2_Port_Pol_Group" >
    <infraRsL2IfPol tnL2IfPolName="VRF64_L2_int_pol"/>
</infraAccPortGrp>
```

ステップ 4 次の例のように、XML でインターフェイス セレクタとともにリーフ プロファイルを設定します。

例：

```
<infraAccPortP name="VRF64_dot1q_leaf_profile" >
    <infraHPortS name="vrf64_access_port_selector" type="range">
        <infraPortBlk name="block2" toPort="15" toCard="1" fromPort="13" fromCard="1"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-VRF64_L2_Port_Pol_Group" />
    </infraHPortS>
</infraAccPortP>
```

例

次の例のように、2 個の POST 上のエッジ ポートのポート設定を示します。

Post 1 の XML：

```
<polUni>
  <infraInfra>
    <l2IfPol name="testL2IfPol" qinq="edgePort"/>
      <infraNodeP name="Node_101_phys">
      <infraLeafS name="phys101" type="range">
        <infraNodeBlk name="test" from_="101" to_="101"/>
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-phys21"/>
    </infraNodeP>
    <infraAccPortP name="phys21">
      <infraHPortS name="physHPortS" type="range">
        <infraPortBlk name="phys21" fromCard="1" toCard="1" fromPort="21" toPort="21"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-21"/>
      </infraHPortS>
    </infraAccPortP>
    <infraFuncP>
      <infraAccPortGrp name="21">
        <infraRsL2IfPol tnL2IfPolName="testL2IfPol"/>
        <infraRsAttEntP tDn="uni/infra/attentp-AttEntityProf1701"/>
      </infraAccPortGrp>
    </infraFuncP>
    <l2IfPol name='testL2IfPol' qinq='edgePort'/>
    <infraAttEntityP name="AttEntityProf1701">
      <infraRsDomP tDn="uni/phys-dom1701"/>
    </infraAttEntityP>
  </infraInfra>
</polUni>
```

Post 2 の XML：

```
<polUni>
  <fvTenant dn="uni/tn-Coke" name="Coke">
    <fvTnlEPg name="WEB5" qiqL2ProtTunMask="lldp" accEncap="vlan-10"
fwdCtrl="mac-learn-disable" >
        <fvRsTnlpathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/21]"/>
      </fvTnlEPg>
  </fvTenant>
</polUni>
```

REST API を使用した PC を持つトンネル 802.1Q の設定

PC を使用して **Dot1q トンネル** を作衛して、次の例のような手順でインターフェイスを設定します。

始める前に

Dot1q トンネル を使用するテナントを設定します。

ステップ 1 次の例のように XML で REST API を使用して **Dot1q トンネル** を作成します。

例では、LLDP レイヤ 2 トンネリング プロトコルでトンネルを設定し、アクセス カプセル化 VLAN を追加し、トンネルで MAC ラーニングを無効にします。

例：

```
<fvTnlEPg name="WEB" qiqL2ProtTunMask=lldp accEncap="vlan-10" fwdCtrl="mac-learn-disable" >
    <fvRsTnlpathAtt tDn="topology/pod-1/paths-101/pathep-[po2]"/>
  </fvTnlEPg>
```

ステップ 2 次の例のように XML で静的にバインディングするレイヤ 2 インターフェイス ポリシーを設定します。

この例では、エッジ スイッチ ポートにレイヤ 2 インターフェイス ポリシーを設定します。コア スイッチ ポートのレイヤ 2 インターフェイス ポリシーを設定するには、`l2IfPol` MO の `edgePort` 代わりに `corePort` を使用します。

例：

```
<l2IfPol name="testL2IfPol" qinq="edgePort"/>
```

ステップ 3 次の例のように、XML で PC インターフェイス ポリシー グループにレイヤ 2 インターフェイス ポリシーを適用します。

例：

```
<infraAccBndlGrp name="po2" lagT="link">
   <infraRsL2IfPol tnL2IfPolName="testL2IfPol"/>
    </infraAccBndlGrp>
```

ステップ 4 次の例のように、XML でインターフェイス セレクタを持つリーフ プロファイルを設定します。

例：

```
<infraAccPortP name="PC">
   <infraHPortS name="allow" type="range">
    <infraPortBlk name="block2" fromCard="1" toCard="1" fromPort="10" toPort="11" />
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-po2"/>
   </infraHPortS>
  </infraAccPortP>
```

例

次の例では、2 個のポストの PC 設定を示します。

この例では、エッジ ポートとして PC ポートを設定します。コア ポートとして設定を使用するには、Post 1 にある `l2IfPol` MO の `edgePort` 代わりに `corePort` を使用します。

Post 1 の XML：

```
<infraInfra dn="uni/infra">
    <infraNodeP name="bLeaf3">
        <infraLeafS name="leafs3" type="range">
            <infraNodeBlk name="nblk3" from_="101" to_="101">
            </infraNodeBlk>
        </infraLeafS>
        <infraRsAccPortP tDn="uni/infra/accportprof-shipping3"/>
    </infraNodeP>
    <infraAccPortP name="shipping3">
    <infraHPortS name="pselc3" type="range">
        <infraPortBlk name="blk3" fromCard="1" toCard="1" fromPort="24" toPort="25"/>
```

```
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-accountingLag3" />
    </infraHPortS>
  </infraAccPortP>
<infraFuncP>
   <infraAccBndlGrp name="accountingLag3" lagT='link'>
    <infraRsAttEntP tDn="uni/infra/attentp-default"/>
        <infraRsLacpPol tnLacpLagPolName='accountingLacp3'/>
        <infraRsL2IfPol tnL2IfPolName="testL2IfPol3"/>
   </infraAccBndlGrp>
</infraFuncP>
<lacpLagPol name='accountingLacp3' ctrl='15' descr='accounting' maxLinks='14' minLinks='1'
 mode='active' />
<l2IfPol name='testL2IfPol3' qinq='edgePort'/>
<infraAttEntityP name="default">
 </infraAttEntityP>
</infraInfra>
```

Post 2 の XML：

```
<polUni>
  <fvTenant dn="uni/tn-Coke" name="Coke">
    <!-- bridge domain -->
    <fvTnlEPg name="WEB6" qiqL2ProtTunMask="lldp" accEncap="vlan-10"
fwdCtrl="mac-learn-disable" >
        <fvRsTnlpathAtt tDn="topology/pod-1/paths-101/pathep-[accountingLag1]"/>
      </fvTnlEPg>
  </fvTenant>
</polUni>
```

REST API を使用した vPC での 802.1 Q トンネルの設定

vPC を使用して **Dot1q トンネル**を作成し、次の例のような手順でインターフェイスを設定します。

始める前に

Dot1q トンネル を使用するテナントを設定します。

ステップ 1 次の例のように、XML とともに REST API を使用して 802.1 q トンネルを作成します。

例ではレイヤ 2 トンネル プロトコルとともにトンネルを設定し、アクセス カプセル化 VLAN を追加し、トンネルで MAC 学習を無効にします。

例：

```
<fvTnlEPg name="WEB" qiqL2ProtTunMask=lldp accEncap="vlan-10" fwdCtrl="mac-learn-disable" >
        <fvRsTnlpathAtt tDn="topology/pod-1/protpaths-101-102/pathep-[po4]" />
      </fvTnlEPg>
```

ステップ 2 次の例のように、XMLとともに静的バインディングでレイヤ2インターフェイス ポリシーを設定します。

この例では、エッジ スイッチ ポートのレイヤ 2 インターフェイス ポリシーを設定します。コア スイッチ ポートのレイヤ2インターフェイス ポリシーを設定するには、`qinq ="corePort"` ポート タイプを使用します。

例：

```
<l2IfPol name="testL2IfPol" qinq="edgePort"/>
```

ステップ 3 次の例のように、XML を持つ VPC インターフェイス ポリシー グループにレイヤ 2 インターフェイス ポリシーを適用します。

例：

```
<infraAccBndlGrp name="po4" lagT="node">
   <infraRsL2IfPol tnL2IfPolName="testL2IfPol"/>
   </infraAccBndlGrp>
```

ステップ 4 次の例のように、XML を持つインターフェイス セレクタでリーフ プロファイルを設定します。

例：

```
  <infraAccPortP name="VPC">
   <infraHPortS name="allow" type="range">
    <infraPortBlk name="block2" fromCard="1" toCard="1" fromPort="10" toPort="11" />
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-po4"/>
   </infraHPortS>
  </infraAccPortP>
```

例

次の例は、3つのポストでのvPC設定を示しています。

この例では、vPC ポートをエッジ ポートとして設定します。コア ポートとして設定するには、ポスト 2 で `corePort` を `edgePort` の代わりに `l2IfPol` MO で使用します。

Post 1 の XML：

```
<polUni>
    <fabricInst>
        <fabricProtPol pairT="explicit">
            <fabricExplicitGEp name="101-102-vpc1" id="30">
                <fabricNodePEp id="101"/>
                <fabricNodePEp id="102"/>
            </fabricExplicitGEp>
        </fabricProtPol>
    </fabricInst>
</polUni>
```

Post 2 の XML：

```
<infraInfra dn="uni/infra">
    <infraNodeP name="bLeaf1">
        <infraLeafS name="leafs" type="range">
            <infraNodeBlk name="nblk" from_="101" to_="101">
            </infraNodeBlk>
        </infraLeafS>
        <infraRsAccPortP tDn="uni/infra/accportprof-shipping1"/>
    </infraNodeP>

    <infraNodeP name="bLeaf2">
        <infraLeafS name="leafs" type="range">
            <infraNodeBlk name="nblk" from_="102" to_="102">
            </infraNodeBlk>
        </infraLeafS>
        <infraRsAccPortP tDn="uni/infra/accportprof-shipping2"/>
    </infraNodeP>
```

```
  <infraAccPortP name="shipping1">
    <infraHPortS name="pselc" type="range">
        <infraPortBlk name="blk" fromCard="1" toCard="1" fromPort="4" toPort="4"/>

        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-accountingLag1" />
    </infraHPortS>
  </infraAccPortP>

  <infraAccPortP name="shipping2">
    <infraHPortS name="pselc" type="range">
        <infraPortBlk name="blk" fromCard="1" toCard="1" fromPort="2" toPort="2"/>

        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-accountingLag2" />
    </infraHPortS>
  </infraAccPortP>

<infraFuncP>
   <infraAccBndlGrp name="accountingLag1" lagT='node'>
    <infraRsAttEntP tDn="uni/infra/attentp-default"/>
        <infraRsLacpPol tnLacpLagPolName='accountingLacp1'/>
        <infraRsL2IfPol tnL2IfPolName="testL2IfPol"/>
   </infraAccBndlGrp>
   <infraAccBndlGrp name="accountingLag2" lagT='node'>
    <infraRsAttEntP tDn="uni/infra/attentp-default"/>
         <infraRsLacpPol tnLacpLagPolName='accountingLacp1'/>
         <infraRsL2IfPol tnL2IfPolName="testL2IfPol"/>
   </infraAccBndlGrp>
</infraFuncP>
<lacpLagPol name='accountingLacp1' ctrl='15' descr='accounting' maxLinks='14' minLinks='1'
 mode='active' />
<l2IfPol name='testL2IfPol' qinq='edgePort'/>

 <infraAttEntityP name="default">
 </infraAttEntityP>
</infraInfra>
```

Post 3 の XML：

```
<polUni>
  <fvTenant dn="uni/tn-Coke" name="Coke">
    <!-- bridge domain -->
    <fvTnlEPg name="WEB6" qiqL2ProtTunMask="lldp" accEncap="vlan-10"
fwdCtrl="mac-learn-disable" >
       <fvRsTnlpathAtt tDn="topology/pod-1/protpaths-101-102/pathep-[accountingLag2]"/>

      </fvTnlEPg>
  </fvTenant>
</polUni>
```

翻訳について
このドキュメントは、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。