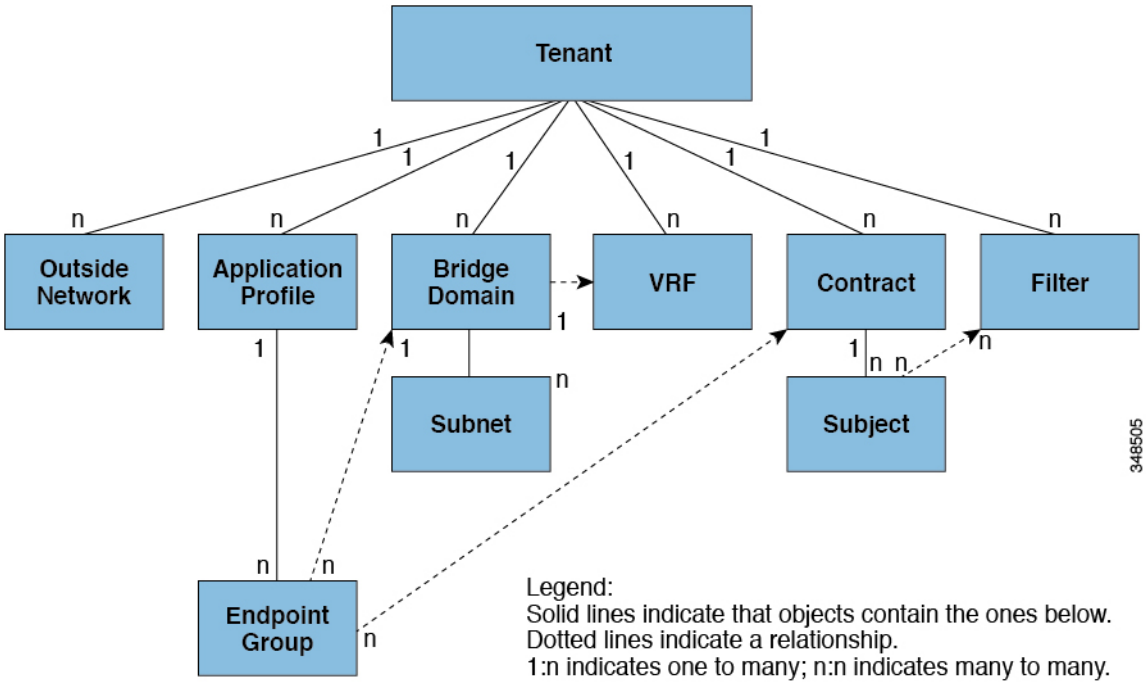
Basic User Tenant Configuration

This chapter contains the following sections:

Tenants

A tenant (`fvTenant`) is a logical container for application policies that enable an administrator to exercise domain-based access control. A tenant represents a unit of isolation from a policy perspective, but it does not represent a private network. Tenants can represent a customer in a service provider setting, an organization or domain in an enterprise setting, or just a convenient grouping of policies. The following figure provides an overview of the tenant portion of the management information tree (MIT).

Figure 1: Tenants



Tenants can be isolated from one another or can share resources. The primary elements that the tenant contains are filters, contracts, outside networks, bridge domains, Virtual Routing and Forwarding (VRF) instances, and application profiles that contain endpoint groups (EPGs). Entities in the tenant inherit its policies. VRFs are also known as contexts; each VRF can be associated with multiple bridge domains.

Note In the APIC GUI under the tenant navigation path, a VRF (context) is called a private network.

Tenants are logical containers for application policies. The fabric can contain multiple tenants. You must configure a tenant before you can deploy any Layer 4 to Layer 7 services. The ACI fabric supports IPv4, IPv6, and dual-stack configurations for tenant networking.

Routing Within the Tenant

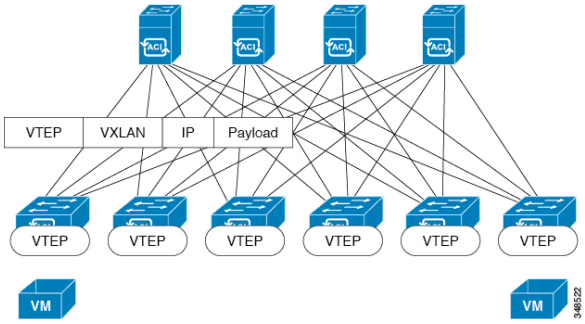
The Application Centric Infrastructure (ACI) fabric provides tenant default gateway functionality and routes between the fabric virtual extensible local area (VXLAN) networks. For each tenant, the fabric provides a virtual default gateway or Switched Virtual Interface (SVI) whenever a subnet is created on the APIC. This spans any switch that has a connected endpoint for that tenant subnet. Each ingress interface supports the default gateway interface and all of the ingress interfaces across the fabric share the same router IP address and MAC address for a given tenant subnet.

Layer 3 VNIDs Facilitate Transporting Inter-subnet Tenant Traffic

The ACI fabric provides tenant default gateway functionality that routes between the ACI fabric VXLAN networks. For each tenant, the fabric provides a virtual default gateway that spans all of the leaf switches assigned to the tenant. It does this at the ingress interface of the first leaf switch connected to the endpoint. Each ingress interface supports the default gateway interface. All of the ingress interfaces across the fabric share the same router IP address and MAC address for a given tenant subnet.

The ACI fabric decouples the tenant endpoint address, its identifier, from the location of the endpoint that is defined by its locator or VXLAN tunnel endpoint (VTEP) address. Forwarding within the fabric is between VTEPs. The following figure shows decoupled identity and location in ACI.

Figure 2: ACI Decouples Identity and Location



VXLAN uses VTEP devices to map tenant end devices to VXLAN segments and to perform VXLAN encapsulation and de-encapsulation. Each VTEP function has two interfaces:

- A switch interface on the local LAN segment to support local endpoint communication through bridging
- An IP interface to the transport IP network

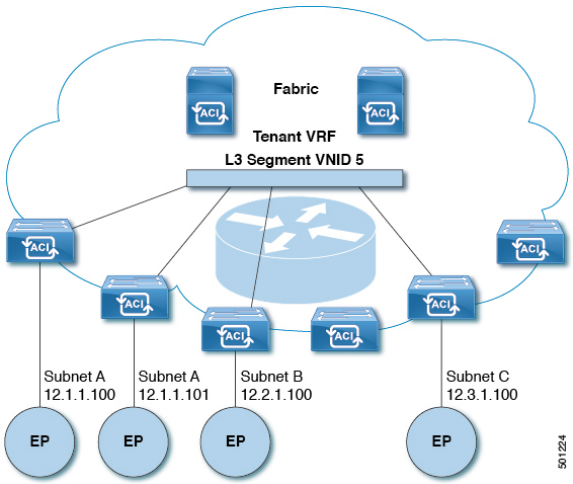
The IP interface has a unique IP address that identifies the VTEP device on the transport IP network known as the infrastructure VLAN. The VTEP device uses this IP address to encapsulate Ethernet frames and transmit the encapsulated packets to the transport network through the IP interface. A VTEP device also discovers the remote VTEPs for its VXLAN segments and learns remote MAC Address-to-VTEP mappings through its IP interface.

The VTEP in ACI maps the internal tenant MAC or IP address to a location using a distributed mapping database. After the VTEP completes a lookup, the VTEP sends the original data packet encapsulated in VXLAN with the destination address of the VTEP on the destination leaf switch. The destination leaf switch de-encapsulates the packet and sends it to the receiving host. With this model, ACI uses a full mesh, single hop, loop-free topology without the need to use the spanning-tree protocol to prevent loops.

The VXLAN segments are independent of the underlying network topology; conversely, the underlying IP network between VTEPs is independent of the VXLAN overlay. It routes the encapsulated packets based on the outer IP address header, which has the initiating VTEP as the source IP address and the terminating VTEP as the destination IP address.

The following figure shows how routing within the tenant is done.

Figure 3: Layer 3 VNIDs Transport ACI Inter-subnet Tenant Traffic



For each tenant VRF in the fabric, ACI assigns a single L3 VNID. ACI transports traffic across the fabric according to the L3 VNID. At the egress leaf switch, ACI routes the packet from the L3 VNID to the VNID of the egress subnet.

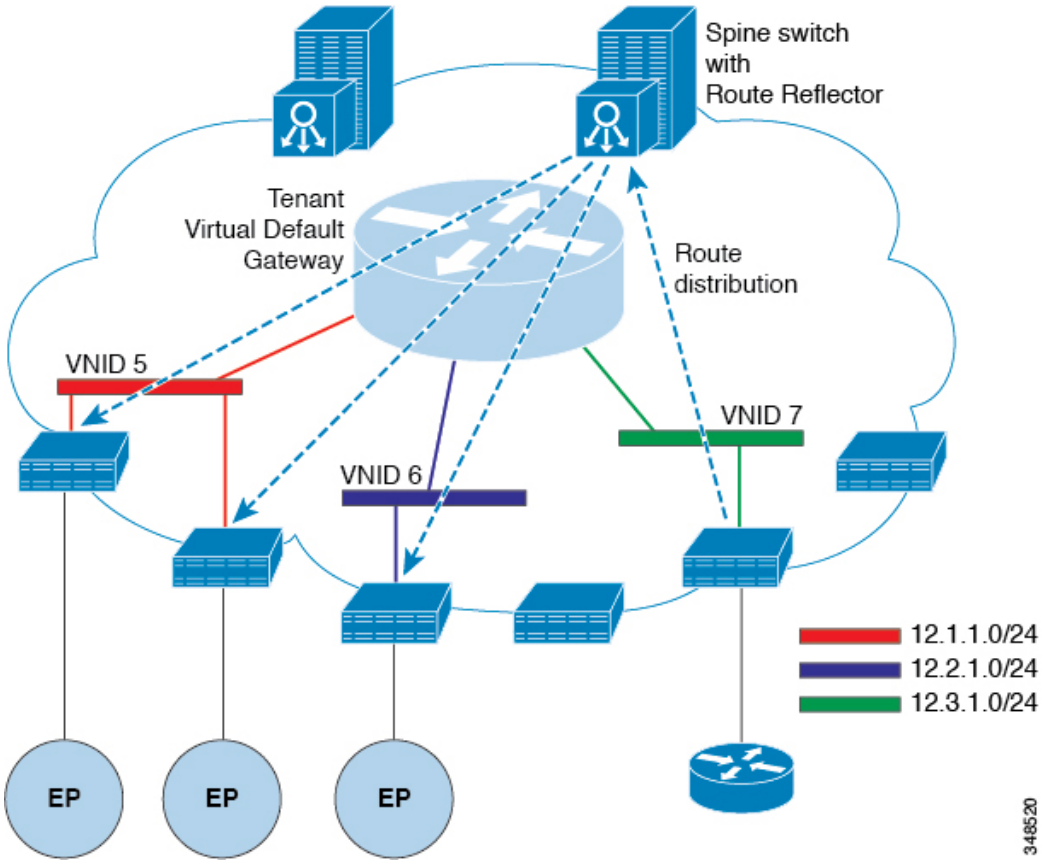
Traffic arriving at the fabric ingress that is sent to the ACI fabric default gateway is routed into the Layer 3 VNID. This provides very efficient forwarding in the fabric for traffic routed within the tenant. For example, with this model, traffic between 2 VMs belonging to the same tenant, on the same physical host, but on different subnets, only needs to travel to the ingress switch interface before being routed (using the minimal path cost) to the correct destination.

To distribute external routes within the fabric, ACI route reflectors use multiprotocol BGP (MP-BGP). The fabric administrator provides the autonomous system (AS) number and specifies the spine switches that become route reflectors.

Router Peering and Route Distribution

As shown in the figure below, when the routing peer model is used, the leaf switch interface is statically configured to peer with the external router's routing protocol.

Figure 4: Router Peering

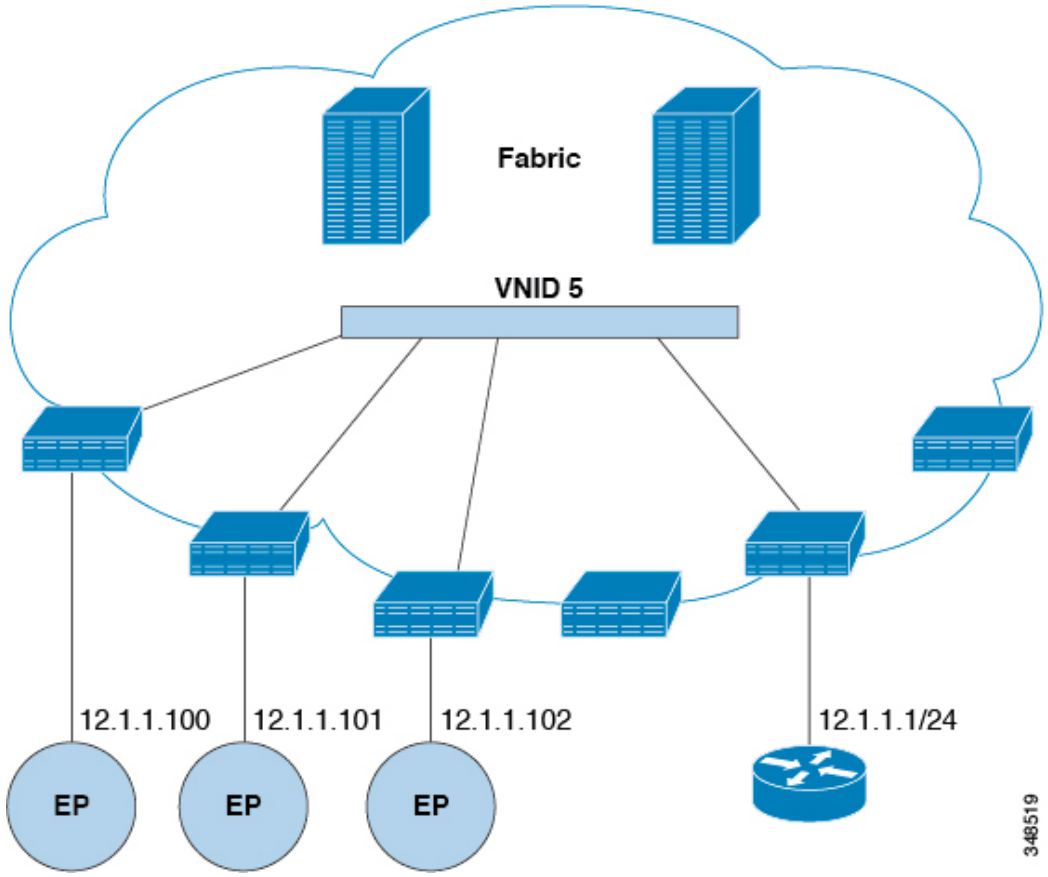


The routes that are learned through peering are sent to the spine switches. The spine switches act as route reflectors and distribute the external routes to all of the leaf switches that have interfaces that belong to the same tenant. These routes are longest prefix match (LPM) summarized addresses and are placed in the leaf switch's forwarding table with the VTEP IP address of the remote leaf switch where the external router is connected. WAN routes have no forwarding proxy. If the WAN routes do not fit in the leaf switch's forwarding table, the traffic is dropped. Because the external router is not the default gateway, packets from the tenant endpoints (EPs) are sent to the default gateway in the ACI fabric.

Bridged Interface to an External Router

As shown in the figure below, when the leaf switch interface is configured as a bridged interface, the default gateway for the tenant VNID is the external router.

Figure 5: Bridged External Router



The ACI fabric is unaware of the presence of the external router and the APIC statically assigns the leaf switch interface to its EPG.

Configuring Route Reflectors

ACI fabric route reflectors use multiprotocol BGP (MP-BGP) to distribute external routes within the fabric. To enable route reflectors in the ACI fabric, the fabric administrator must select the spine switches that will be the route reflectors, and provide the autonomous system (AS) number. It is recommended to configure at least two spine nodes per pod as MP-BGP route reflectors for redundancy.

After route reflectors are enabled in the ACI fabric, administrators can configure connectivity to external networks through leaf nodes using a component called Layer 3 Out (L3Out). A leaf node configured with an L3Out is called a border leaf. The border leaf exchanges routes with a connected external device via a routing protocol specified in the L3Out. You can also configure static routes via L3Outs.

After both L3Outs and spine route reflectors are deployed, border leaf nodes learn external routes via L3Outs, and those external routes are distributed to all leaf nodes in the fabric via spine MP-BGP route reflectors.

Check the *Verified Scalability Guide for Cisco APIC* for your release to find the maximum number of routes supported by a leaf.

Configuring External Connectivity Using a Layer 3 Out

This section provides a step-by-step configuration required for the ACI fabric to connect to an external routed network through L3Outs and MP-BGP route reflectors.

This example uses Open Shortest Path First (OSPF) as the routing protocol in an L3Out under the 'mgmt' tenant.

Configuring an MP-BGP Route Reflector Using the GUI

Procedure

Step 1 On the menu bar, choose **System** > **System Settings**.

Step 2 In the **Navigation** pane, right-click **BGP Route Reflector**, and click **Create Route Reflector Node Policy EP**.

Step 3 In the **Create Route Reflector Node Policy EP** dialog box, from the **Spine Node** drop-down list, choose the appropriate spine node. Click **Submit**.

Note Repeat the above steps to add additional spine nodes as required.

The spine switch is marked as the route reflector node.

Step 4 In the **BGP Route Reflector** properties area, in the **Autonomous System Number** field, choose the appropriate number. Click **Submit**.

Note The autonomous system number must match the leaf connected router configuration if Border Gateway Protocol (BGP) is configured on the router. If you are using routes learned using static or Open Shortest Path First (OSPF), the autonomous system number value can be any valid value.

Step 5 On the menu bar, choose **Fabric** > **Fabric Policies** > **POD Policies**.

Step 6 In the **Navigation** pane, expand and right-click **Policy Groups**, and click **Create POD Policy Group**.

Step 7 In the **Create POD Policy Group** dialog box, in the **Name** field, enter the name of a pod policy group.

Step 8 In the **BGP Route Reflector Policy** drop-down list, choose the appropriate policy (default). Click **Submit**.
The BGP route reflector policy is associated with the route reflector pod policy group, and the BGP process is enabled on the leaf switches.

Step 9 In the **Navigation** pane, choose **Pod Policies** > **Profiles** > **default**. In the **Work** pane, from the **Fabric Policy Group** drop-down list, choose the pod policy that was created earlier. Click **Submit**.
The pod policy group is now applied to the fabric policy group.

Configuring an MP-BGP Route Reflector for the ACI Fabric

To distribute routes within the ACI fabric, an MP-BGP process must first be operating, and the spine switches must be configured as BGP route reflectors.

The following is an example of an MP-BGP route reflector configuration:

Note In this example, the BGP fabric ASN is 100. Spine switches 104 and 105 are chosen as MP-BGP route-reflectors.

```
apic1(config)# bgp-fabric
apic1(config-bgp-fabric)# asn 100
apic1(config-bgp-fabric)# route-reflector spine 104,105
```

Configuring an MP-BGP Route Reflector Using the REST API

Procedure

Step 1 Mark the spine switches as route reflectors.

Example:

```
POST https://apic-ip-address/api/policymgr/mo/uni/fabric.xml

<bgpInstPol name="default">
  <bgpAsP asn="1" />
  <bgpRRP>
     <bgpRRNodePEp id="<spine_id1>"/>
     <bgpRRNodePEp id="<spine_id2>"/>
  </bgpRRP>
</bgpInstPol>
```

Step 2 Set up the pod selector using the following post.

Example:

For the FuncP setup—

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

<fabricFuncP>
  <fabricPodPGrp name="bgpRRPodGrp">
    <fabricRsPodPGrpBGPRRP tnBgpInstPolName="default" />
  </fabricPodPGrp>
</fabricFuncP>
```

Example:

For the PodP setup—

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

<fabricPodP name="default">
  <fabricPodS name="default" type="ALL">
    <fabricRsPodPGrp tDn="uni/fabric/funcprof/podpgrp-bgpRRPodGrp"/>
  </fabricPodS>
</fabricPodP>
```

Verifying the MP-BGP Route Reflector Configuration

Procedure

Step 1 Verify the configuration by performing the following actions:

a) Use secure shell (SSH) to log in as an administrator to each leaf switch as required.

b) Enter the **show processes | grep bgp** command to verify the state is S.

If the state is NR (not running), the configuration was not successful.

Step 2 Verify that the autonomous system number is configured in the spine switches by performing the following actions:

a) Use the SSH to log in as an administrator to each spine switch as required.

b) Execute the following commands from the shell window

Example:

cd /mit/sys/bgp/inst

Example:

grep asn summary

The configured autonomous system number must be displayed. If the autonomous system number value displays as 0, the configuration was not successful.

Creating an OSPF L3Out for Management Tenant Using the GUI

- You must verify that the router ID and the logical interface profile IP address are different and do not overlap.
- The following steps are for creating an OSPF L3Out for a management tenant. To create an OSPF L3Out for a tenant, you must choose a tenant and create a VRF for the tenant.
- For more details, see *Cisco APIC and Transit Routing*.

Procedure

Step 1 On the menu bar, choose **Tenants** > **mgmt**.

Step 2 In the **Navigation** pane, expand **Networking** > **L3Outs**.

Step 3 Right-click **L3Outs**, and click **Create L3Out**.

The **Create L3Out** wizard appears.

Step 4 In the **Identity** window in the **Create L3Out** wizard, perform the following actions:

a) In the **Name** field, enter a name (RtdOut).

b) In the **VRF** field, from the drop-down list, choose the VRF (inb).

Note This step associates the routed outside with the in-band VRF.

c) From the **L3 Domain** drop-down list, choose the appropriate domain.
d) Check the **OSPF** check box.
e) In the **OSPF Area ID** field, enter an area ID.
f) In the **OSPF Area Control** field, check the appropriate check box.
g) In the **OSPF Area Type** field, choose the appropriate area type.
h) In the **OSPF Area Cost** field, choose the appropriate value.
i) Click **Next**.

The **Nodes and Interfaces** window appears.

Step 5 In the **Nodes and Interfaces** window, perform the following actions:

a) Uncheck the **Use Defaults** box.

This allows you to edit the **Node Profile Name** field.

b) In the **Node Profile Name** field, enter a name for the node profile. (borderLeaf).
c) In the **Node ID** field, from the drop-down list, choose the first node. (leaf1).
d) In the **Router ID** field, enter a unique router ID.
e) In the **Loopback Address** field, use a different IP address or leave this field empty if you do not want to use the router ID for the loopback address.

Note The **Loopback Address** field is automatically populated with the same entry that you provide in the **Router ID** field. This is the equivalent of the **Use Router ID for Loopback Address** option in previous builds. Use a different IP address or leave this field empty if you do not want to use the router ID for the loopback address.

f) Enter the appropriate information in the **Interface**, **IP Address**, **Interface Profile Name** and **MTU** fields for this node, if necessary.
g) In the **Nodes** field, click + icon to add a second set of fields for another node.

Note You are adding a second node ID.

h) In the **Node ID** field, from the drop-down list, choose the first node. (leaf1).
i) In the **Router ID** field, enter a unique router ID.
j) In the **Loopback Address** field, use a different IP address or leave this field empty if you do not want to use the router ID for the loopback address.

Note The **Loopback Address** field is automatically populated with the same entry that you provide in the **Router ID** field. This is the equivalent of the **Use Router ID for Loopback Address** option in previous builds. Use a different IP address or leave this field empty if you do not want to use the router ID for the loopback address.

k) Enter the appropriate information in the **Interface**, **IP Address**, **Interface Profile Name** and **MTU** fields for this node, if necessary.
l) Click **Next**.

The **Protocols** window appears.

Step 6 In the **Protocols** window, in the **Policy** area, click **default**, then click **Next**.

The **External EPG** window appears.

Step 7 In the **External EPG** window, perform the following actions:

a) In the **Name** field, enter a name for the external network (extMgmt).
b) Uncheck the **Default EPG for all external networks** field.

The **Subnets** area appears.

c) Click + to access the **Create Subnet** dialog box.
d) In the **Create Subnet** dialog box, in the **IP address** field, enter an IP address and mask for the subnet.
e) In the **Scope** field, check the desired check boxes. Click **OK**.
f) In the **External EPG** dialog box, click **Finish**.

Note In the **Work** pane, in the **L3Outs** area, the L3Out icon (RtdOut) is now displayed.

Creating an OSPF External Routed Network for a Tenant Using the NX-OS CLI

Configuring external routed network connectivity involves the following steps:

1. Create a VRF under Tenant.
2. Configure L3 networking configuration for the VRF on the border leaf switches, which are connected to the external routed network. This configuration includes interfaces, routing protocols (BGP, OSPF, EIGRP), protocol parameters, route-maps.
3. Configure policies by creating external-L3 EPGs under tenant and deploy these EPGs on the border leaf switches. External routed subnets on a VRF which share the same policy within the ACI fabric form one "External L3 EPG" or one "prefix EPG".

Configuration is realized in two modes:

- Tenant mode: VRF creation and external-L3 EPG configuration
- Leaf mode: L3 networking configuration and external-L3 EPG deployment

The following steps are for creating an OSPF external routed network for a tenant. To create an OSPF external routed network for a tenant, you must choose a tenant and then create a VRF for the tenant.

Note The examples in this section show how to provide external routed connectivity to the "web" epg in the "OnlineStore" application for tenant "exampleCorp".

Procedure

Step 1 Configure the VLAN domain.

Example:

```
apic1(config)# vlan-domain dom_exampleCorp
apic1(config-vlan)# vlan 5-1000
apic1(config-vlan)# exit
```

Step 2 Configure the tenant VRF and enable policy enforcement on the VRF.

Example:

```
apic1(config)# tenant exampleCorp
apic1(config-tenant)# vrf context
 exampleCorp_v1
apic1(config-tenant-vrf)# contract enforce
apic1(config-tenant-vrf)# exit
```

Step 3 Configure the tenant BD and mark the gateway IP as “public”. The entry "scope public" makes this gateway address available for advertisement through the routing protocol for external-L3 network.

Example:

```
apic1(config-tenant)# bridge-domain exampleCorp_b1
apic1(config-tenant-bd)# vrf member exampleCorp_v1
apic1(config-tenant-bd)# exit
apic1(config-tenant)# interface bridge-domain exampleCorp_b1
apic1(config-tenant-interface)# ip address 172.1.1.1/24 scope public
apic1(config-tenant-interface)# exit
```

Step 4 Configure the VRF on a leaf.

Example:

```
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant exampleCorp vrf exampleCorp_v1
```

Step 5 Configure the OSPF area and add the route map.

Example:

```
apic1(config-leaf)# router ospf default
apic1(config-leaf-ospf)# vrf member tenant exampleCorp vrf exampleCorp_v1
apic1(config-leaf-ospf-vrf)# area 0.0.0.1 route-map map100 out
apic1(config-leaf-ospf-vrf)# exit
apic1(config-leaf-ospf)# exit
```

Step 6 Assign the VRF to the interface (sub-interface in this example) and enable the OSPF area.

Example:

Note For the sub-interface configuration, the main interface (ethernet 1/11 in this example) must be converted to an L3 port through “no switchport” and assigned a vlan-domain (dom_exampleCorp in this example) that contains the encapsulation VLAN used by the sub-interface. In the sub-interface ethernet1/11.500, 500 is the encapsulation VLAN.

```
apic1(config-leaf)# interface ethernet 1/11
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# vlan-domain member dom_exampleCorp
apic1(config-leaf-if)# exit
apic1(config-leaf)# interface ethernet 1/11.500
apic1(config-leaf-if)# vrf member tenant exampleCorp vrf exampleCorp_v1
apic1(config-leaf-if)# ip address 157.10.1.1/24
apic1(config-leaf-if)# ip router ospf default area 0.0.0.1
```

Step 7 Configure the external-L3 EPG policy. This includes the subnet to match for identifying the external subnet and consuming the contract to connect with the epg "web".

Example:

```
apic1(config)# tenant t100
apic1(config-tenant)# external-l3 epg  l3epg100
apic1(config-tenant-l3ext-epg)# vrf member v100
apic1(config-tenant-l3ext-epg)# match ip 145.10.1.0/24
apic1(config-tenant-l3ext-epg)# contract consumer web
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)#exit
```

Step 8 Deploy the external-L3 EPG on the leaf switch.

Example:

```
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant t100 vrf v100
apic1(config-leaf-vrf)# external-l3 epg l3epg100
```

Creating Tenants, VRFs, and Bridge Domains

Tenants Overview

- A tenant contains policies that enable qualified users domain-based access control. Qualified users can access privileges such as tenant administration and networking administration.
- A user requires read/write privileges for accessing and configuring policies in a domain. A tenant user can have specific privileges into one or more domains.
- In a multitenancy environment, a tenant provides group user access privileges so that resources are isolated from one another (such as for endpoint groups and networking). These privileges also enable different users to manage different tenants.

Tenant Creation

A tenant contains primary elements such as filters, contracts, bridge domains, and application profiles that you can create after you first create a tenant.

VRF and Bridge Domains

You can create and specify a VRF and a bridge domain for the tenant. The defined bridge domain element subnets reference a corresponding Layer 3 context.

For details about enabling IPv6 Neighbor Discovery see*IPv6 and Neighbor Discovery* in *Cisco APIC Layer 3 Networking Guide*.

Creating a Tenant, VRF, and Bridge Domain Using the GUI

If you have a public subnet when you configure the routed outside, you must associate the bridge domain with the outside configuration.

Procedure

Step 1 On the menu bar, choose **Tenants** > **Add Tenant**.

Step 2 In the **Create Tenant** dialog box, perform the following tasks:

a) In the **Name** field, enter a name.
b) Click the **Security Domains +** icon to open the **Create Security Domain** dialog box.
c) In the **Name** field, enter a name for the security domain. Click **Submit**.
d) In the **Create Tenant** dialog box, check the check box for the security domain that you created, and click **Submit**.

Step 3 In the **Navigation** pane, expand **Tenant-name** > **Networking**, and in the **Work** pane, drag the **VRF** icon to the canvas to open the **Create VRF** dialog box, and perform the following tasks:

a) In the **Name** field, enter a name.
b) Click **Submit** to complete the VRF configuration.

Step 4 In the **Networking** pane, drag the **BD** icon to the canvas while connecting it to the **VRF** icon. In the **Create Bridge Domain** dialog box that displays, perform the following tasks:

a) In the **Name** field, enter a name.
b) Click the **L3 Configurations** tab.
c) Expand **Subnets** to open the **Create Subnet** dialog box, enter the subnet mask in the **Gateway IP** field and click **OK**.
d) Click **Submit** to complete bridge domain configuration.

Step 5 In the **Networks** pane, drag the **L3** icon down to the canvas while connecting it to the **VRF** icon. In the **Create Routed Outside** dialog box that displays, perform the following tasks:

a) In the **Name** field, enter a name.
b) Expand **Nodes And Interfaces Protocol Profiles** to open the **Create Node Profile** dialog box.
c) In the **Name** field, enter a name.
d) Expand **Nodes** to open the **Select Node** dialog box.
e) In the **Node ID** field, choose a node from the drop-down list.
f) In the **Router ID** field, enter the router ID.
g) Expand **Static Routes** to open the **Create Static Route** dialog box.
h) In the **Prefix** field, enter the IPv4 or IPv6 address.
i) Expand **Next Hop Addresses** and in the **Next Hop IP** field, enter the IPv4 or IPv6 address.
j) In the **Preference** field, enter a number, then click **UPDATE** and then **OK**.
k) In the **Select Node** dialog box, click **OK**.
l) In the **Create Node Profile** dialog box, click **OK**.
m) Check the **BGP**, **OSPF**, or **EIGRP** check boxes if desired, and click **NEXT**. Click **OK** to complete the Layer 3 configuration.

To confirm L3 configuration, in the **Navigation** pane, expand **Networking** > **VRFs**.

Deploying EPGs

Statically Deploying an EPG on a Specific Port

This topic provides a typical example of how to statically deploy an EPG on a specific port when using Cisco APIC.

Deploying an EPG on a Specific Node or Port Using the GUI

Before you begin

The tenant where you deploy the EPG is already created.

You can create an EPG on a specific node or a specific port on a node.

Procedure

Step 1 Log in to the Cisco APIC.

Step 2 Choose **Tenants** > ***tenant*** .

Step 3 In the left navigation pane, expand ***tenant*** , **Application Profiles**, and the ***application profile*** .

Step 4 Right-click **Application EPGs** and choose **Create Application EPG**.

Step 5 In the **Create Application EPG STEP 1 > Identity** dialog box, complete the following steps:

a) In the **Name** field, enter a name for the EPG.

b) From the **Bridge Domain** drop-down list, choose a bridge domain.

c) Check the **Statically Link with Leaves/Paths** check box.

This check box allows you to specify on which port you want to deploy the EPG.

d) Click **Next**.

e) From the **Path** drop-down list, choose the static path to the destination EPG.

Step 6 In the **Create Application EPG STEP 2 > Leaves/Paths** dialog box, from the **Physical Domain** drop-down list, choose a physical domain.

Step 7 Complete one of the following sets of steps:

Option	Description
If you want to deploy the EPG on...	Then
A node	**a.** Expand the **Leaves** area. **b.** From the **Node** drop-down list, choose a node. **c.** In the **Encap** field, enter the appropriate VLAN. **d.** (Optional) From the **Deployment Immediacy** drop-down list, accept the default **On Demand** or choose **Immediate**.

Option	Description
	e. (Optional) From the Mode drop-down list, accept the default **Trunk** or choose another mode.
A port on the node	**a.** Expand the **Paths** area. **b.** From the **Path** drop-down list, choose the appropriate node and port. **c.** (Optional) In the **Deployment Immediacy** field drop-down list, accept the default **On Demand** or choose **Immediate**. **d.** (Optional) From the Mode drop-down list, accept the default **Trunk** or choose another mode. **e.** In the **Port Encap** field, enter the secondary VLAN to be deployed. **f.** (Optional) In the **Primary Encap** field, enter the primary VLAN to be deployed.

Step 8 Click **Update** and click **Finish**.

Step 9 In the left navigation pane, expand the EPG that you created.

Step 10 Complete one of the following actions:

- If you created the EPG on a node, click **Static Leafs**, and in the work pane view details of the static binding paths.
- If you created the EPG on a port of the node, click **Static Ports**, and in the work pane view details of the static binding paths.

Deploying an EPG on a Specific Port with APIC Using the NX-OS Style CLI

Procedure

Step 1 Configure a VLAN domain:

Example:

```
apic1(config)# vlan-domain dom1
apic1(config-vlan)# vlan 10-100
```

Step 2 Create a tenant:

Example:

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

Step 3 Create a private network/VRF:

Example:

```
apic1(config-tenant)# vrf context ctx1
apic1(config-tenant-vrf)# exit
```

Step 4 Create a bridge domain:

Example:

```
apic1(config-tenant)# bridge-domain bd1
apic1(config-tenant-bd)# vrf member ctx1
apic1(config-tenant-bd)# exit
```

Step 5 Create an application profile and an application EPG:

Example:

```
apic1(config-tenant)# application AP1
apic1(config-tenant-app)# epg EPG1
apic1(config-tenant-app-epg)# bridge-domain member bd1
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)# exit
apic1(config-tenant)# exit
```

Step 6 Associate the EPG with a specific port:

Example:

```
apic1(config)# leaf 1017
apic1(config-leaf)# interface ethernet 1/13
apic1(config-leaf-if)# vlan-domain member dom1
apic1(config-leaf-if)# switchport trunk allowed vlan 20 tenant t1 application AP1 epg EPG1
```

Note The vlan-domain and vlan-domain member commands mentioned in the above example are a pre-requisite for deploying an EPG on a port.

Deploying an EPG on a Specific Port with APIC Using the REST API

Before you begin

The tenant where you deploy the EPG is created.

Procedure

Deploy an EPG on a specific port.

Example:

```
<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>
```

Creating Domains, Attach Entity Profiles, and VLANs to Deploy an EPG on a Specific Port

This topic provides a typical example of how to create physical domains, Attach Entity Profiles (AEP), and VLANs that are mandatory to deploy an EPG on a specific port.

All endpoint groups (EPGs) require a domain. Interface policy groups must also be associated with Attach Entity Profile (AEP), and the AEP must be associated with a domain, if the AEP and EPG have to be in same domain. Based on the association of EPGs to domains and of interface policy groups to domains, the ports and VLANs that the EPG uses are validated. The following domain types associate with EPGs:

- Application EPGs
- Layer 3 external outside network instance EPGs
- Layer 2 external outside network instance EPGs
- Management EPGs for out-of-band and in-band access

The APIC checks if an EPG is associated with one or more of these types of domains. If the EPG is not associated, the system accepts the configuration but raises a fault. The deployed configuration may not function properly if the domain association is not valid. For example, if the VLAN encapsulation is not valid for use with the EPG, the deployed configuration may not function properly.

Note EPG association with the AEP without static binding does not work in a scenario when you configure the EPG as **Trunk** under the AEP with one end point under the same EPG supporting Tagging and the other end point in the same EPG does not support VLAN tagging. While associating AEP under the EPG, you can configure it as Trunk, Access (Tagged) or Access (Untagged).

Creating Domains, and VLANS to Deploy an EPG on a Specific Port Using the GUI

Before you begin

- The tenant where you deploy the EPG is already created.
- An EPG is statically deployed on a specific port.

Procedure

Step 1 On the menu bar, click **Fabric** > **Access Policies**.

Step 2 In the **Navigation** pane, click **Quick Start**.

Step 3 In the **Work** pane, click **Configure an Interface, PC, and vPC**.

Step 4 In the **Configure an Interface, PC, and vPC** dialog box, click the + icon to select switches and perform the following actions:

a) From the **Switches** drop-down list, check the check box for the desired switch.
b) In the **Switch Profile Name** field, a switch name is automatically populated.

Note Optionally, you can enter a modified name.

c) Click the + icon to configure the switch interfaces.
d) In the **Interface Type** field, click the **Individual** radio button.
e) In the **Interfaces** field, enter the range of desired interfaces.
f) In the **Interface Selector Name** field, an interface name is automatically populated.

Note Optionally, you can enter a modified name.

g) In the **Interface Policy Group** field, choose the **Create One** radio button.
h) From the **Link Level Policy** drop-down list, choose the appropriate link level policy.

Note Create additional policies as desired, otherwise the default policy settings are available.

i) From the **Attached Device Type** field, choose the appropriate device type.
j) In the **Domain** field, click the **Create One** radio button.
k) In the **Domain Name** field, enter a domain name.
l) In the **VLAN** field, click the **Create One** radio button.
m) In the **VLAN Range** field, enter the desired VLAN range. Click **Save**, and click **Save** again.
n) Click **Submit**.

Step 5 On the menu bar, click **Tenants**. In the **Navigation** pane, expand the appropriate ***Tenant_name*** > **Application Profiles** > **Application EPGs** > ***EPG_name*** and perform the following actions:

a) Right-click **Domains (VMs and Bare-Metals)**, and click **Add Physical Domain Association**.
b) In the **Add Physical Domain Association** dialog box, from the **Physical Domain Profile** drop-down list, choose the appropriate domain.
c) Click **Submit**.
The AEP is associated with a specific port on a node and with a domain. The physical domain is associated with the VLAN pool and the Tenant is associated with this physical domain.

The switch profile and the interface profile are created. The policy group is created in the port block under the interface profile. The AEP is automatically created, and it is associated with the port block and with the domain. The domain is associated with the VLAN pool and the Tenant is associated with the domain.

Creating AEP, Domains, and VLANs to Deploy an EPG on a Specific Port Using the NX-OS Style CLI

Before you begin

- The tenant where you deploy the EPG is already created.
- An EPG is statically deployed on a specific port.

Procedure

Step 1 Create a VLAN domain and assign VLAN ranges:

Example:

```
apic1(config)# vlan-domain domP
apic1(config-vlan)# vlan 10
apic1(config-vlan)# vlan 25
apic1(config-vlan)# vlan 50-60
apic1(config-vlan)# exit
```

Step 2 Create an interface policy group and assign a VLAN domain to the policy group:

Example:

```
apic1(config)# template policy-group PortGroup
apic1(config-pol-grp-if)# vlan-domain member domP
```

Step 3 Create a leaf interface profile, assign an interface policy group to the profile, and assign the interface IDs on which the profile will be applied:

Example:

```
apic1(config)# leaf-interface-profile InterfaceProfile1
apic1(config-leaf-if-profile)# leaf-interface-group range
apic1(config-leaf-if-group)# policy-group PortGroup
apic1(config-leaf-if-group)# interface ethernet 1/11-13
apic1(config-leaf-if-profile)# exit
```

Step 4 Create a leaf profile, assign the leaf interface profile to the leaf profile, and assign the leaf IDs on which the profile will be applied:

Example:

```
apic1(config)# leaf-profile SwitchProfile-1019
apic1(config-leaf-profile)# leaf-interface-profile InterfaceProfile1
apic1(config-leaf-profile)# leaf-group range
apic1(config-leaf-group)# leaf 1019
apic1(config-leaf-group)#
```

Creating AEP, Domains, and VLANs to Deploy an EPG on a Specific Port Using the REST API

Before you begin

- The tenant where you deploy the EPG is already created.
- An EPG is statically deployed on a specific port.

Procedure

Step 1 Create the interface profile, switch profile and the Attach Entity Profile (AEP).

Example:

```
   <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>
```

Step 2 Create a domain.

Example:

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

Step 3 Create a VLAN range.

Example:

```
<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>
```

Step 4 Associate the EPG with the domain.

Example:

```
<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>
```

Deploying an Application EPG through an AEP or Interface Policy Group to Multiple Ports

Through the APIC Advanced GUI and REST API, you can associate attached entity profiles directly with application EPGs. By doing so, you deploy the associated application EPGs to all those ports associated with the attached entity profile in a single configuration.

Through the APIC REST API or the NX-OS style CLI, you can deploy an application EPG to multiple ports through an Interface Policy Group.

Deploying an EPG through an AEP to Multiple Interfaces Using the APIC GUI

You can quickly associate an application with an attached entity profile to quickly deploy that EPG over all the ports associated with that attached entity profile.

Before you begin

- The target application EPG is created.
- The VLAN pools has been created containing the range of VLANs you wish to use for EPG Deployment on the AEP.
- The physical domain has been created and linked to the VLAN Pool and AEP.
- The target attached entity profile is created and is associated with the ports on which you want to deploy the application EPG.

Procedure

Step 1 Navigate to the target attached entity profile.

a) Open the page for the attached entity profile to use. In the GUI, click **Fabric** > **Access Policies** > **Policies** > **Global** > **Attachable Access Entity Profiles**.

b) Click the target attached entity profile to open its Attachable Access Entity Profile window.

Step 2 Click the **Show Usage** button to view the leaf switches and interfaces associated with this attached entity profile.

the application EPGs associated with this attached entity profile are deployed to all the ports on all the switches associated with this attached entity profile.

Step 3 Use the **Application EPGs** table to associate the target application EPG with this attached entity profile. Click + to add an application EPG entry. Each entry contains the following fields:

Field	Action
Application EPGs	Use the drop down to choose the associated Tenant, Application Profile, and target application EPG.
Encap	Enter the name of the VLAN over which the target application EPG will communicate.
Primary Encap	If the application EPG requires a primary VLAN, enter the name of the primary VLAN.

Field	Action
Mode	Use the drop down to specify the mode in which data is transmitted: • **Trunk** -- Choose if traffic from the host is tagged with a VLAN ID. • **Access** -- Choose if traffic from the host is tagged with an 802.1p tag. • **Access Untagged** -- Choose if the traffic from the host is untagged.

Step 4 Click **Submit**.
the application EPGs associated with this attached entity profile are deployed to all the ports on all the switches associated with this attached entity profile.

Deploying an EPG through an Interface Policy Group to Multiple Interfaces Using the NX-OS Style CLI

In the NX-OS CLI, an attached entity profile is not explicitly defined to associate with an EPG for rapid deployment; instead the interface policy group is defined, assigned a domain, applied to all the ports associated with a VLAN and configured to include the application EPG to be deployed over that VLAN.

Before you begin

- The target application EPG is created.
- The VLAN pools has been created containing the range of VLANs you wish to use for EPG Deployment on the AEP.
- The physical domain has been created and linked to the VLAN Pool and AEP.
- The target attached entity profile is created and is associated with the ports on which you want to deploy the application EPG.

Procedure

Step 1 Associate the target EPG with the interface policy group.

The sample command sequence specifies an interface policy group **pg3** associated with VLAN domain, **domain1**, and with VLAN **1261**. The application EPG, **epg47** is deployed to all interfaces associated with this policy group.

Example:

```
 apic1# configure terminal
apic1(config)# template policy-group pg3
apic1(config-pol-grp-if)# vlan-domain member domain1
apic1(config-pol-grp-if)# switchport trunk allowed vlan 1261 tenant tn10 application pod1-AP

      epg epg47
```

Step 2 Check the target ports to ensure deployment of the policies of the interface policy group associated with application EPG.

The output of the sample **show** command sequence indicates that policy group **pg3** is deployed on Ethernet port **1/20** on leaf switch **1017**.

Example:

```
 apic1# show run leaf 1017 int eth 1/20
# Command: show running-config leaf 1017 int eth 1/20
# Time: Mon Jun 27 22:12:10 2016
  leaf 1017
    interface ethernet 1/20
      policy-group pg3
      exit
    exit
ifav28-ifc1#
```

Deploying an EPG through an AEP to Multiple Interfaces Using the REST API

The interface selectors in the AEP enable you to configure multiple paths for an AEPg. The following can be selected:

1. A node or a group of nodes
2. An interface or a group of interfaces

   The interfaces consume an interface policy group (and so an `infra:AttEntityP`).
3. The infra:AttEntityP is associated to the AEPg, thus specifying the VLANs to use.

   An infra:AttEntityP can be associated with multiple AEPgs with different VLANs.

When you associate the infra:AttEntityP with the AEPg, as in 3, this deploys the AEPg on the nodes selected in 1, on the interfaces in 2, with the VLAN provided by 3.

In this example, the AEPg `uni/tn-Coke/ap-AP/epg-EPG1` is deployed on interfaces 1/10, 1/11, and 1/12 of nodes 101 and 102, with vlan-102.

Before you begin

- Create the target application EPG (`AEPg`).
- Create the VLAN pool containing the range of VLANs you wish to use for EPG deployment with the Attached Entity Profile (AEP).
- Create the physical domain and link it to the VLAN pool and AEP.

Procedure

To deploy an AEPg on selected nodes and interfaces, send a post with XML such as the following:

Example:

```
<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>
```

Microsegmented EPGs

Using Microsegmentation with Network-based Attributes on Bare Metal

You can use Cisco APIC to configure Microsegmentation with Cisco ACI to create a new, attribute-based EPG using a network-based attribute, a MAC address or one or more IP addresses. You can configure Microsegmentation with Cisco ACI using network-based attributes to isolate VMs or physical endpoints within a single base EPG or VMs or physical endpoints in different EPGs.

Using an IP-based Attribute

You can use an IP-based filter to isolate a single IP address, a subnet, or multiple of noncontiguous IP addresses in a single microsegment. You might want to isolate physical endpoints based on IP addresses as a quick and simply way to create a security zone, similar to using a firewall.

Using a MAC-based Attribute

You can use a MAC-based filter to isolate a single MAC address or multiple MAC addresses. You might want to do this if you have a server sending bad traffic int he network. By creating a microsegment with a MAC-based filter, you can isolate the server.

Configuring Network-based Microsegmented EPGs in a Bare-Metal environment Using the GUI

You can use Cisco APIC to configure microsegmentation to put physical endpoint devices that belong to different base EPGs or the same EPG into a new attribute-based EPG.

Procedure

Step 1 Log into the Cisco APIC.

Step 2 Choose **Tenants** and then choose the tenant within which you want to create a microsegment.

Step 3 In the tenant navigation pane, expand the tenant folder, the **Application Profiles** folder, the ***profile*** folder, and the **Application EPGs** folder.

Step 4 Take one of the following actions:

- If you want to put physical endpoint devices from the same base EPG into a new, attribute-based EPG, click the base EPG containing the physical endpoint devices.
- If you want to put physical endpoint devices from different base EPGs into a new, attribute-based EPG, click one of the base EPG containing the physical endpoint devices.

The properties for the base EPG appear in the work pane.

Step 5 In the work pane, click the **Operational** tab at the top right of the screen.

Step 6 Below the **Operational** tab, ensure that the **Client End-Points** tab is active.
The work pane displays all the physical endpoints that belong to the base EPG.

Step 7 Note the IP address or MAC address for the endpoint device or endpoint devices that you want to put into a new microsegment.

Step 8 If you want to put endpoint devices from different base EPGs into a new attribute-based EPG, repeat Step 4 through Step 7 for each of the base EPGs.

Step 9 In the tenant navigation pane, right-click the **uSeg EPGs** folder, and then choose **Create uSeg EPG**.

Step 10 Complete the following series of steps to begin creation of an attribute-based EPG for one of the groups of endpoint devices:

a) In the **Create uSeg EPG** dialog box, in the **Name** field, enter a name.

We recommend that you choose a name that indicates that the new attribute-based EPG is a microsegment.

b) In the intra-EPG isolation field, select **enforced** or **unenforced**.

If you select **enforced**, ACI prevents all communication between the endpoint devices within this uSeg EPG.

c) In the **Bridge Domain** area, choose a bridge domain from the drop-down list.

d) In the **uSeg Attributes** area, choose **IP Address Filter** or **MAC Address Filter** from the **+** drop-down list on the right side of the dialog box.

Step 11 Complete one of the following series of steps to configure the filter.

If you want to use...	Then...
An IP-based attribute	**a.** In the **Create IP Attribute** dialog box, in the **Name** field, enter a name. We recommend that you choose a name that reflects the filter's function. **b.** In the **IP Address** field, enter an IP address or a subnet with the appropriate subnet mask. **c.** Click **OK**. **d.** (Optional) Create a second IP Address filter by repeating Step 10 c through Step 11 c.

If you want to use...	Then...
	You might want to do this to include discontinuous IP addresses in the microsegment. **e.** In the **Create uSeg EPG** dialog box, click **Submit**.
A MAC-based attribute	**a.** In the **Create MAC Attribute** dialog box, in the **Name** field, enter a name. We recommend that you choose a name that reflects the filter's function. **b.** In the **MAC Address** field, enter a MAC address. **c.** Click **OK**. **d.** In the **Create uSeg EPG** dialog box, click **Submit**.

Step 12 Complete the following steps to associate the uSeg EPG with a physical domain.

a) In the navigation pane, ensure that the uSeg EPG folder is open and then open the container for the microsegment that you just created.
b) Click the folder **Domains (VMs and Bare-Metals)**.
c) On the right side of the work pane, click **Actions** and then choose **Add Physical Domain Association** from the drop-down list.
d) In the **Add Physical Domain Association** dialog box, choose a profile from the **Physical Domain Profile** drop-down list.
e) In the **Deploy Immediacy** area, accept the default **On Demand**.
f) In the **Resolution Immediacy** area, accept the default **Immediate**.
g) Click **Submit**.

Step 13 Associate the uSeg EPG with the appropriate leaf switch.

a) In the navigation pane, ensure the uSeg EPG folder is open then click **Static Leafs**.
b) In the Static Leafs window, click **Actions** > **Statically Link with Node**
c) In the Statically Link With Node dialog, select the leaf node and mode.
d) Click **Submit**.

Step 14 Repeat Step 9 through Step 13 for any other network attribute-based EPGs that you want to create.

What to do next

Verify that the attribute-based EPG was created correctly.

If you configured an IP-based or MAC-based attribute, make sure that traffic is running on the end pont devices that you put into the new microsegments.

Configuring a Network-Based Microsegmented EPG in a Bare-Metal Environment Using the NX-OS Style CLI

This section describes how to configure microsegmentation with Cisco ACI using network-based attributes (IP address or MAC address) within a base EPG in a bare-metal environment.

Procedure

	Command or Action	Purpose

Step 1	In the CLI, enter configuration mode:
	Example:
	`apic1# configure`
	`apic1(config)#`	

Step 2	Create the microsegment:
	Example:
	This example uses a filter based on an IP address.
	`apic1(config)# tenant cli-ten1`
	`apic1(config-tenant)# application cli-a1`
	`apic1(config-tenant-app)# epg cli-uepg1 type micro-segmented`
	`apic1(config-tenant-app-uepg)# bridge-domain member cli-bd1`
	`apic1(config-tenant-app-uepg)# attribute cli-upg-att match ip <X.X.X.X>`
	`#Schemes to express the ip`
	`A.B.C.D IP Address`
	`A.B.C.D/LEN IP Address and mask`
	Example:
	This example uses a filter based on a MAC address.
	`apic1(config)# tenant cli-ten1`
	`apic1(config-tenant)# application cli-a1`
	`apic1(config-tenant-app)# epg cli-uepg1 type micro-segmented`
	`apic1(config-tenant-app-uepg)# bridge-domain member cli-bd1`
	`apic1(config-tenant-app-uepg)# attribute cli-upg-att match mac`
	`<FF-FF-FF-FF-FF-FF>`
	`#Schemes to express the mac`
	`E.E.E MAC address (Option 1)`
	`EE-EE-EE-EE-EE-EE MAC address (Option 2)`
	`EE:EE:EE:EE:EE:EE MAC address (Option 3)`
	`EEEE.EEEE.EEEE MAC address (Option 4)`
	Example:
	This example uses a filter based on a MAC address and enforces intra-EPG isolation between all members of this uSeg EPG:
	`apic1(config)# tenant cli-ten1`
	`apic1(config-tenant)# application cli-a1`
	`apic1(config-tenant-app)# epg cli-uepg1 type micro-segmented`
	`apic1(config-tenant-app-uepg)# isolation enforced`
	`apic1(config-tenant-app-uepg)# bridge-domain member cli-bd1`
	`apic1(config-tenant-app-uepg)# attribute`	

	Command or Action	Purpose
	cli-upg-att match mac <FF-FF-FF-FF-FF-FF> #Schemes to express the mac E.E.E MAC address (Option 1) EE-EE-EE-EE-EE-EE MAC address (Option 2) EE:EE:EE:EE:EE:EE MAC address (Option 3) EEEE.EEEE.EEEE MAC address (Option 4)	
Step 3	Deploy the EPG. **Example:** This example deploys the EPG and bids to the leaf. apic1(config)# leaf 101 apic1(config-leaf)# deploy-epg tenant cli-ten1 application cli-a1 epg cli-uepg1 type micro-segmented	
Step 4	Verify the microsegment creation: **Example:** apic1(config-tenant-app-uepg)# show running-config # Command: show running-config tenant cli-ten1 application cli-app1 epg cli-uepg1 type micro-segmented # Time: Thu Oct 8 11:54:32 2015 tenant cli-ten1 application cli-app1 epg cli-esx1bu type micro-segmented bridge-domain cli-bd1 attribute cli-uepg-att match mac 00:11:22:33:44:55 exit exit exit	

Configuring a Network-Based Microsegmented EPG in a Bare-Metal Environment Using the REST API

This section describes how to configure network attribute microsegmentation with Cisco ACI in a bare-metal environment using the REST API.

Procedure

Step 1 Log in to the Cisco APIC.

Step 2 Post the policy to https://*apic-ip-address*/api/node/mo/.xml.

Example:

A: The following example configures a microsegment named 41-subnet using an IP-based attribute.

```
<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>
```

Example:

This example is for base EPG for Example A: .

```
<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>
```

Example:

B: The following example configures a microsegment named useg-epg using a MAC-based attribute.

```
<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>
```

IP Address-Based Microsegmented EPG as a Shared Resource

You can configure an IP address-based microsegemented EPG as a resource that can be accessed from both within and without the VRF on which it is located. The method of doing so is to configure an existing IP address-based microsegmented EPG with a subnet (assigned a unicast IP address) and enable that subnet for being advertised and shared by devices located on VRFs other than the one on which this EPG is native. Then you define an IP attribute with an option enabled that associates the EPG with the IP address of the shared subnet.

Configuring an IP-based Microsegmented EPG as a Shared Resource Using the GUI

You can configure a microsegmented EPG with an IP-Address with 32 bit mask as a shared service, accessible by clients outside of the VRF and the current fabric.

Before you begin

The following GUI description of configuring assumes the preconfiguration of an IP address-based microsegmented EPG configured whose subnet mask is /32.

Note

- For directions on configuring an IP address based EPG in a physical environment, see Using Microsegmentation with Network-based Attributes on Bare Metal, on page 25
- For directions on configuring an IP address based EPG in a virtual environment, see *Configuring Microsegmentation with Cisco ACI* in the *Cisco ACI Virtualization Guide*.

Procedure

Step 1 Navigate to the target IP-address-based EPG.

a) In the APIC GUI, click **Tenant** > **tenant_name** > **uSeg EPGs** > **uSeg_epg_name** to display the EPG's **Properties** dialog.

Step 2 For the target EPG, configure an IP attribute to match the EPG subnet address.

a) In the **Properties** dialog, locate the **uSeg Attributes** table, and click **+**. When prompted, choose **IP Address Filter** to display the **Create IP Attribute** dialog.

b) Enter a name in the Name field

c) Check the box for **Use FV Subnet**.

Enabling this option, indicates that the IP attribute value matches the IP address of a shared subnet.

d) Click **Submit**.

Step 3 Create a shared subnet for the target EPG.

a) With the folder for the target IP address-based uSeg EPG still open in the APIC navigation pane, right-click the **Subnets** folder and select **Create EPG Subnets**.

b) In the **Default Gateway** field, enter the IP address/mask of the IP address-based microsegmented EPG.

Note

- In all cases the subnet mask must be /32.
- In the context of an IP address-based EPG, you are not actually entering the default address for a gateway, rather you are entering the IP address for the shared EPG subnet.

c) Select **Treat as a virtual IP address**.

d) Under Scope select **Advertised Externally** and **Shared between VRFs**.

e) Click **Submit**.

Configuring an IP-based Microsegmented EPG as a Shared Resource Using the NX-OS CLI

Before you begin

The following GUI description of configuring assumes the preconfiguration of an IP address-based microsegmented EPG configured whose subnet mask is /32.

Procedure

	Command or Action	Purpose

Step 1	Enable the IP address microsegmented EPG for shared service by associating the EPG with the IP address of its subnet.
	Example:
	`apic-1(config)# tenant t0`
	`apic-1(config-tenant-app)# epg cli-epg type micro-segmented`
	`apic-1(config-tenant-app-uepg)# bridge-domain member b0`
	`apic-1(config-tenant-app-uepg)# attribute ip match ip-use-epg-subnet`
	
	`apic-1(config-tenant-app-uepg)# show run`
	`# Command: show running-config tenant t0 application a0 epg cli-epg type micro-segmented`
	`# Time: Thu Sep 22 00:17:07 2016`
	`tenant t0`
	`application a0`
	`epg cli-epg type micro-segmented`
	`bridge-domain member b0`
	`attribute ip match ip-use-epg-subnet`
	`exit`
	`exit`
	`Exit`
		In this example, microsegmented EPG, cli-epg, is configured with the **ip-use-epg-subnet** option (useFvSubnet), thus associating the EPG with the IP address of its subnet. APIC then advertises that subnet address, thus making the EPG accessible as a service to devices on VRFs other than the one on which the EPG is native.

Step 2	Deploy the EPG to a leaf.	In this example, microsegmented EPG, cli-epg, is deployed to leaf 102.
		`apic-1(config)# leaf 102`
		`apic-1(config-leaf)# deploy-epg tenant t0 application a0 epg cli-epg type micro-segmented`
		
		`apic-1(config-leaf)# show run`
		`# Command: show running-config leaf 102`
		`# Time: Thu Sep 22 00:18:46 2016`
		`leaf 102`
		`deploy-epg tenant t0 application a0 epg cli-epg type micro-segmented`

Configuring an IP-based Microsegmented EPG as a Shared Resource Using the REST API

You can configure a microsegmented EPG with an IP-Address with 32 bit mask as a shared service, accessible by clients outside of the VRF and the current fabric.

Procedure

To configure an IP address-attribute microsegmented EPG `epg3` with a shared subnet, with an IP address and 32-bit mask, send a post with XML such as the following example. In the IP attributes, the attribute **usefvSubnet** is set to "yes."

Example:

```
<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>
```

Unconfiguring an IP-based Microsegmented EPG as a Shared Resource Using the GUI

When you unconfigure an IP address-Based microsegmented EPG as a shared service, you must remove the shared subnet and also disable the option to use that subnet as a shared resource.

Before you begin

Before you unconfigure an IP address-based microsegmented EPG as a shared service, you should know the following:

- Know which subnet is configured as a shared service address for the IP address-based microsegmented EPG.
- Know which IP attribute is configured with the **Use FV Subnet** option enabled.

Procedure

Step 1 Remove subnet from the IP addressed-based microsegmented EPG.

a) In the APIC GUI, click **Tenant** > **tenant_name** > **Application Profiles** > **epg_name** > **uSeg EPGs** > **uSeg EPGs** > **uSeg_epg_name**.

b) With the folder for the target IP address-based uSeg EPG still open in the APIC navigation pane, click the **Subnets** folder.

c) In the **Subnets** window, select the subnet that is advertised and shared with other VRFs and click **Actions** > **Delete**. then
d) Click **Yes** to confirm the deletion.

Step 2 Disable the **Use FV Subnet** option.

a) With the folder for the target IP address-based uSeg EPG still open in the APIC navigation pane, click the name of the micro-segmented EPG to display the to display the EPG's **Properties** dialog.
b) In the **Properties** dialog, locate the **uSeg Attributes** table, and locate the IP attribute item with the **Use FV Subnet** option enabled.
c) Double-click that item to display the **Edit IP Attribute** dialog.
d) In the **`Edit IP Attribute`** dialog, deselect the **Use FV Subnet** option.
e) Assign another IP address attribute in the IP Address field.

Note This address must be a unicast address with a 32 bit mask (for example: 124.124.124.123/32).

f) Click **Submit**.

Unconfiguring an IP-based Microsegmented EPG as a Shared Resource Using the NX-OS Style CLI

To unconfigure an IP address-based microsegmented EPG as a shared service, disable the ip-use-epg-subnet option for that EPG.

Before you begin

Procedure

	Command or Action	Purpose

Step 1	Disable the ip-use-epg-subnet option.
	Example:
	`apic-1(config)# tenant t0`
	`apic-1(config-tenant-app)# epg cli-epg type micro-segmented`
	`apic-1(config-tenant-app-uepg)# no attribute ip match ip-use-epg-subnet`
	`apic-1(config-tenant-app-uepg)# exit`
	`apic-1(config-tenant-app)# exit`
		The example code disables the ip-use-epg-subnet option for the microsegmented EPG cli-epg.

Unconfiguring an IP-based Microsegmented EPG as a Shared Resource Using the REST API

You can disable an IP address-based microsegmented EPG by setting the **usefvSubnet** property to "no."

Procedure

In the API structure for the microsegmented EPG currently configured as a shared service, change the value of the usefvSubnet property from "yes" to "no."

In the example, the IP address-based microsegmented EPG, epg3, is disabled as a shared service.

Example:

```
<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="no"/>

   </fvCrtrn>
   <fvRsProv matchT="AtleastOne" prio="unspecified" tnVzBrCPName="ip-epg"/>
   <fvRsProv matchT="AtleastOne" prio="unspecified" tnVzBrCPName="shared-svc"/>
</fvAEPg>
```

Deploying Application Profiles and Contracts

Security Policy Enforcement

As traffic enters the leaf switch from the front panel interfaces, the packets are marked with the EPG of the source EPG. The leaf switch then performs a forwarding lookup on the packet destination IP address within the tenant space. A hit can result in any of the following scenarios:

1. A unicast (/32) hit provides the EPG of the destination endpoint and either the local interface or the remote leaf switch VTEP IP address where the destination endpoint is present.
2. A unicast hit of a subnet prefix (not /32) provides the EPG of the destination subnet prefix and either the local interface or the remote leaf switch VTEP IP address where the destination subnet prefix is present.
3. A multicast hit provides the local interfaces of local receivers and the outer destination IP address to use in the VXLAN encapsulation across the fabric and the EPG of the multicast group.

Note Multicast and external router subnets always result in a hit on the ingress leaf switch. Security policy enforcement occurs as soon as the destination EPG is known by the ingress leaf switch.

A miss result in the forwarding table causes the packet to be sent to the forwarding proxy in the spine switch. The forwarding proxy then performs a forwarding table lookup. If it is a miss, the packet is dropped. If it is a hit, the packet is sent to the egress leaf switch that contains the destination endpoint. Because the egress leaf switch knows the EPG of the destination, it performs the security policy enforcement. The egress leaf switch must also know the EPG of the packet source. The fabric header enables this process because it carries the EPG from the ingress leaf switch to the egress leaf switch. The spine switch preserves the original EPG in the packet when it performs the forwarding proxy function.

On the egress leaf switch, the source IP address, source VTEP, and source EPG information are stored in the local forwarding table through learning. Because most flows are bidirectional, a return packet populates the forwarding table on both sides of the flow, which enables the traffic to be ingress filtered in both directions.

Contracts Contain Security Policy Specifications

In the ACI security model, contracts contain the policies that govern the communication between EPGs. The contract specifies what can be communicated and the EPGs specify the source and destination of the communications. Contracts link EPGs, as shown below.

EPG 1 ---------- CONTRACT ---------- EPG 2

Endpoints in EPG 1 can communicate with endpoints in EPG 2 and vice versa if the contract allows it. This policy construct is very flexible. There can be many contracts between EPG 1 and EPG 2, there can be more than two EPGs that use a contract, and contracts can be reused across multiple sets of EPGs, and more.

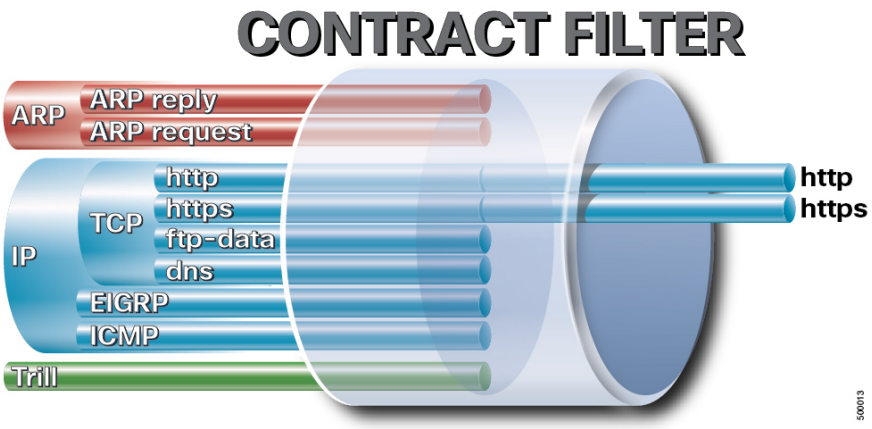
There is also directionality in the relationship between EPGs and contracts. EPGs can either provide or consume a contract. An EPG that provides a contract is typically a set of endpoints that provide a service to a set of client devices. The protocols used by that service are defined in the contract. An EPG that consumes a contract is typically a set of endpoints that are clients of that service. When the client endpoint (consumer) tries to connect to a server endpoint (provider), the contract checks to see if that connection is allowed. Unless otherwise specified, that contract would not allow a server to initiate a connection to a client. However, another contract between the EPGs could easily allow a connection in that direction.

This providing/consuming relationship is typically shown graphically with arrows between the EPGs and the contract. Note the direction of the arrows shown below.

EPG 1 <-------consumes-------- CONTRACT <-------provides-------- EPG 2

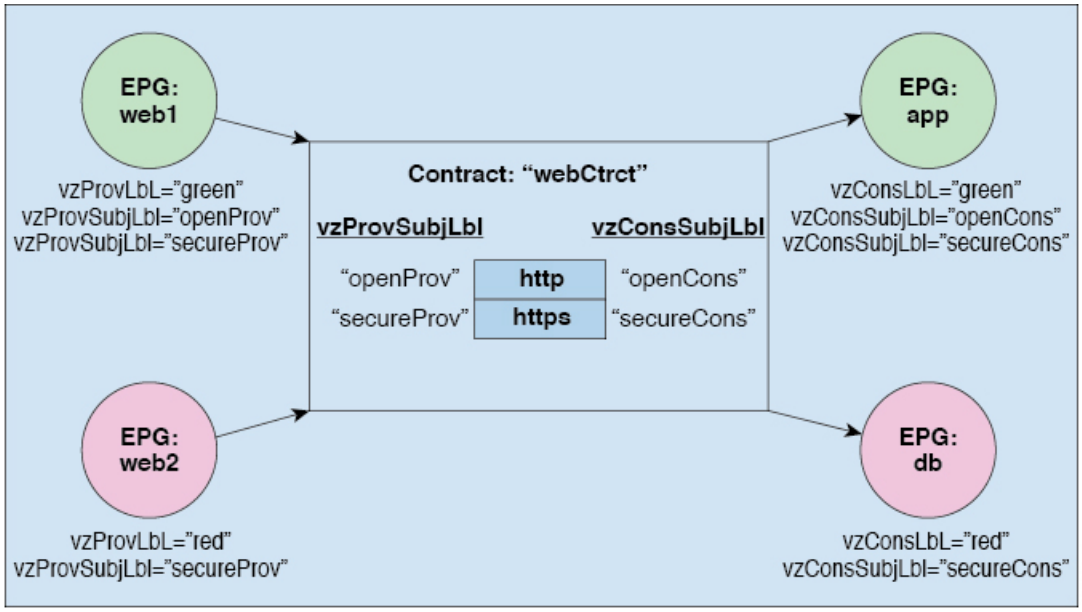
The contract is constructed in a hierarchical manner. It consists of one or more subjects, each subject contains one or more filters, and each filter can define one or more protocols.

Figure 6: Contract Filters



The following figure shows how contracts govern EPG communications.

Figure 7: Contracts Determine EPG to EPG Communications



For example, you may define a filter called HTTP that specifies TCP port 80 and port 8080 and another filter called HTTPS that specifies TCP port 443. You might then create a contract called webCtrct that has two sets of subjects. openProv and openCons are the subjects that contain the HTTP filter. secureProv and secureCons are the subjects that contain the HTTPS filter. This webCtrct contract can be used to allow both secure and non-secure web traffic between EPGs that provide the web service and EPGs that contain endpoints that want to consume that service.

These same constructs also apply for policies that govern virtual machine hypervisors. When an EPG is placed in a virtual machine manager (VMM) domain, the APIC downloads all of the policies that are associated with the EPG to the leaf switches with interfaces connecting to the VMM domain. For a full explanation of VMM domains, see the *Virtual Machine Manager Domains* chapter of *Application Centric Infrastructure Fundamentals*. When this policy is created, the APIC pushes it (pre-populates it) to a VMM domain that specifies which switches allow connectivity for the endpoints in the EPGs. The VMM domain defines the set of switches and ports that allow endpoints in an EPG to connect to. When an endpoint comes on-line, it is associated with the appropriate EPGs. When it sends a packet, the source EPG and destination EPG are derived from the packet and the policy defined by the corresponding contract is checked to see if the packet is allowed. If yes, the packet is forwarded. If no, the packet is dropped.

Contracts consist of 1 or more subjects. Each subject contains 1 or more filters. Each filter contains 1 or more entries. Each entry is equivalent to a line in an Access Control List (ACL) that is applied on the Leaf switch to which the endpoint within the endpoint group is attached.

In detail, contracts are comprised of the following items:

- Name—All contracts that are consumed by a tenant must have different names (including contracts created under the common tenant or the tenant itself).
- Subjects—A group of filters for a specific application or service.
- Filters—Used to classify traffic based upon layer 2 to layer 4 attributes (such as Ethernet type, protocol type, TCP flags and ports).
- Actions—Action to be taken on the filtered traffic. The following actions are supported:

- Permit the traffic (regular contracts, only)
- Mark the traffic (DSCP/CoS) (regular contracts, only)
- Redirect the traffic (regular contracts, only, through a service graph)
- Copy the traffic (regular contracts, only, through a service graph or SPAN)
- Block the traffic (taboo contracts)

  With Cisco APIC Release 3.2(x) and switches with names that end in EX or FX, you can alternatively use a subject Deny action or Contract or Subject Exception in a standard contract to block traffic with specified patterns.
- Log the traffic (taboo contracts and regular contracts)

- Aliases—(Optional) A changeable name for an object. Although the name of an object, once created, cannot be changed, the Alias is a property that can be changed.

Thus, the contract allows more complex actions than just allow or deny. The contract can specify that traffic that matches a given subject can be re-directed to a service, can be copied, or can have its QoS level modified. With pre-population of the access policy in the concrete model, endpoints can move, new ones can come on-line, and communication can occur even if the APIC is off-line or otherwise inaccessible. The APIC is removed from being a single point of failure for the network. Upon packet ingress to the ACI fabric, security policies are enforced by the concrete model running in the switch.

Three-Tier Application Deployment

A filter specifies the data protocols to be allowed or denied by a contract that contains the filter. A contract can contain multiple subjects. A subject can be used to realize uni- or bidirectional filters. A unidirectional filter is a filter that is used in one direction, either from consumer-to-provider (IN) or from provider-to-consumer (OUT) filter. A bidirectional filter is the same filter that is used in both directions. It is not reflexive.

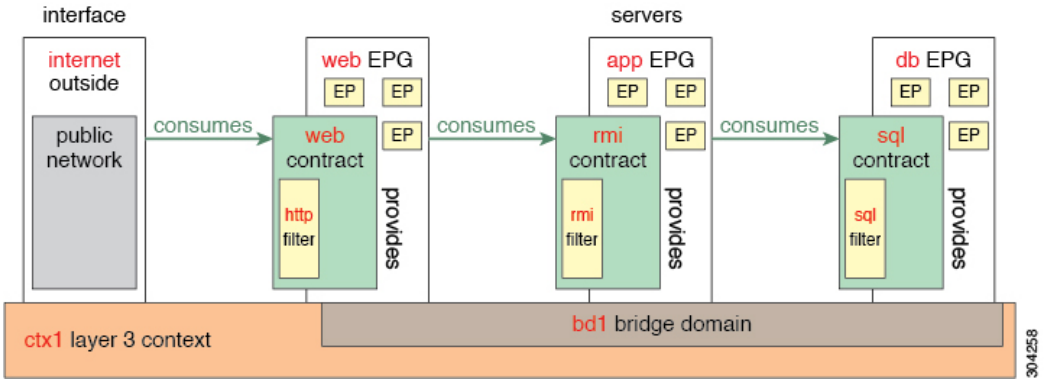
Contracts are policies that enable inter-End Point Group (inter-EPG) communication. These policies are the rules that specify communication between application tiers. If no contract is attached to the EPG, inter-EPG communication is disabled by default. No contract is required for intra-EPG communication because intra-EPG communication is always allowed.

Application profiles enable you to model application requirements that the APIC then automatically renders in the network and data center infrastructure. The application profiles enable administrators to approach the resource pool in terms of applications rather than infrastructure building blocks. The application profile is a container that holds EPGs that are logically related to one another. EPGs can communicate with other EPGs in the same application profile and with EPGs in other application profiles.

To deploy an application policy, you must create the required application profiles, filters, and contracts. Typically, the APIC fabric hosts a three-tier application within a tenant network. In this example, the application is implemented by using three servers (a web server, an application server, and a database server). See the following figure for an example of a three-tier application.

The web server has the HTTP filter, the application server has the Remote Method Invocation (RMI) filter, and the database server has the Structured Query Language (SQL) filter. The application server consumes the SQL contract to communicate with the database server. The web server consumes the RMI contract to communicate with the application server. The traffic enters from the web server and communicates with the application server. The application server then communicates with the database server, and the traffic can also communicate externally.

Figure 8: Three-Tier Application Diagram



Parameters to Create a Filter for http

The parameters to create a filter for http in this example is as follows:

Parameter Name	Filter for http
Name	http
Number of Entries	2
Entry Name	Dport-80 Dport-443
Ethertype	IP
Protocol	tcp tcp
Destination Port	http https

Parameters to Create Filters for rmi and sql

The parameters to create filters for rmi and sql in this example are as follows:

Parameter Name	Filter for rmi	Filter for sql
Name	rmi	sql
Number of Entries	1	1
Entry Name	Dport-1099	Dport-1521
Ethertype	IP	IP
Protocol	tcp	tcp

Parameter Name	Filter for rmi	Filter for sql
Destination Port	1099	1521

Example Application Profile Database

The application profile database in this example is as follows:

EPG	Provided Contracts	Consumed Contracts
web	web	rmi
app	rmi	sql
db	sql	--

Creating an Application Profile Using the GUI

Procedure

Step 1 On the menu bar, choose **TENANTS**. In the **Navigation** pane, expand the tenant, right-click **Application Profiles**, and click **Create Application Profile**.

Step 2 In the **Create Application Profile** dialog box, in the **Name** field, add the application profile name (OnlineStore).

Creating EPGs Using the GUI

The port the EPG uses must belong to one of the VM Managers (VMM) or physical domains associated with the EPG.

Procedure

Step 1 On the menu bar, choose **Tenants** and the tenant where you want to create an EPG.

Step 2 In the navigation pane, expand the folder for the tenant, the **Application Profiles** folder, and the folder for the application profile.

Step 3 Right-click the **Application EPG** folder, and in the **Create Application EPG** dialog box, perform the following actions:

a) In the **Name** field, add the EPG name (db).

b) In the **Bridge Domain** field, choose the bridge domain from the drop-down list (bd1).

c) Check the **Associate to VM Domain Profiles** check box. Click **Next**.

d) In the **STEP 2 > Domains** area, expand **Associate VM Domain Profiles** and from the drop-down list, choose the desired VMM domain.

e) From the **Deployment Immediacy** drop-down list, accept the default or choose when policies are deployed from Cisco APIC to the physical leaf switch.

f) From the **Resolution Immediacy** drop-down list, choose when policies are deployed from the physical leaf switch to the virtual leaf.

If you have Cisco AVS, choose **Immediate** or **On Demand**; if you have Cisco ACI Virtual Edge or VMware VDS, choose **Immediate**, **On Demand**, or **Pre-provision**.

g) (Optional) In the **Delimiter** field, enter one of the following symbols: |, ~, !, @, ^, +, or =.

If you do not enter a symbol, the system uses the default | delimiter in the VMware portgroup name.

h) If you have Cisco ACI Virtual Edge or Cisco AVS, from the **Encap Mode** drop-down list, choose an encapsulation mode.

You can choose one of the following encapsulation modes:

- **VXLAN**:This overrides the domain's VLAN configuration, and the EPG uses VXLAN encapsulation. However, a fault is for the EPG if a multicast pool is not configured on the domain.
- **VLAN**: This overrides the domain's VXLAN configuration, and the EPG uses VLAN encapsulation. However, a fault is triggered for the EPG if a VLAN pool is not configured on the domain.
- **Auto**: This causes the EPG to use the same encapsulation mode as the VMM domain. This is the default configuration.

i) If you have Cisco ACI Virtual Edge, from the **Switching Mode** drop-down list, choose **native** or **AVE**.

If you choose **native**, the EPG is switched through the VMware VDS; if you choose **AVE**, the EPG is switched through the Cisco ACI Virtual Edge. The default is **native**.

j) Click **Update** and then click **Finish**.

Step 4 In the **Create Application Profile** dialog box, create two more EPGs. Create the three EPGs—db, app, and web—in the same bridge domain and data center.

Configuring Contracts Using the APIC GUI

Guidelines and Limitations for Contracts and Filters

If your fabric consists of first-generation Cisco Nexus 9300 leaf switches, such as Cisco Nexus 93128TX, 93120TX, 9396TX, 9396PX, 9372PX, 9372PX-E, 9372TX and 9372TX-E, only **IP** as an **EtherType** match is supported with contract filters. The capability to match more granular options, such as **IPv4** or **IPv6**, in the **EtherType** field for contract filters is supported only on leaf switch models with -EX, -FX, or -FX2 at the end of the switch name.

Creating a Filter Using the GUI

Create three separate filters. In this example they are HTTP, RMI, SQL. This task shows how to create the HTTP filter. The task is identical for creating the other filters.

Before you begin

Verify that the tenant, network, and bridge domain have been created.

Procedure

Step 1 On the menu bar, choose **Tenants**. In the **Navigation** pane, expand the ***tenant-name*** > **Contracts**, right-click **Filters**, and click **Create Filter**.

Note In the **Navigation** pane, you expand the tenant where you want to add filters.

Step 2 In the **Create Filter** dialog box, perform the following actions:

a) In the **Name** field, enter the filter name (http).
b) Expand **Entries**, and in the **Name** field, enter the name (Dport-80).
c) From the **EtherType** drop-down list, choose the EtherType (IP).
d) From the **IP Protocol** drop-down list, choose the protocol (tcp).
e) From the **Destination Port/Range** drop-down lists, choose **http** in the **From** and **To** fields. (http)
f) Click **Update**, and click **Submit**.
The newly added filter appears in the **Navigation** pane and in the **Work** pane.

Step 3 Expand **Entries** in the **Name** field. Follow the same process to add another entry with HTTPS as the **Destination** port, and click **Update**.

This new filter rule is added.

Step 4 Follow the same process in the earlier steps to create two more filters (rmi and sql) and use the parameters provided in Parameters to Create Filters for rmi and sql, on page 39.

Creating a Contract Using the GUI

Procedure

Step 1 On the menu bar, choose **Tenants** and the tenant name on which you want to operate. In the **Navigation** pane, expand the ***tenant-name*** > **Contracts**.

Step 2 Right-click **Standard** > **Create Contract**.

Step 3 In the **Create Contract** dialog box, perform the following tasks:

a) In the **Name** field, enter the contract name (web).
b) Click the **+** sign next to **Subjects** to add a new subject.
c) In the **Create Contract Subject** dialog box, enter a subject name in the **Name** field. (web)
d) **Note** This step associates the filters created that were earlier with the contract subject.

In the **Filter Chain** area, click the **+** sign next to **Filters**.
e) In the dialog box, from the drop-down menu, choose the filter name (http), and click **Update**.

Step 4 In the **Create Contract Subject** dialog box, click **OK**.

Step 5 Create two more contracts for rmi and for sql following the same steps in this procedure. For the rmi contract, choose the rmi subject and for sql, choose the sql subject.

Consuming and Providing Contracts Using the GUI

You can associate contracts that were created earlier to create policy relationships between the EPGs.

When you name the provided and consumed contracts, verify that you give the same name for both provided and consumed contracts.

Procedure

Step 1 Click and drag across the APIC GUI window from the db EPG to the app EPG.
The **Add Consumed Contract** dialog box is displayed.

Note The db, app, and web EPGs are displayed as icons.

Step 2 In the **Name** field, from the drop-down list, choose **sql** contract. Click **OK**.
This step enables the db EPG to provide the sql contract and the app EPG to consume the sql contract.

Step 3 Click and drag across the APIC GUI screen from the app ePG to the web EPG.
The **Add Consumed Contract** dialog box is displayed.

Step 4 In the **Name** field, from the drop-down list, choose **rmi** contract. Click **OK**.
This step enables the app EPG to provide the rmi contract and the web EPG to consume the rmi contract.

Step 5 Click the web EPG icon, and click the **+** sign in the **Provided Contracts** area.
The **Add Provided Contract** dialog box is displayed.

Step 6 In the **Name** field, from the drop-down list, choose **web** contract. Click **OK**. Click **Submit**.
You have created a three-tier application profile called OnlineStore.

Step 7 To verify, in the **Navigation** pane, navigate to and click **OnlineStore** under **Application Profiles**.
In the **Work** pane, you can see the three EPGs app, db, and web are displayed.

Step 8 In the **Work** pane, choose **Operational** > **Contracts**.
You can see the EPGs and contracts displayed in the order that they are consumed and provided.

Configuring Contracts Using the NX-OS Style CLI

Configuring Contracts

Contracts are configured under a tenant with the following tasks:

- Define filters as access lists
- Define the contract and subjects
- Link the contract to an EPG

The tasks need not follow this order. For example, you can link a contract name to an EPG before you have defined the contract.

Note Filters (ACLs) in APIC use **match** instead of **permit | deny** as in the traditional NX-OS ACL. The purpose of a filter entry is only to match a given traffic flow. The traffic will be permitted or denied when the ACL is applied on a contract or on a taboo contract.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters configuration mode.
Step 2	**tenant** *tenant-name* **Example:** `tenant exampleCorp`	Creates a tenant if it does not exist and enters the tenant configuration mode.
Step 3	**access-list** *acl-name* **Example:** `apic1(config-tenant)# access-list http_acl`	Creates an access list (filter) that can be used in a contract.
Step 4	(Optional) **match** {**arp** | **icmp** | **ip**} **Example:** `apic1(config-tenant-acl)# match arp`	Creates a rule to match traffic of the selected protocol.

Step 5	(Optional) **match** {**tcp** | **udp**} [**src** *from*[*-to*]] [**dest** *from*[*-to*]]
	Example:
	`apic1(config-tenant-acl)# match tcp dest 80`
	`apic1(config-tenant-acl)# match tcp dest 443`
		Creates a rule to match TCP or UDP traffic.

Step 6	(Optional) **match raw** *options* **Example:** `apic1(config-tenant-acl)#`	Creates a rule to match a raw vzEntry.
Step 7	**exit** **Example:** `apic1(config-tenant-acl)# exit`	Returns to the tenant configuration mode.
Step 8	**contract** *contract-name* **Example:** `apic1(config-tenant)# contract web80`	Creates a contract and enters the contract configuration mode.
Step 9	**subject** *subject-name* **Example:** `apic1(config-tenant-contract)# subject web80`	Creates a contract subject and enters the subject configuration mode.

	Command or Action	Purpose
Step 10	(Optional) [**no**] **access-group** *acl-name* [**in** | **out** | **both**] **Example:** `apic1(config-tenant-contract-subj)# access-group http_acl both`	Adds (removes) an access list from the contract, specifying the direction of the traffic to be matched.
Step 11	(Optional) [**no**] **label name** *label-name* {**provider** | **consumer**} **Example:** `apic1(config-tenant-contract-subj)#`	Adds (removes) a provider or consumer label to the subject.
Step 12	(Optional) [**no**] **label match** {**provider** | **consumer**} [**any** | **one** | **all** | **none**] **Example:** `apic1(config-tenant-contract-subj)#`	Specifies the match type for the provider or consumer label: • **any**—Match if any label is found in the contract relation. • **one**—Match if exactly one label is found in the contract relation. • **all**—Match if all labels are found in the contract relation. • **none**—Match if no labels are found in the contract relation.
Step 13	**exit** **Example:** `apic1(config-tenant-contract-subj)# exit`	Returns to the contract configuration mode.
Step 14	**exit** **Example:** `apic1(config-tenant-contract)# exit`	Returns to the tenant configuration mode.
Step 15	**application** *app-name* **Example:** `apic1(config-tenant)# application OnlineStore`	Enters application configuration mode.
Step 16	**epg** *epg-name* **Example:** `apic1(config-tenant-app)# epg exampleCorp_webepg1`	Enters configuration mode for the EPG to be linked to the contract.
Step 17	**bridge-domain member** *bd-name* **Example:** `apic1(config-tenant-app-epg)# bridge-domain member exampleCorp_bd1`	Specifies the bridge domain for this EPG.

	Command or Action	Purpose
Step 18	**contract provider** *provider-contract-name* **Example:** `apic1(config-tenant-app-epg)# contract provider web80`	Specifies the provider contract for this EPG. Communication with this EPG can be initiated from other EPGs as long as the communication complies with this provider contract.
Step 19	**contract consumer** *consumer-contract-name* **Example:** `apic1(config-tenant-app-epg)# contract consumer rmi99`	Specifies the consumer contract for this EPG. The endpoints in this EPG may initiate communication with any endpoint in an EPG that is providing this contract.

Examples

This example shows how to create and apply contracts to an EPG.

```
apic1# configure
apic1(config)# tenant exampleCorp

    # CREATE FILTERS
apic1(config-tenant)# access-list http_acl
apic1(config-tenant-acl)# match tcp dest 80
apic1(config-tenant-acl)# match tcp dest 443
apic1(config-tenant-acl)# exit

    # CREATE CONTRACT WITH FILTERS
apic1(config-tenant)# contract web80
apic1(config-tenant-contract)# subject web80
apic1(config-tenant-contract-subj)# access-group http_acl both
apic1(config-tenant-contract-subj)# exit
apic1(config-tenant-contract)# exit

    # ASSOCIATE CONTRACTS TO EPG
apic1(config-tenant)# application OnlineStore
apic1(config-tenant-app)# epg exampleCorp_webepg1
apic1(config-tenant-app-epg)# bridge-domain member exampleCorp_bd1
apic1(config-tenant-app-epg)# contract consumer rmi99
apic1(config-tenant-app-epg)# contract provider web80
apic1(config-tenant-app-epg)# exit
apic1(config-tenant-app)#exit
apic1(config-tenant)#exit

    # ASSOCIATE PORT AND VLAN TO EPG
apic1(config)#leaf 101
apic1(config-leaf)# interface ethernet 1/4
apic1(config-leaf-if)# switchport trunk allowed vlan 102 tenant exampleCorp application
 OnlineStore epg exampleCorp_webepg1
```

This example shows a simpler method for defining a contract by declaring the filters inline in the contract itself.

```
apic1# configure
apic1(config)# tenant exampleCorp
apic1(config-tenant)# contract web80
apic1(config-tenant-contract)# match tcp 80
```

```
apic1(config-tenant-contract)# match tcp 443
```

Exporting a Contract to Another Tenant

You can export a contract from one tenant and import it to another. In the tenant that imports the contract, the contract can be applied only as a consumer contract. The contract can be renamed during the export.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters configuration mode.
Step 2	**tenant** *tenant-name* **Example:** `apic1(config)# tenant RedCorp`	Enters the tenant configuration mode for the exporting tenant.
Step 3	**contract** *contract-name* **Example:** `apic1(config-tenant)# contract web80`	Enters the contract configuration mode for the contract to be exported.
Step 4	**scope** {**application** | **exportable** | **tenant** | **vrf**} **Example:** `apic1(config-tenant-contract)# scope exportable`	Configures how the contract can be shared. The scope can be: • **application**—Can be shared among the EPGs of the same application. • **exportable**—Can be shared across tenants. • **tenant**—Can be shared among the EPGs of the same tenant. • **vrf**—Can be shared among the EPGs of the same VRF.
Step 5	**export to tenant** *other-tenant-name* **as** *new-contract-name* **Example:** `apic1(config-tenant-contract)# export to tenant BlueCorp as webContract1`	Exports the contract to the other tenant. You can use the same contract name or you can rename it.
Step 6	**exit** **Example:** `apic1(config-tenant-contract)# exit`	Returns to the tenant configuration mode.

	Command or Action	Purpose
Step 7	**exit** **Example:** `apic1(config-tenant)# exit`	Returns to the global configuration mode.
Step 8	**tenant** *tenant-name* **Example:** `tenant BlueCorp`	Enters the tenant configuration mode for the importing tenant.
Step 9	**application** *app-name* **Example:** `apic1(config-tenant)# application BlueStore`	Enters application configuration mode.
Step 10	**epg** *epg-name* **Example:** `apic1(config-tenant-app)# epg BlueWeb`	Enters configuration mode for the EPG to be linked to the contract.
Step 11	**contract consumer** *consumer-contract-name* **imported** **Example:** `apic1(config-tenant-app-epg)# contract consumer webContract1 imported`	Specifies the imported consumer contract for this EPG. The endpoints in this EPG may initiate communication with any endpoint in an EPG that is providing this contract.

Examples

This example shows how to export a contract from the tenant RedCorp to the tenant BlueCorp, where it will be a consumer contract.

```
apic# configure
apic1(config)# tenant RedCorp
apic1(config-tenant)# contract web80
apic1(config-tenant-contract)# scope exportable
apic1(config-tenant-contract)# export to tenant BlueCorp as webContract1
apic1(config-tenant-contract)# exit
apic1(config-tenant)# exit
apic1(config)# tenant BlueCorp
apic1(config-tenant)# application BlueStore
apic1(config-tenant-application)# epg BlueWeb
apic1(config-tenant-application-epg)# contract consumer webContract1 imported
```

Configuring Contracts Using the REST API

Configuring a Contract Using the REST API

Procedure

Configure a contract using an XML POST request similar to the following example:

Example:

```
<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>
```

Configuring a Taboo Contract Using the REST API

Before you begin

The following objects must be created:

- The tenant that will be associated with this **Taboo Contract**
- An application profile for the tenant
- At least one EPG for the tenant

Procedure

To create a taboo contract with the REST API, use XML such as in the following example:

Example:

```
<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>
```

Verifying Contracts, Taboo Contracts, and Filters Using the REST API

This topic provides the REST API XML to verify contracts, taboo contracts, and filters.

Procedure

Step 1 Verify a contract for an EPG or an external network with XML such as the following example for a provider:

Example:

```
QUERY https://apic-ip-address/api/node/class/fvRsProv.xml
```

Step 2 Verify a contract on an EPG with XML such as the following example for a consumer:

Example:

```
QUERY https://apic-ip-address/api/node/class/fvRsCons.xml
```

Step 3 Verify exported contracts using XML such as the following example:

Example:

```
QUERY https://apic-ip-address/api/node/class/vzCPif.xml
```

Step 4 Verify contracts for a VRF with XML such as the following example:

Example:

```
QUERY https://apic-ip-address/api/node/class/vzBrCP.xml
```

Step 5 Verify taboo contracts with XML such as the following example:

Example:

```
QUERY https://apic-ip-address/api/node/class/vzTaboo.xml
```

For taboo contracts for an EPG, use the same query as for contracts for EPGs.

Step 6 Verify filters using XML such as the following example:

Example:

```
QUERY https://apic-ip-address/api/node/class/vzFilter.xml
```

Optimize Contract Performance

Optimize Contract Performance

Starting with Cisco APIC, Release 3.2, you can configure bidirectional contracts that support more efficient hardware TCAM storage of contract data. With optimization enabled, contract statistics for both directions are aggregated.

TCAM Optimization is supported on the second generation Cisco Nexus 9000 Series top of rack (TOR) switches, which are those with suffixes of EX, FX, and FX2, and later (for example, N9K-C93180LC-EX or N9K-C93180YC-FX).

To configure efficient TCAM contract data storage, you enable the following options:

- Mark the contracts to be applied in both directions between the provider and consumer.
- For filters with IP TCP or UDP protocols, enable the reverse port option.
- When configuring the contract subjects, select the **Enable Policy Compression** directive, which adds the `no_stats` option to the `action` attribute of the `actrl:Rule` managed object.

Limitations

With the **Enable Policy Compression** (`no_stats`) option selected, per-rule statistics are lost. However, combined rule statistics for both directions are present in the hardware statistics.

After upgrading to Cisco APIC 3.2(1), to add the `no_stats` option to a pre-upgrade contract subject (with filters or filter entries), you must delete the contract subject and reconfigure it with the **Enable Policy Compression** directive. Otherwise, compression does not occur.

For each contract with a bi-directional subject filter, Cisco NX-OS creates 2 rules:

- A rule with an `sPcTag` and `dPcTag` that is marked `direction=bi-dir`, which is programmed in hardware
- A rule marked with `direction=uni-dir-ignore` which is not programmed

Rules with the following settings are not compressed:

- Rules with priority other than `fully_qual`
- Opposite rules (`bi-dir` and `uni-dir-ignore` marked) with non-identical properties, such as **action** including **directives**, **prio, qos** or **markDscp**
- Rule with `Implicit` or `implarp` filters
- Rules with the actions `Deny`, `Redir`, `Copy`, or `Deny-log`

The following MO query output shows the two rules for a contract, that is considered for compression:

```
apic1# moquery -c actrlRule
Total Objects shown: 2

# actrl.Rule
scopeId            : 2588677
sPcTag             : 16388
dPcTag             : 49156
fltId              : 67
action             : no_stats,permit
actrlCfgFailedBmp  :
actrlCfgFailedTs   : 00:00:00:00.000
actrlCfgState      : 0
childAction        :
ctrctName          :
descr              :
direction          : bi-dir
dn                 : sys/actrl/scope-2588677/rule-2588677-s-16388-d-49156-f-67
id                 : 4112
lcOwn              : implicit
markDscp           : unspecified
modTs              : 2019-04-27T09:01:33.152-07:00
monPolDn           : uni/tn-common/monepg-default
name               :
nameAlias          :
operSt             : enabled
```

```
operStQual          :
prio                : fully_qual
qosGrp              : unspecified
rn                  : rule-2588677-s-16388-d-49156-f-67
status              :
type                : tenant


# actrl.Rule
scopeId             : 2588677
sPcTag              : 49156
dPcTag              : 16388
fltId               : 64
action              : no_stats,permit
actrlCfgFailedBmp   :
actrlCfgFailedTs    : 00:00:00:00.000
actrlCfgState       : 0
childAction         :
ctrctName           :
descr               :
direction           : uni-dir-ignore
dn                  : sys/actrl/scope-2588677/rule-2588677-s-49156-d-16388-f-64
id                  : 4126
lcOwn               : implicit
markDscp            : unspecified
modTs               : 2019-04-27T09:01:33.152-07:00
monPolDn            : uni/tn-common/monepg-default
name                :
nameAlias           :
operSt              : enabled
operStQual          :
prio                : fully_qual
qosGrp              : unspecified
rn                  : rule-2588677-s-49156-d-16388-f-64
status              :
type                : tenant
```

Table 1: Compression Matrix

Reverse Filter Port Enabled	TCP or UDP Source Port	TCP or UCP Destination Port	Compressed
Yes	Port A	Port B	Yes
Yes	Unspecified	Port B	Yes
Yes	Port A	Unspecified	Yes
Yes	Unspecified	Unspecified	Yes
No	Port A	Port B	No
No	Unspecified	Port B	No
No	Port A	Unspecified	No
No	Unspecified	Unspecified	Yes

Configure a Contract with Optimized TCAM Usage Using the GUI

This procedure describes how to configure a contract that optimizes TCAM storage of contract data on hardware.

Before you begin

- Create the tenant, VRF, and EPGs that will provide and consume the contract.
- Create one or more filters that define the traffic to be permitted or denied by this contract.

Procedure

Step 1 On the menu bar, choose **Tenants** and the tenant name on which you want to operate. In the **Navigation** pane, expand the *tenant-name* and **Contracts**.

Step 2 Right-click **Standard** > **Create Contract**.

Step 3 In the **Create Contract** dialog box, perform the following tasks:

a) In the **Name** field, enter the contract name.
b) Click the + icon next to **Subjects** to add a new subject.
c) In the **Create Contract Subject** dialog box, enter a subject name in the **Name** field.

Note This step associates filters with the contract subject.

d) To enable the TCAM-contract usage optimization feature, ensure that **Apply Both Directions** and **Reverse Filter Ports** are enabled.
e) Click the + icon to expand **Filters**.
f) In the dialog box, from the drop-down menu, choose a default filter, a previously configured filter, or **Create Filter**.
g) In the **Directives** field, choose **Enable Policy Compression**
h) In the **Action** field, choose **Permit** or **Deny**.

Note Currently, the **Deny** action is not supported. Optimization only occurs for the **Permit** action.

i) (Optional) In the **Priority** field, choose the priority level.
j) Click **Update**.

Step 4 In the **Create Contract Subject** dialog box, click **OK**.

Step 5 In the **Create Contract** dialog box, click **Submit**.

Configure a Contract with Optimized TCAM Usage Using the REST API

Before you begin

Create the tenant, VRF, and EPGs that will provide and consume the contract.

Procedure

To configure a filter and contract that optimizes TCAM storage of contract data on hardware, send a post with XML similar to the following example:

Example:

```
<vzFilter dn="uni/tn-Tenant64/flt-webFilter" name="webFilter>
    <vzEntry applyToFrag="no" dFromPort="https" dToPort="https"
        dn="uni/tn-Tenant64/flt-webFilter/e-https" etherT="ip" name="https" prot="tcp"
stateful="no"/>
    </vzFilter>
<vzBrCP dn="uni/tn-Tenant64/brc-OptimizedContract" name="OptimizedContract"
provMatchT="AtleastOne" revFltPorts="yes">
    <vzSubj consMatchT="AtleastOne" dn="uni/tn-Tenant64/brc-OptimizedContract/subj-WebSubj"
 lcOwn="local" name="WebSubj"
          provMatchT="AtleastOne" revFltPorts="yes">
         <vzRsSubjFiltAtt action="permit" directives="no_stats" forceResolve="yes"
lcOwn="local" tCl="vzFilter"
          tDn="uni/tn-Tenant64/flt-webFilter" tRn="flt-webFilter" tType="name"
tnVzFilterName="webFilter"/>
    </vzSubj>
</vzBrCP>
```

Contract and Subject Exceptions

Configuring Contract or Subject Exceptions for Contracts

In Cisco APIC Release 3.2(1), contracts between EPGs are enhanced to enable denying a subset of contract providers or consumers from participating in the contract. Inter-EPG contracts and Intra-EPG contracts are supported with this feature.

You can enable a provider EPG to communicate with all consumer EPGs except those that match criteria configured in a subject or contract exception. For example, if you want to enable an EPG to provide services to all EPGs for a tenant, except a subset, you can enable those EPGs to be excluded. To configure this, you create an exception in the contract or one of the subjects in the contract. The subset is then denied access to providing or consuming the contract.

Labels, counters, and permit and deny logs are supported with contracts and subject exceptions.

To apply an exception to all subjects in a contract, add the exception to the contract. To apply an exception only to a single subject in the contract, add the exception to the subject.

When adding filters to subjects, you can set the action of the filter (to permit or deny objects that match the filter criteria). Also for **Deny** filters, you can set the priority of the filter. **Permit** filters always have the default priority. Marking the subject-to-filter relation to deny automatically applies to each pair of EPGs where there is a match for the subject. Contracts and subjects can include multiple subject-to-filter relationships that can be independently set to permit or deny the objects that match the filters.

Exception Types

Contract and subject exceptions can be based on the following types and include regular expressions, such as the * wildcard:

Exception criteria exclude these objects as defined in the Consumer Regex and Provider Regex fields	Example	Description
Tenant	`<vzException consRegex= "common" field= "Tenant" name= "excep03" provRegex= "t1" />`	This example, excludes EPGs using the `common` tenant from consuming contracts provided by the `t1` tenant.
VRF	`<vzException consRegex= "ctx1" field= "Ctx" name= "excep05" provRegex= "ctx1" />`	This example excludes members of `ctx1` from consuming the services provided by the same VRF.
EPG	`<vzException consRegex= "EPgPa.*" field= "EPg" name= "excep03" provRegex= "EPg03" />`	The example assumes that multiple EPGs exist, with names starting with `EPGPa`, and they should all be denied as consumers for the contract provided by `EPg03`
Dn	`<vzException consRegex= "uni/tn-t36/ap-customer/epg-epg193" field= "Dn" name="excep04" provRegex= "uni/tn-t36/ap-customer/epg-epg200" />`	This example excludes `epg193` from consuming the contract provided by `epg200`.
Tag	`<vzException consRegex= "red" field= "Tag" name= "excep01" provRegex= "green" />`	The example excludes objects marked with the `red` tag from consuming and those marked with the `green` tag from participating in the contract.

Configure a Contract or Subject Exception Using the GUI

In this task, you configure a contract that will allow most of the EPGs to communicate, but deny access to a subset of them.

Before you begin

Configure the tenant, VRF, application profile, and EPGs that provide and consume the contract.

Procedure

Step 1 Click **Tenants** > **All Tenants** on the menu bar.

Step 2 Double-click the tenant in which you are creating the contract.

Step 3 On the navigation bar, expand **Contracts**, right-click **Filter**, and choose **Create Filter**.

A filter is essentially an Access Control List (ACL) that defines the traffic that is permitted or denied access through the contract. You can create multiple filters that define objects that can be permitted or denied.

Step 4 Enter the filter name and add the criteria that define the traffic to permit or deny, then click **Submit**.

Step 5 Right-click **Standard**, and choose **Create Contract**.

Step 6 Enter the contract name, set the scope, and click the + icon to add a subject.

Step 7 Repeat to add another subject.

Step 8 Click **Submit**

Step 9 To add an exception to all subjects in the contract, perform the following steps:

a) Click the contract, then click **Contract Exception**.
b) Add subjects and set them to be permitted or denied.
c) Click the + icon to add a contract exception.
d) Enter the exception name and type.
e) Add regular expressions in the **Consumer Regex** and **Provider Regex** fields to define the EPGs to be excluded from all subjects in the contract.

Step 10 To add an exception to one subject in the contract, perform the following steps:

a) Click the subject, then click **Subject Exception**.
b) Click the + icon to add a contract exception.
c) Enter the exception name and type.
d) Add regular expressions in the **Consumer Regex** and **Provider Regex** to define the EPGs to be excluded from all subjects in the contract.

Configure a Contract or Subject Exception Using the NX-OS Style CLI

In this task, you configure a contract that will allow most of the EPGs to communicate, but deny access to a subset of them. Multiple exceptions can be added to a contract or a subject.

Before you begin

Configure the tenant, VRF, application profile, and EPGs to provide and consume the contract.

Procedure

Step 1 Configure filters for HTTP and HTTPS, using commands as in the following example:

Example:

```
apic1(config)# tenant t2
apic1(config-tenant)# access-list ac1
apic1(config-tenant-acl)# match ip
apic1(config-tenant-acl)# match tcp dest 80
apic1(config-tenant-acl)# exit
apic1(config-tenant)# access-list ac2
apic1(config-tenant-acl)# match ip
apic1(config-tenant-acl)# match tcp dest 443
```

Step 2 Configure a contract that excludes `EPg01` from consuming it and `EPg03` from providing it.

Example:

```
apic1(config-tenant)# contract webCtrct
apic1(config-tenant-contract)# subject https-subject
apic1(config-tenant-contract-subj)# exception name EPG consumer-regexp EPg01 field EPg
provider-regexp EPg03
apic1(config-tenant-contract-subj)# access-group ac1 in blacklist
apic1(config-tenant-contract-subj)# access-group ac2 in whitelist
```

Configure a Contract or Subject Exception Using the REST API

In this task, you configure a contract that will allow most of the EPGs to communicate, but deny access to a subset of them. Multiple exceptions can be added to a contract or a subject.

Before you begin

Configure the tenant, VRF, application profile, and EPGs to provide and consume the contract.

Procedure

Step 1 Create a filter by sending a post with XML, such as the following example:

Example:

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

Step 2 Create a contract that excludes `EPg01` from consuming the subject and `EPg03` from providing it, by sending a post with XML, such as the following example:

The vzException MO can be contained by the vzBrCP or vzSubj MOs.

Example:

```
<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>
```

Intra-EPG Contracts

Intra-EPG Contracts

You can configure contracts to control communication between EPGs. Beginning in Cisco APIC Release 3.0(1), you can also configure contracts within an EPG.

Without intra-EPG contracts, communication between endpoints in an EPG is all-or-nothing. Communication is unrestricted by default, or you can configure intra-EPG isolation to bar any communication between endpoints.

However, with intra-EPG contracts, you can control communication between endpoints in the same EPG, allowing some traffic and barring the rest. For example, you may want to allow web traffic but block the rest. Or you can allow all ICMP traffic and TCP port 22 traffic while blocking all other traffic.

Guidelines and Limitations for Intra-EPG Contracts

Observe the following guidelines and limitations when planning intra-EPG contracts:

- Intra-EPG contracts can be configured for application EPGs and microsegment EPGs (uSegs) on VMware VDS, Open vSwitch (OVS), and baremetal servers.

Note OVS is available in the Kubernetes integration with Cisco Application Centric Infrastructure (ACI) feature. In Kubernetes, you can create EPGs and assign namespaces to them. You can then apply intra-EPG policies to the EPGs in Cisco Application Policy Infrastructure Controller (APIC) as you would for VMware VDS or baremetal servers.

- Intra-EPG contracts require that the leaf switch support proxy Address Resolution Protocol (ARP). Intra-EPG contracts are supported on Cisco Nexus 9000 Series switches with EX or FX at the end of their model name or later models.
- Intra-EPG Contracts are not supported in Cisco Application Virtual Switch, Cisco ACI Virtual Edge, and Microsoft domains. Attempting to set intra-EPG contracts to be enforced in these domains may cause ports to go into a blocked state.
- Intra-EPG contracts in service graphs:
  - A service graph cannot be associated with a subject of an intra-EPG contract that has an action of deny.
  - Support for intra-EPG contracts in service graphs is limited to single node one-arm mode policy-based redirect and copy service.

Adding an Intra-EPG Contract to an Application EPG Using the GUI

After you configure a contract, you can add the contract to an EPG as an intra-EPG contract. The procedure is the same for VMware VDS, OVS, and baremetal servers.

Before you begin

- You must have an application EPG configured.
- You must have a contract with filters configured for this application. See Creating a Contract Using the GUI, on page 42.

Procedure

Step 1 Log in to the APIC GUI.

Step 2 Go to **Tenants** > ***tenant*** .

Step 3 Complete one of the following sets of steps, depending on the type of EPG:

If you want to apply an intra-EPG contract to...	Then...
An application EPG	Then... **a.** In the left navigation pane, expand **Application Profiles** > ***application profile*** > **Application EPGs** > ***epg*** . **b.** Right-click the **Contracts** folder and then choose **Add Intra-EPG Contract**. **c.** In the **Add Intra-EPG Contract** dialog box, from the **Contract** drop-down list, choose a contract. **d.** Click **Submit**.
A uSeg EPG	**a.** In the left navigation pane, expand **Application Profiles** > ***application profile*** > **uSeg EPGs** > ***epg*** . **b.** Right-click the **Contracts** folder and then choose **Add Intra-EPG Contract**. **c.** In the **Add Intra-EPG Contract** dialog box, from the **Contract** drop-down list, choose a contract. **d.** Click **Submit**.

Adding an Intra-EPG Contract to an Application EPG Using the NX-OS-Style CLI

After you configure a contract, you can configure the contract as an intra-EPG contract. The procedure is the same for VMware VDS, OVS, and baremetal servers.

Before you begin

- You must have an EPG configured.
- You must have a contract with filters configured.

Procedure

Step 1 Enter the configuration mode.

Example:

```
apic1# configure
```

Step 2 Create or choose a tenant.

Example:

```
apic1(config)# tenant Tenant-l3out
```

Step 3 Create or choose an external Layer 3 EPG.

Example:

```
apic1(config-tenant)# external-l3 epg ext-epg
```

Step 4 Bind the external EPG to a VRF instance.

Example:

```
apic1(config-tenant-l3ext-epg)# vrf member vrf1
```

Step 5 Enable intra-EPG isolation.

Example:

```
(config-tenant-l3ext-epg)# isolation enforce
```

Afterward, if necessary, you can disable intra-EPG isolation by preceding the command with `no`.

Example:

```
(config-tenant-l3ext-epg)# no isolation enforce
```

Step 6 Assign a contract to the intra-external EPG to allow the desired traffic between the endpoints.

Example:

```
apic1(config-tenant-l3ext-epg)# contract intra-epg contr-intra
```

Adding an Intra-EPG Contract to an Application EPG Using the REST API

After you configure a contract, you can add the contract to an EPG as an intra-EPG contract. The procedure is the same for VMware VDS, OVS, and baremetal servers.

Before you begin

- You must have configured an EPG.
- You must have configured a contract with filters.

Procedure

Step 1 Configure the selectors using an XML POST request similar to the following example:

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<polUni>
  <infraInfra>
    <infraAccPortP name="Ports-1-12" status="deleted"/>

    <!-- VMM VLAN range -->
    <fvnsVlanInstP name="test" allocMode="dynamic">
      <fvnsEncapBlk name="encap" from="vlan-10" to="vlan-100"/>
    </fvnsVlanInstP>

    <!-- Static VLAN range -->
    <fvnsVlanInstP name="test" allocMode="static">
      <fvnsEncapBlk name="default" from="vlan-101" to="vlan-4095"/>
    </fvnsVlanInstP>

    <infraAttEntityP name="test">
      <infraRsDomP tDn="uni/phys-test"/>
      <infraRsDomP tDn="uni/l3dom-test"/>
      <infraRsDomP tDn="uni/vmmp-VMware/dom-test"/>
    </infraAttEntityP>

    <!-- Node profile -->
    <infraNodeP name="test">
      <infraLeafS name="test" type="range">
        <infraNodeBlk name="default" from_="101" to_="102"/>
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-test"/>
    </infraNodeP>

    <!-- Port profile -->
    <infraAccPortP name="test">
      <!-- 12 regular ports -->
      <infraHPortS name="ports1Through12" type="range">
        <infraPortBlk name="default" fromCard="1" toCard="1" fromPort="1" toPort="12"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-test"/>
      </infraHPortS>

      <!-- 2 ports in PC -->
      <infraHPortS name="portsForPc1" type="range">
        <infraPortBlk name="default" fromCard="1" toCard="1" fromPort="13" toPort="14"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-testPc"/>
      </infraHPortS>

      <!-- 2 ports in PC -->
      <infraHPortS name="portsForPc2" type="range">
        <infraPortBlk name="blk1" fromCard="1" toCard="1" fromPort="15" toPort="16"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-pc"/>
      </infraHPortS>

      <!-- 2 ports in PC for FEX -->
      <infraHPortS name="portsForFex" type="range">
        <infraPortBlk name="blk1" fromCard="1" toCard="1" fromPort="17" toPort="18"/>
        <infraRsAccBaseGrp tDn="uni/infra/fexprof-default/fexbundle-test" fexId="111"/>
      </infraHPortS>

      <!-- 2 ports in PC for VPC -->
      <infraHPortS name="portsForVpc" type="range">
        <infraPortBlk name="blk1" fromCard="1" toCard="1" fromPort="19" toPort="20"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-testVpc"/>
      </infraHPortS>
    </infraAccPortP>

    <!-- FEX profile -->
```

```
  <infraFexP name="default">
    <infraFexBndlGrp name="default"/>

    <!-- 12 FEX ports -->
    <infraHPortS name="ports1Through12" type="range">
      <infraPortBlk name="default" fromCard="1" toCard="1" fromPort="1" toPort="12"/>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-test"/>
    </infraHPortS>

    <!-- 3 ports in FEX PC -->
    <infraHPortS name="portsForPc" type="range">
      <infraPortBlk name="blk1" fromCard="1" toCard="1" fromPort="13" toPort="16"/>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-testPcOnFex"/>
    </infraHPortS>

    <!-- 3 ports in FEX VPC -->
    <infraHPortS name="portsForVpc" type="range">
      <infraPortBlk name="blk1" fromCard="1" toCard="1" fromPort="17" toPort="19"/>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-testVpcOnFex"/>
    </infraHPortS>
  </infraFexP>

  <!-- Functional profile -->
  <infraFuncP>
    <!-- Regular port group -->
    <infraAccPortGrp name="test">
      <infraRsAttEntP tDn="uni/infra/attentp-test"/>
    </infraAccPortGrp>

    <!-- PC -->
    <infraAccBndlGrp name="testPc" lagT="link">
      <infraRsLacpPol tnLacpLagPolName="testPc"/>
      <infraRsAttEntP tDn="uni/infra/attentp-test"/>
    </infraAccBndlGrp>

    <!-- VPC -->
    <infraAccBndlGrp name="testVpc" lagT="node">
      <infraRsLacpPol tnLacpLagPolName="testVpc"/>
      <infraRsAttEntP tDn="uni/infra/attentp-test"/>
    </infraAccBndlGrp>

    <!-- PC on FEX -->
    <infraAccBndlGrp name="testPcOnFex" lagT="link">
      <infraRsLacpPol tnLacpLagPolName="testPcOnFex"/>
      <infraRsAttEntP tDn="uni/infra/attentp-test"/>
    </infraAccBndlGrp>

    <!-- VPC on FEX -->
    <infraAccBndlGrp name="testVpcOnFex" lagT="node">
      <infraRsLacpPol tnLacpLagPolName="testVpcOnFex"/>
      <infraRsAttEntP tDn="uni/infra/attentp-test"/>
    </infraAccBndlGrp>
  </infraFuncP>

  <!-- Link aggregation policies -->
  <lacpLagPol name="testPc" minLinks='1' maxLinks='10'/>
  <lacpLagPol name="testVpc" minLinks='1' maxLinks='10'/>
  <lacpLagPol name="testPcOnFex" minLinks='2' maxLinks='5'/>
  <lacpLagPol name="testVpcOnFex" minLinks='2' maxLinks='10'/>
</infraInfra>

<fabricInst>
  <fabricProtPol name="testVpc">
    <fabricExplicitGEp name="testVpc" id="101">
```

```
        <fabricNodePEp id="101"/>
        <fabricNodePEp id="102"/>
      </fabricExplicitGEp>
    </fabricProtPol>
  </fabricInst>

  <physDomP name="test">
    <infraRsVlanNs tDn="uni/infra/vlanns-test-static"/>
  </physDomP>

  <l3extDomP name="test">
    <infraRsVlanNs tDn="uni/infra/vlanns-test-static"/>
  </l3extDomP>
</polUni>
```

Step 2 Configure the tenant using an XML POST request similar to the following example:

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<polUni>
   <fvTenant name="Tenant-l3out">
      <vzBrCP intent="install" name="contr-intra" scope="context">
         <vzSubj consMatchT="AtleastOne" name="subj" revFltPorts="yes">
            <vzRsSubjFiltAtt action="permit" priorityOverride="default"
              tnVzFilterName="flt-ssh" />
         </vzSubj>
      </vzBrCP>
      <vzBrCP intent="install" name="contr2" scope="context">
         <vzSubj consMatchT="AtleastOne" name="contr2-subj" revFltPorts="yes">
            <vzRsSubjFiltAtt action="permit" priorityOverride="default"
              tnVzFilterName="flt-ftp" />
         </vzSubj>
      </vzBrCP>
      <vzBrCP intent="install" name="contr1" scope="context">
         <vzSubj consMatchT="AtleastOne" name="subj-http" revFltPorts="yes">
            <vzRsSubjFiltAtt action="deny" priorityOverride="default"
              tnVzFilterName="flt-http" />
         </vzSubj>
      </vzBrCP>
      <l3extOut enforceRtctrl="export" mplsEnabled="no" name="l3out1">
         <l3extRsL3DomAtt tDn="uni/l3dom-test" />
         <l3extRsEctx tnFvCtxName="vrf1" />
         <l3extLNodeP name="l3out1_nodeProfile" tag="yellow-green">
            <l3extRsNodeL3OutAtt rtrId="172.16.0.1" rtrIdLoopBack="yes"
              tDn="topology/pod-1/node-101" />
            <l3extLIfP name="l3out1_interfaceProfile" tag="yellow-green">
               <l3extRsPathL3OutAtt addr="192.168.15.1/24" autostate="disabled"
                 encap="unknown" encapScope="local" ifInstT="l3-port" ipv6Dad="enabled"
                 isMultiPodDirect="no" llAddr="::" mac="00:22:BD:F8:19:FF"
                 mode="regular" mtu="inherit"
                 tDn="topology/pod-1/paths-101/pathep-[eth1/10]" />
            </l3extLIfP>
         </l3extLNodeP>

         <!--
           Set pcEnfPref to "enforced" to enable intra-Ext-EPG isolation.
           Set pcEnfPref to "unenforced" to disable intra-Ext-EPG isolation.
         -->
         <l3extInstP floodOnEncap="disabled" matchT="AtleastOne"
           name="l3epg1" pcEnfPref="unenforced" prefGrMemb="exclude">
            <l3extSubnet ip="172.16.0.0/16" scope="import-security" />
            <fvRsCons tnVzBrCPName="contr2" />
            <fvRsIntraEpg tnVzBrCPName="contr-intra" />
         </l3extInstP>
```

```
</l3extOut>
<fvCtx bdEnforcedEnable="no" ipDataPlaneLearning="enabled" knwMcastAct="permit"
  name="vrf1" pcEnfDir="egress" pcEnfPref="unenforced" vrfIndex="0">
   <fvRsVrfValidationPol />
   <vzAny matchT="AtleastOne" prefGrMemb="disabled" />
</fvCtx>
<fvBD OptimizeWanBandwidth="no" arpFlood="yes" epClear="no" hostBasedRouting="no"
  intersiteBumTrafficAllow="no" intersiteL2Stretch="no" ipLearning="yes"
  ipv6McastAllow="no" limitIpLearnToSubnets="yes" llAddr="::"
  mac="00:22:BD:F8:19:FF" mcastAllow="no" multiDstPktAct="bd-flood" name="bd-web"
  type="regular" unicastRoute="yes" unkMacUcastAct="proxy" unkMcastAct="flood"
  v6unkMcastAct="flood" vmac="not-applicable">
   <fvSubnet ip="192.168.1.254/24" ipDPLearning="enabled" preferred="no"
     scope="private" virtual="no" />
   <fvRsCtx tnFvCtxName="vrf1" />
   <fvRsBdToEpRet resolveAct="resolve" />
</fvBD>
<fvBD OptimizeWanBandwidth="no" arpFlood="yes" epClear="no" hostBasedRouting="no"
  intersiteBumTrafficAllow="no" intersiteL2Stretch="no" ipLearning="yes"
  ipv6McastAllow="no" limitIpLearnToSubnets="yes" llAddr="::"
  mac="00:22:BD:F8:19:FF" mcastAllow="no" multiDstPktAct="bd-flood" name="bd-app"
  type="regular" unicastRoute="yes" unkMacUcastAct="proxy" unkMcastAct="flood"
  v6unkMcastAct="flood" vmac="not-applicable">
   <fvSubnet ip="192.168.2.254/24" ipDPLearning="enabled" preferred="no"
     scope="private" virtual="no" />
   <fvRsCtx tnFvCtxName="vrf1" />
   <fvRsBdToEpRet resolveAct="resolve" />
</fvBD>
<vzFilter name="flt-ftp">
   <vzEntry applyToFrag="no" arpOpc="unspecified" dFromPort="ftpData"
     dToPort="ftpData" etherT="ipv4" icmpv4T="unspecified" icmpv6T="unspecified"
     matchDscp="unspecified" name="ftp" prot="tcp" sFromPort="unspecified"
     sToPort="unspecified" stateful="no" />
</vzFilter>
<vzFilter name="flt-ssh">
   <vzEntry applyToFrag="no" arpOpc="unspecified" dFromPort="ssh" dToPort="ssh"
     etherT="ipv4" icmpv4T="unspecified" icmpv6T="unspecified"
     matchDscp="unspecified" name="ssh" prot="tcp" sFromPort="unspecified"
     sToPort="unspecified" stateful="no" />
</vzFilter>
<vzFilter name="flt-http">
   <vzEntry applyToFrag="no" arpOpc="unspecified" dFromPort="http" dToPort="http"
     etherT="ipv4" icmpv4T="unspecified" icmpv6T="unspecified"
     matchDscp="unspecified" name="flt1" prot="tcp" sFromPort="unspecified"
     sToPort="unspecified" stateful="no" />
</vzFilter>
<fvAp name="ap-app1">
   <fvAEPg floodOnEncap="disabled" hasMcastSource="no" isAttrBasedEPg="no"
     matchT="AtleastOne" name="epg-app" pcEnfPref="unenforced"
     prefGrMemb="exclude" shutdown="no">
      <fvRsProv intent="install" matchT="AtleastOne" tnVzBrCPName="contr2" />
      <fvRsProv intent="install" matchT="AtleastOne" tnVzBrCPName="contr1" />
      <fvRsPathAtt encap="vlan-103" instrImedcy="immediate" mode="native"
        primaryEncap="unknown" tDn="topology/pod-1/paths-101/pathep-[eth1/3]" />
      <fvRsDomAtt bindingType="none" classPref="encap" encap="unknown"
        encapMode="auto" epgCos="Cos0" epgCosPref="disabled" instrImedcy="lazy"
        netflowDir="both" netflowPref="disabled" numPorts="0" portAllocation="none"
        primaryEncap="unknown" primaryEncapInner="unknown" resImedcy="immediate"
        secondaryEncapInner="unknown" switchingMode="native" tDn="uni/phys-test"
        untagged="no" vnetOnly="no" />
      <fvRsBd tnFvBDName="bd-app" />
   </fvAEPg>
   <fvAEPg floodOnEncap="disabled" hasMcastSource="no"
     isAttrBasedEPg="no" matchT="AtleastOne" name="epg-web" pcEnfPref="unenforced"
```

```
            prefGrMemb="exclude" shutdown="no">
             <fvRsPathAtt encap="vlan-104" instrImedcy="immediate" mode="native"
               primaryEncap="unknown" tDn="topology/pod-1/paths-101/pathep-[eth1/4]" />
             <fvRsDomAtt bindingType="none" classPref="encap" encap="unknown"
               encapMode="auto" epgCos="Cos0" epgCosPref="disabled" instrImedcy="lazy"
               netflowDir="both" netflowPref="disabled" numPorts="0" portAllocation="none"
               primaryEncap="unknown" primaryEncapInner="unknown" resImedcy="immediate"
               secondaryEncapInner="unknown" switchingMode="native" tDn="uni/phys-test"
               untagged="no" vnetOnly="no" />
             <fvRsCons intent="install" tnVzBrCPName="contr1" />
             <fvRsBd tnFvBDName="bd-web" />
           </fvAEPg>
         </fvAp>
    </fvTenant>
</polUni>
```

EPG Contract Inheritance

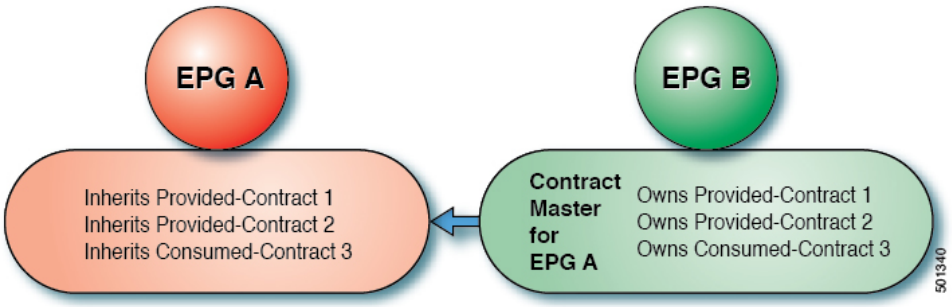
About Contract Inheritance

To streamline associating contracts to new EPGs, you can now enable an EPG to inherit all the (provided and consumed) contracts associated directly to another EPG in the same tenant. Contract inheritance can be configured for application, microsegmented, L2Out, and L3Out EPGs.

With release 3.x, you can also configure contract inheritance for Inter-EPG contracts, both provided and consumed. Inter-EPG contracts are supported on Cisco Nexus 9000 Series switches with EX or FX at the end of their model name or later models.

You can enable an EPG to inherit all the contracts associated directly to another EPG, using the APIC GUI, NX-OS style CLI, and the REST API.

Figure 9: Contract Inheritance



In the diagram above, EPG A is configured to inherit Provided-Contract 1 and 2 and Consumed-Contract 3 from EPG B (contract master for EPG A).

Use the following guidelines when configuring contract inheritance:

- Contract inheritance can be configured for application, microsegmented (uSeg), external L2Out EPGs, and external L3Out EPGs. The relationships must be between EPGs of the same type.

- Both provided and consumed contracts are inherited from the contract master when the relationship is established.
- Contract masters and the EPGs inheriting contracts must be within the same tenant.
- Changes to the masters' contracts are propagated to all the inheritors. If a new contract is added to the master, it is also added to the inheritors.
- An EPG can inherit contracts from multiple contract masters.
- Contract inheritance is only supported to a single level (cannot be chained) and a contract master cannot inherit contracts.
- Labels with contract inheritance is supported. When EPG A inherits a contract from EPG B, if different subject labels are configured under EPG A and EPG B, APIC uses the label configured under EPG B for the contract inherited from EPG B. APIC uses the label configured under EPG A for the contract where EPG A is directly involved.
- Whether an EPG is directly associated to a contract or inherits a contract, it consumes entries in TCAM. So contract scale guidelines still apply. For more information, see the *Verified Scalability Guide* for your release.
- vzAny security contracts and taboo contracts are not supported.
- Beginning in Cisco APIC release 4.2(6), contract inheritance with a service graph is supported if the contract and EPGs are in the same tenant.

For information about configuring Contract Inheritance and viewing inherited and standalone contracts, see *Cisco APIC Basic Configuration Guide*.

Configuring EPG Contract Inheritance Using the GUI

Configuring Application EPG Contract Inheritance Using the GUI

To configure contract inheritance for an application EPG, in the APIC Basic or Advanced mode GUI, use the following steps.

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Optional. Configure the bridge domain to be used by the EPG that will inherit contracts.

Configure at least one application EPG, to serve as the **EPG Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

Procedure

Step 1 Navigate to **Tenants** > ***tenant-name*** > **Application Profiles**, and expand ***AP-name***

Step 2 Right-click **Application EPGs** and select **Create Application EPG**.

Step 3 Type the name of the EPG that will inherit contracts from the **EPG Contract Master**.

Step 4 On the **Bridge Domain** field, select the common/default bridge domain or a previously created bridge domain, or create a bridge domain for this EPG.

Step 5 On the **EPG Contract Master** field, click the + symbol, select the previously configured Application Profile and EPG, and click **Update**.

Step 6 Click **Finish**.

Step 7 To view information about the EPG, including the contract master, navigate to **Tenants** > ***tenant-name*** > **Application Profiles** > ***AP-name*** > **Application EPGs** > ***EPG-name*** . To view the **EPG Contract Master**, click **General**.

Step 8 To view information about the inherited contracts, expand ***EPG-name*** and click **Contracts.**

Configuring uSeg EPG Contract Inheritance Using the GUI

To configure contract inheritance for a uSeg EPG, in the APIC Basic or Advanced mode GUI, use the following steps.

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Optional. Configure the bridge domain to be used by the EPG that will inherit contracts.

Configure the uSeg EPG, to serve as the **EPG Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

Procedure

Step 1 Navigate to **Tenants** > ***tenant-name*** > **Application Profiles**, expand ***AP-name*** .

Step 2 Right-click **uSeg EPGs** and select **Create uSeg EPG**.

Step 3 Type the name of the EPG that will inherit contracts from the contract master.

Step 4 On the **Bridge Domain** field, select the common/default bridge domain or a previously created bridge domain, or create a bridge domain for this EPG.

Step 5 Click ***uSeg-EPG-name*** . In the **EPG Contract Master** field, click the + symbol, select the Application Profile and EPG (to serve as contract master), and click **Update**.

Step 6 Click **Finish**.

Step 7 To view information about the contracts, navigate to **Tenants** > ***tenant-name*** > **Application Profiles** > ***AP-name*** > **uSeg EPGs** > , expand the ***EPG-name*** and click **Contracts.**.

Configuring L2Out EPG Contract Inheritance Using the GUI

To configure contract inheritance for an external L2Out EPG, in the APIC Advanced mode GUI, use the following steps.

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Configure an external bridged network (L2Out) and the External Network Instance Profile (L2extInstP) that will serve as the **L2Out Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

Procedure

Step 1 To configure contract inheritance for an external L2Out EPG, navigate to **Tenants** > ***tenant-name*** > **Networking** > **External Bridged Networks**, and perform the following steps:

Step 2 Expand the ***L2Out-name*** .

Step 3 Right-click **Networks** and select **Create External Network.**

Step 4 Type the name of the external network and optionally add other attributes.

Step 5 Click **Submit**.

Step 6 Expand **Networks**.

Step 7 Click the ***network-name*** .

Step 8 In the **External Network Instance Profile** panel, click the + symbol on the **L2Out Contract Masters** field.

Step 9 Select the L2Out and the L2Out Contract Master for this external L2Out EPG.

Step 10 Click **Update**.

Step 11 To view the contracts inherited by this external L2Out EPG, click on the External Network Instance Profile name and click **Contracts** > **Inherited Contracts**.

Configuring External L3Out EPG Contract Inheritance Using the Advanced GUI

To configure contract inheritance for an external L3Out EPG, in the APIC Advanced mode GUI, use the following steps.

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Configure an external routed network (L3Out) and the external network instance profile (L3extInstP) that will serve as the **L3Out Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

Procedure

Step 1 To configure contract inheritance for an external L3Out EPG, navigate to **Tenants** > ***tenant-name*** > **Networking** > **External Routed Networks**, and perform the following steps:

Step 2 Expand the ***L3Out-name*** leading to the external L3Out EPG.

Step 3 Right-click **Networks** and select **Create External Network.**

Step 4 Type the name of the external network and optionally add subnets and other attributes.

Step 5 Click **Submit**.

Step 6 Expand **Networks**.

Step 7 Click the ***network-name*** .

Step 8 In the **External Network Instance Profile** panel, click the + symbol on the **L3Out Contract Masters** field.

Step 9 Select the L3Out and Interface Profile to serve as L3Out contract master for this external L3Out EPG.

Step 10 Click **Update**.

Step 11 To view the contracts inherited by this external L3Out EPG, click on the External Network Instance Profile name and click **Contracts** > **Inherited Contracts**.

Configuring Contract Inheritance Using the NX-OS Style CLI

Configuring Application or uSeg EPG Contract Inheritance Using the NX-OS Style CLI

To configure contract inheritance for application or uSeg EPGs, use the following commands:

Before you begin

Configure the tenant, application profile, and bridge-domain to be used by the EPGs.

Configure the contracts to be shared by the EPGs at the VRF level.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters configuration mode.
Step 2	**tenant** *tenant-name* **Example:** `apic1# (config) tenant Tn1`	Creates or specifies the tenant to be configured; and enters into tenant configuration mode.
Step 3	**application** *application-name* **Example:** `apic1(config-tenant)# application AP1`	Creates or specifies an application and enters into application mode.
Step 4	**epg** *epg-name* [**type micro-segmented**] **Example:** `apic1(config-tenant-app)# epg AEPg403`	Creates or specifies the application or uSeg EPG to be configured and enters into EPG configuration mode. For uSeg EPGs add the type. In this example, this is the application EPG contract master.
Step 5	**bridge-domain member** *bd-name* **Example:** `apic1(config-tenant-app-epg)# bridge-domain member T1BD1`	Associates the EPG with the bridge domain.

	Command or Action	Purpose
Step 6	**contract consumer** *contract-name* **Example:** `apic1(config-tenant-app-epg)# contract consumer cctr5`	Adds a contract to be consumed by this EPG.
Step 7	**contract provider** [**label** *label*] **Example:** `apic1(config-tenant-app-epg)# contract provider T1ctrl_cif`	Adds a contract to be provided by this EPG, including an optional list of subject or EPG labels (must be previously configured).
Step 8	**exit** **Example:** `apic1(config-tenant-app-epg)# exit`	Exits the configuration mode
Step 9	**epg** *epg-name* [**type micro-segmented**] **Example:** `apic1(config-tenant-app)# epg AEPg404`	Creates or specifies the application or uSeg EPG to be configured and enters into EPG configuration mode. For uSeg EPGs add the type. In this example, this is the EPG inheriting contracts.
Step 10	**bridge-domain member** *bd-name* **Example:** `apic1(config-tenant-app-epg)# bridge-domain member T1BD1`	Associates the EPG with the bridge domain.
Step 11	**inherit-from-epg application** *application-name***epg** *EPG-contract-master-name*] **Example:** `apic1(config-tenant-app-epg)# inherit-from-epg application AP1 epg AEPg403`	Configures this EPG to inherit contracts from the EPG contract master.
Step 12	**exit** **Example:** `apic1(config-tenant-app-epg)# exit`	Exits the configuration mode
Step 13	**epg** *epg-name* [**type micro-segmented**] **Example:** `apic1(config-tenant-app)# epg uSeg1_403_10 type micro-segmented`	Creates or specifies the application or uSeg EPG to be configured and enters into EPG configuration mode. In this example, this is the uSeg EPG contract master.
Step 14	**bridge-domain member** *bd-name* **Example:**	Associates the EPG with the bridge domain.

	Command or Action	Purpose
	`apic1(config-tenant-app-epg)# bridge-domain member T1BD1`	
Step 15	**contract provider** [**label** *label*] **Example:** `apic1(config-tenant-app-epg)# contract provider T1ctrl_uSeg_l3out`	Adds a contract to be provided by this EPG, including an optional list of subject or EPG labels (must be previously configured).
Step 16	**attribute-logical-expression** *logical-expression* **Example:** `apic1(config-tenant-app-epg)# attribute-logical-expression 'ip equals 192.168.103.10 force'`	Adds a logical expression to the uSeg EPG as matching criteria.
Step 17	**exit** **Example:** `apic1(config-tenant-app-epg)# exit`	Exits the configuration mode
Step 18	**epg** *epg-name* [**type micro-segmented**] **Example:** `apic1(config-tenant-app)# epg uSeg1_403_30 type micro-segmented`	Creates or specifies the application or uSeg EPG to be configured and enters into EPG configuration mode. In this example, this is the uSeg EPG that inherits contracts from the EPG contract master.
Step 19	**bridge-domain member** *bd-name* **Example:** `apic1(config-tenant-app-epg)# bridge-domain member T1BD1`	Associates the EPG with the bridge domain.
Step 20	**attribute-logical-expression** *logical-expression* **Example:** `apic1(config-tenant-app-epg)# attribute-logical-expression 'ip equals 192.168.103.30 force'`	Adds a logical expression to the uSeg EPG as criteria.
Step 21	**inherit-from-epg application** *application-name* **epg** *EPG-contract-master-name* **Example:** `apic1(config-tenant-app-epg)# inherit-from-epg application AP1 epg uSeg1_403_10`	Configures this EPG to inherit contracts from the EPG contract master.
Step 22	**exit** **Example:**	Exits the configuration mode

	Command or Action	Purpose
	`apic1(config-tenant-app-epg)# exit`	
Step 23	**exit** **Example:** `apic1(config-tenant-app)# exit`	Exits the configuration mode
Step 24	**exit** **Example:** `apic1(config-tenant)# exit`	Exits the configuration mode
Step 25	**exit** **Example:** `apic1(config)# exit`	Exits the configuration mode

Example

```
ifav90-ifc1# show running-config tenant Tn1 application AP1
# Command: show running-config tenant Tn1 application AP1
# Time: Fri Apr 28 17:28:32 2017
  tenant Tn1
    application AP1
      epg AEPg403
        bridge-domain member T1BD1
        contract consumer cctr5 imported
        contract provider T1ctr1_cif
        exit
      epg AEPg404
        bridge-domain member T1BD1
        inherit-from-epg application AP1 epg AEPg403
        exit
      epg uSeg1_403_10 type micro-segmented
        bridge-domain member T1BD1
        contract provider T1Ctr1_uSeg_l3out
        attribute-logical-expression 'ip equals 192.168.103.10 force'
        exit
      epg uSeg1_403_30 type micro-segmented
        bridge-domain member T1BD1
        attribute-logical-expression 'ip equals 192.168.103.30 force'
        inherit-from-epg application AP1 epg uSeg1_403_10
        exit
      exit
    exit
```

Configuring L2Out EPG Contract Inheritance Using the NX-OS Style CLI

To configure contract inheritance for an external L2Out EPG, use the following commands:

Before you begin

Configure the tenant, VRF, and bridge-domain to be used by the EPGs.

Configure the Layer 2 outside network (L2Out) that the EPGs will use.

Configure the contracts to be shared by the EPGs, at the VRF level.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters configuration mode.
Step 2	**tenant** *tenant-name* **Example:** `apic1(config)# tenant Tn1`	Creates or specifies the tenant to be configured; and enters into tenant configuration mode.
Step 3	**external-l2 epg** *external-l2-epg-name* **Example:** `apic1(config-tenant)# external-l2 epg l2out1:l2Ext1`	Configures or specifies an external L2Out EPG. In this example, this is the L2out contract master.
Step 4	**bridge-domain member** *bd-name* **Example:** `apic1(config-tenant-l2ext-epg)# bridge-domain member T1BD1`	Associates the L2Out EPG with a bridge domain.
Step 5	**contract provider** *contract-name* [**label** *label*] **Example:** `apic1(config-tenant-l2ext-epg)# contract provider T1ctr_tcp`	Adds a contract to be provided by this EPG.
Step 6	**exit** **Example:** `apic1(config-tenant-l2ext-epg)# exit`	Exits the configuration mode
Step 7	**external-l2 epg** *external-l2-epg-name* **Example:** `apic1(config-tenant)# external-l2 epg L2out12:l2Ext12`	Configures an external L2Out EPG. In this example, this is the EPG that inherits contracts from the L2out contract master.
Step 8	**bridge-domain member** *bd-name* **Example:** `apic1(config-tenant-l2ext-epg)# bridge-domain member T1BD1`	Associates the L2out EPG with the bridge domain.
Step 9	**inherit-from-epg** *L2Out-contract-master-name* **Example:** `apic1(config-tenant-l2ext-epg)# inherit-from-epg epg l2out1:l2Ext1`	Configures this EPG to inherit contracts from the L2Out contract master.

	Command or Action	Purpose
Step 10	**exit** **Example:** `apic1(config-tenant-l2ext-epg)# exit`	Exits the configuration mode

Example

The steps above are taken from the following example:

```
apic1# show running-config tenant Tn1 external-l2
# Command: show running-config tenant Tn1 external-l2
# Time: Thu May 11 13:10:14 2017
  tenant Tn1
    external-l2 epg l2out1:l2Ext1
      bridge-domain member T1BD1
      contract provider T1ctr_tcp
      exit
    external-l2 epg l2out10:l2Ext10
      bridge-domain member T1BD10
      contract provider T1ctr_tcp
      exit
    external-l2 epg l2out11:l2Ext11
      bridge-domain member T1BD11
      contract provider T1ctr_udp
      exit
    external-l2 epg l2out12:l2Ext12
      bridge-domain member T1BD12
      inherit-from-epg epg l2out1:l2Ext1
      inherit-from-epg epg l2out10:l2Ext10
      inherit-from-epg epg l2out11:l2Ext11
      inherit-from-epg epg l2out2:l2Ext2
      inherit-from-epg epg l2out3:l2Ext3
      inherit-from-epg epg l2out4:l2Ext4
      inherit-from-epg epg l2out5:l2Ext5
      inherit-from-epg epg l2out6:l2Ext6
      inherit-from-epg epg l2out7:l2Ext7
      inherit-from-epg epg l2out8:l2Ext8
      inherit-from-epg epg l2out9:l2Ext9
      exit
    external-l2 epg l2out2:l2Ext2
      bridge-domain member T1BD2
      contract provider T1ctr_tcp
      exit
    external-l2 epg l2out3:l2Ext3
      bridge-domain member T1BD3
      contract provider T1ctr_tcp
      exit
    external-l2 epg l2out4:l2Ext4
      bridge-domain member T1BD4
      contract provider T1ctr_tcp
      exit
    external-l2 epg l2out5:l2Ext5
      bridge-domain member T1BD5
      contract provider T1ctr_tcp
      exit
    external-l2 epg l2out6:l2Ext6
      bridge-domain member T1BD6
      contract provider T1ctr_tcp
      exit
```

```
external-l2 epg l2out7:l2Ext7
  bridge-domain member T1BD7
  contract provider T1ctr_tcp
  exit
external-l2 epg l2out8:l2Ext8
  bridge-domain member T1BD8
  contract provider T1ctr_tcp
  exit
external-l2 epg l2out9:l2Ext9
  bridge-domain member T1BD9
  contract provider T1ctr_tcp
  exit
exit
```

Configuring External L3Out EPG Contract Inheritance Using the NX-OS Style CLI

To configure contract inheritance for an external L3Out EPG, use the following commands:

Before you begin

Configure the tenant, VRF, and bridge-domain to be used by the EPGs.

Configure the Layer 3 outside network (L3Out) that the EPGs will use.

Configure the contracts to be shared by the EPGs, at the VRF level.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters configuration mode.
Step 2	**tenant** *tenant-name* **Example:** `apic1(config)# tenant Tn1`	Creates or specifies the tenant to be configured; and enters into tenant configuration mode.
Step 3	**external-l3 epg** *external-l3-epg-name* **l3out** *l3out-name* **Example:** `apic1(config-tenant-app)# external-l3 epg l3Ext108 l3out T1L3out1`	Configures an external L3Out EPG. In this example, this is the L3out contract master.
Step 4	**vrf member** *vrf-name* **Example:** `apic1(tenant-l3out)# vrf member T1ctx1`	Associates the L3out with the VRF.
Step 5	**match ip** *ip-address-and-mask* **Example:** `apic1(config-tenant-l3ext-epg)# match ip 192.168.110.0/24 shared`	Adds a subnet that identifies hosts as part of the EPG and adds the optional shared scope for the subnet.

	Command or Action	Purpose
Step 6	**contract provider** *contract-name* [**label** *label*] **Example:** `apic1(config-tenant-l3ext-epg)# contract provider T1ctrl-L3out`	Adds a contract to be provided by this EPG.
Step 7	**exit** **Example:** `apic1(config-tenant-l3ext-epg)# exit`	Exits the configuration mode
Step 8	**external-l3 epg** *external-l3-epg-name* **l3out** *l3out-name* **Example:** `apic1(config-tenant-app)# external-l3 epg l3Ext110 l3out T1L3out1`	Configures an external L3Out EPG. In this example, this is the EPG that inherits contracts from the L3out contract master.
Step 9	**vrf member** *vrf-name* **Example:** `apic1(tenant-l3out)# vrf member T1ctx1`	Associates the L3out with the VRF.
Step 10	**match ip** *ip-address-and-mask* **Example:** `apic1(config-tenant-l3ext-epg)# match ip 192.168.112.0/24 shared`	Adds a subnet that identifies hosts as part of the EPG and adds the optional shared scope for the subnet.
Step 11	**inherit-from-epg** *L3Out-contract-master-name* **Example:** `apic1(config-tenant-l3ext-epg)# inherit-from-epg l3Ext108`	Configures this EPG to inherit contracts from the L3Out contract master.
Step 12	**exit** **Example:** `apic1(config-tenant-l3ext-epg)# exit`	Exits the configuration mode

Example

```
ifav90-ifc1# show running-config tenant Tn1 external-l3 epg l3Ext110
# Command: show running-config tenant Tn1 external-l3 epg l3Ext110
# Time: Fri Apr 28 17:36:15 2017
  tenant Tn1
    external-l3 epg l3Ext108 l3out T1L3out1
      vrf member T1ctx1
      match ip 192.168.110.0/24 shared
      contract provider T1ctrl-L3out
      exit
    external-l3 epg l3Ext110 l3out T1L3out1
      vrf member T1ctx1
      match ip 192.168.112.0/24 shared
```

```
    inherit-from-epg epg l3Ext108
    exit
  exit
```

Configuring EPG Contract Inheritance Using the REST API

Configuring Application EPG Contract Inheritance Using the REST API

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Configure the application EPG, to serve as the **EPG Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

Procedure

To configure contract inheritance using the REST API, send a post with XML such as the following XML and JSON examples, with a URL directed to the EPG that will inherit the contracts:

Example:

XML Example

```
<?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 Example

```
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":[]}}]}}
```

Configuring uSeg EPG Contract Inheritance Using the REST API

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Configure the application EPG, to serve as the **EPG Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

Procedure

To configure uSeg contract inheritance using the REST API, send a post with XML such as the following example:

Example:

```
<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>
```

What to do next

Configuring L2Out EPG Contract Inheritance Using the REST API

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Configure the L2Out EPG, to serve as the **L2Out Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

Procedure

To configure L2Out EPG contract inheritance using the REST API, send a post with XML such as the following example:

Example:

```
<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>
```

Configuring L3Out EPG Contract Inheritance Using the REST API

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Configure the L3Out EPG, to serve as the **L3Out Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

Procedure

To configure L3Out EPG contract inheritance using the REST API, send a post with XML such as the following example:

Example:

```
<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>
```

Contract Preferred Groups

About Contract Preferred Groups

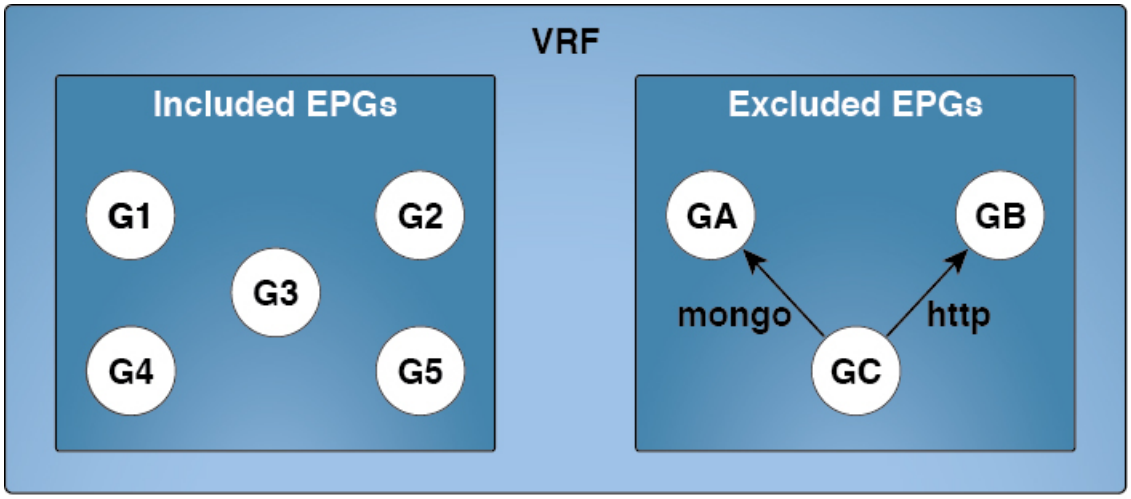
There are two types of policy enforcements available for EPGs in a VRF with a contract preferred group configured:

- Included EPGs: EPGs can freely communicate with each other without contracts, if they have membership in a contract preferred group. This is based on the source-any-destination-any-permit default rule.
- Excluded EPGs: EPGs that are not members of preferred groups require contracts to communicate with each other. Otherwise, the default source-any-destination-any-deny rule applies.

The contract preferred group feature enables greater control of communication between EPGs in a VRF. If most of the EPGs in the VRF should have open communication, but a few should only have limited communication with the other EPGs, you can configure a combination of a contract preferred group and contracts with filters to control inter-EPG communication precisely.

EPGs that are excluded from the preferred group can only communicate with other EPGs if there is a contract in place to override the source-any-destination-any-deny default rule.

Figure 10: Contract Preferred Group Overview



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

Service Graph Support

As of APIC release 4.0(1), EPGs created by service graphs can be included in contract preferred groups. A new policy (Service EPG Policy) is available for defining the preferred group membership type (include or exclude). Once configured, it can be applied through the device selection policy or through the application of a service graph template.

Also, shadow EPGs can now be configured to be included or excluded in preferred groups.

Limitations

The following limitations apply to contract preferred groups:

- In topologies where an L3Out and application EPG are configured in a Contract Preferred Group, and the EPG is deployed only on a VPC, you may find that only one leaf switch in the VPC has the prefix entry for the L3Out. In this situation, the other leaf switch in the VPC does not have the entry, and therefore drops the traffic.

  To workaround this issue, you can do one of the following:

  - Disable and reenable the contract group in the VRF
  - Delete and recreate the prefix entries for the L3Out EPG

- Also, where the provider or consumer EPG in a service graph contract is included in a contract group, the shadow EPG can not be excluded from the contract group. The shadow EPG will be permitted in the contract group, but it does not trigger contract group policy deployment on the node where the shadow EPG is deployed. To download the contract group policy to the node, you deploy a dummy EPG within the contract group .
- Due to CSCvm63145, an EPG in a Contract Preferred Group can consume a shared service contract, but cannot be a provider for a shared service contract with an L3Out EPG as consumer.

Guidelines for Contract Preferred Groups

When configuring contract preferred groups, refer to the following guidelines:

- If the (s, g) entry is installed on a border leaf switch, you might see drops in unicast traffic that comes from the fabric to this source outside the fabric when the following conditions are met:
  - Preferred group is used on the L3Out EPG
  - Unicast routing table for the source is using the default route 0.0.0.0/0

  This behavior is expected.
- Contract Preferred Group-included EPGs are not supported with a 0/0 prefix in external EPG (InstP). If, for the external EPG (InstP) to Tenant EPG, a 0/0 prefix is required with the use of Contract Preferred Group, then 0/0 can be split to 0/1 and 128/1.
- Contract Preferred Group-EPGs are not supported with the GOLF feature. Communication between an application EPG and the L3Out EPG for GOLF must be governed by explicit contracts.

Configuring Contract Preferred Groups Using the GUI

Before you begin

Create the tenants and VRF, and EPGs that will consume the contract preferred group.

Procedure

Step 1 On the menu bar, click **Tenants** > ***tenant_name***.

Step 2 In the **Navigation** pane, expand the tenant, **Networking**, and **VRFs**.

Step 3 Click the VRF name for which you are configuring the contract preferred group.

Step 4 In the **Preferred Group Member** field, click **Enabled**.

Step 5 Click **Submit**.

Step 6 In the **Navigation** pane, expand **Application Profiles** and create or expand an application profile for the tenant VRF.

Step 7 Expand **Application EPGs** and click the EPG that will consume the contract preferred group.

Step 8 Select the **Policy** and **General** tab.

Step 9 In the **Preferred Group Member** field, click **Include**.

Step 10 Click **Submit**.

What to do next

Enable membership in the preferred group for other EPGs that should have unlimited communication with this EPG. You can also configure appropriate contracts to control communication between the EPGs in the preferred group and other EPGs that may not be members.

Note If you want to support preferred group members through L4-L7 service graphs, you must create a L4-L7 service EPG policy. For more information regarding creating an L4-L7 Service EPG Policy, see Creating an L4-L7 Service EPG Policy Using the GUI, on page 86.

Configuring Contract Preferred Groups Using the NX-OS Style CLI

You can use the APIC NX-OS style CLI to configure a contract preferred group. In this example, a contract preferred group is configured for a VRF. One of the EPGs using the VRF is included in the preferred group.

Before you begin

Create the tenants, VRFs, and EPGs that will consume the contract preferred group.

Procedure

	Command or Action	Purpose

Step 1	configure
	Example:
	`apic1# configure`
	`apic1(config)#`
		Enters configuration mode

Step 2	**tenant***tenant-name* **Example:** `apic1(config)# tenant tenant64`	Creates a tenant or enters into tenant configuration mode
Step 3	**vrf context** *vrf-name* **Example:** `apic1(config-tenant)# vrf context vrf64`	Creates a VRF or enters into VRF configuration mode.

Step 4	**whitelist-blacklist-mix**
	Example:
	`apic1(config-tenant-vrf)# whitelist-blacklist-mix`
	`apic1(config-tenant-vrf)# exit`
		Enables a contract preferred group for the VRF and then returns to tenant configuration mode.

Step 5	**bridge-domain** *bd-name* **Example:**	Creates a bridge-domain for the VRF or enters into BD configuration mode.

	Command or Action	Purpose
	`apic1(config-tenant)# bridge-domain bd64`	

Step 6	**vrf member** *vrf-name*
	Example:
	`apic1(config-tenant-bd)# vrf member vrf64`
	`apic1(config-tenant-bd)# exit`
		Associates the VRF with the bridge-domain and returns to teanant configuration mode.

Step 7	**application** *app-name* **Example:** `apic1(config-tenant)# application app-ldap`	Creates an application or enters into application configuration mode.
Step 8	**epg** *epg-name* **Example:** `apic1(config-tenant-app)# epg epg-ldap`	Creates an EPG or enters into EPG tenant-app EPG configuration mode.
Step 9	**bridge-domain member** *bd-name* **Example:** `apic1(config-tenant-app-epg)# bridge-domain member bd64`	Associates the EPG with the bridge-domain .
Step 10	**vrf-blacklist-mode** **Example:** `apic1(config-tenant-app-epg)# vrf-blacklist-mode`	Configures this EPG to be included in the contract preferred group.

Example

The following example creates a contract preferred group for `vrf64` and includes `epg-ldap` in it.

```
apic1# configure
apic1(config)# tenant tenant64
apic1(config-tenant)# vrf context vrf64
apic1(config-tenant-vrf)# whitelist-blacklist-mix
apic1(config-tenant-vrf)# exit

apic1(config-tenant)# bridge-domain bd64
apic1(config-tenant-bd)# vrf member vrf64
apic1(config-tenant-bd)# exit

apic1(config-tenant)# application app-ldap
apic1(config-tenant-app)# epg epg-ldap
apic1(config-tenant-app-epg)# bridge-domain member bd64
apic1(config-tenant-app-epg)# vrf-blacklist-mode
```

Configuring Contract Preferred Groups Using the REST API

The following example creates a contract preferred group in `vrf64`, and creates three EPGs in the VRF:

- `epg-ldap`—Included in the preferred group
- `mail`—Included in the preferred group
- `radius`—Excluded from the preferred group

Before you begin

Create the tenants, VRFs, and the EPGs in the VRF.

Procedure

Create a contract preferred group by sending a post, with XML such as the example:

Example:

```
<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>
```

What to do next

Create a contract governing the communication of the `radius` EPG with other EPGs.

Creating an L4-L7 Service EPG Policy Using the GUI

This task creates a policy that defines if EPGs are to be included in, or excluded from, a preferred group. Preferred groups membership allows endpoints to communicate with each other without requiring a contract. After the policy is created, it can be selected during the application of a service graph template to the EPGs.

Before you begin

You must have configured a tenant.

Procedure

Step 1 On the Menu bar, choose **Tenants** > ***tenant_name*** .

Step 2 In the Navigation pane, choose **Policies** > **Protocol** > **L4-L7 Service EPG Policy**.

Step 3 In the Navigation pane, right-click **L4-L7 Service EPG Policy** and choose **Create L4-L7 Service EPG Policy**.

The Create L4-L7 Service EPG Policy dialog box appears.

Step 4 Enter a unique name for the policy in the **Name** field.

Step 5 Optional. Enter a description of the policy in the **Description** field.

Step 6 Choose whether to exclude or include EPGs as preferred members in the **Preferred Group Member** field.

Step 7 Click **Submit**.

The newly created policy appears in the L4-L7 Service EPG Policy Work pane list. To edit a policy in the Work pane, double-click the list line containing the policy.

What to do next

The new L4-L7 service EPG policy can now be selected in a service graph template when applying the graph to EPGs. Refer to Applying a Service Graph Template to Endpoint Groups Using the GUI in the Using the GUI chapter of the *Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*.

Contracts with Permit and Deny Rules

About Contracts with Permit and Deny Rules

Starting with the Cisco Application Policy Infrastructure Controller (Cisco APIC) release 3.2, You can configure contracts with both permit and deny actions, instead of just permit. You can configure the deny action with different priorities: default, highest, medium and lowest.

Rule conflicts are resolved as follows:

- The implicit deny has the lowest priority of all rules.
- Contracts between vzAny have higher priority than the implicit deny.
- Contracts between specific EPG pairs win over contracts with vzAny, because EPG-to-EPG contract rules have higher priority than vzAny-to-vzAny rules.
- Deny rules with the default priority for a contract between a specific EPG pair have the same level of priority as the permit rules for that EPG pair. When traffic matches both a permit and a deny rule with the same priority, the deny rule wins.

- Deny rules with the default priority for a contract between vzAny has the same level of priority as the permit rules for the vzAny pair. When traffic matches both a permit and a deny rule with the same priority, the deny rule wins.
- Deny rules with the highest priority are handled at the same level as EPG-to-EPG contracts.
- Deny rules with medium priority are handled at the same level as vzAny-to-EPG contracts.
- Deny rules with the lowest priority are handled at the same level as vzAny-to-vzAny contracts.
- If the deny priority is lowered in a contract between EPGs, a permit rule match between EPGs would win over deny.