Configuring Tenant Policies

Basic Tenant Configuration

Creating a Tenant, VRF, and Bridge Domain Using the REST API

SUMMARY STEPS

1. Create a tenant.
2. Create a VRF and bridge domain.

DETAILED STEPS

Step 1 Create a tenant.

Example:

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

When the POST succeeds, you see the object that you created in the output.

Step 2 Create a VRF and bridge domain.

Note The Gateway Address can be an IPv4 or an IPv6 address. For more about details IPv6 gateway address, see the related KB article, *KB: Creating a Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery* .

Example:

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

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

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

Tenants in Multiple Private Networks

About Multiple Private Networks with Inter-Tenant Communication

- This use case may be typical for environments where an ACI administrator wishes to create multiple tenants with the ability to support inter-tenant communications.

  This method has the following advantages and disadvantages:

  Advantages:

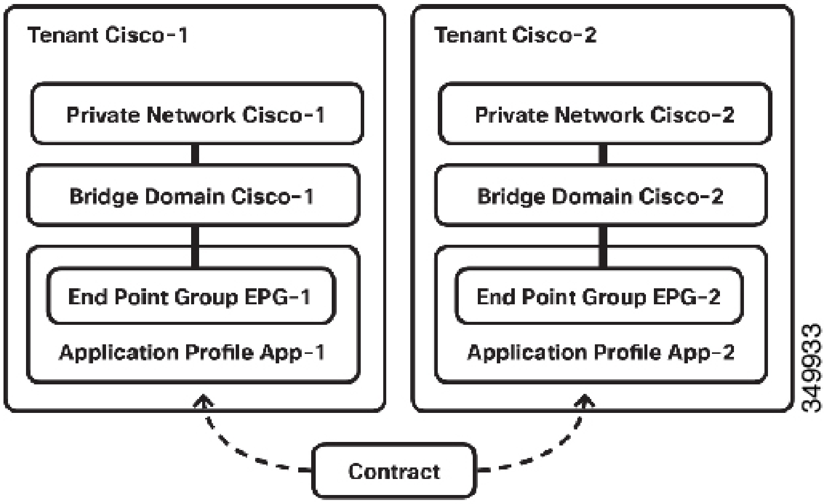
  - Each tenant container can be managed separately
  - Allows for maximum isolation between tenants

  Disadvantages:

  - Tenant address space must be unique

  From a containment and relationship perspective, this topology looks as follows:

Figure 1: Multiple Private Networks with Inter-Tenant Communication



Configuring Multiple Private Networks with Inter-Tenant Communication Using the REST API

Configure the Cisco-1 and Cisco-2 private networks, with communication between them, using the REST API in the following steps:

Step 1 Configure Cisco-1 tenant using the following XML posted to the APIC REST API:

Example:

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

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

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

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

<fvBD arpFlood="yes" mac="00:22:BD:F8:19:FF" name="CiscoBD2" unicastRoute="yes"
unkMacUcastAct="flood" unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx2"/>
</fvBD>
```

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

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

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

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

Step 2 Configure Cisco-2 tenanat using the following XML posted to the APIC REST API:

Example:

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

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

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

About Multiple Private Networks with Intra-Tenant Communication

Another use case that may be desirable to support is the option to have a single tenant with multiple private networks. This may be a result of needing to provide multitenancy at a network level, but not at a management level. It may also be caused by needing to support overlapping subnets within a single tenant, due to mergers and acquisitions or other business changes.

This method has the following advantages and disadvantages:

Advantages:

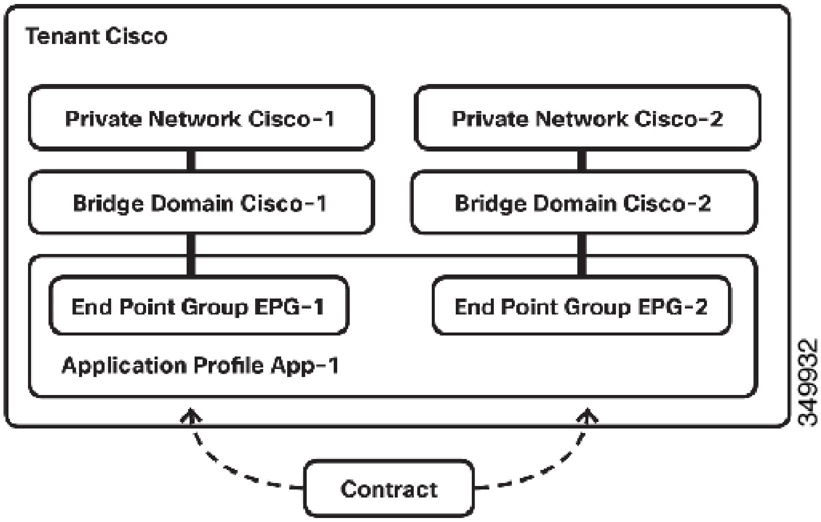
- Ability to have overlapping subnets within a single tenant

Disadvantages:

- EPGs residing in overlapping subnets cannot have policy applied between one another

The object containment for this particular setup can be depicted as shown below:

Figure 2: Multiple Private Networks with Intra-Tenant Communication



Configuring Multiple Tenants with Intra-Tenant Communication Using the REST API

SUMMARY STEPS

1. Configure the Tenant Cisco, with Cisco-1 and Cisco-2 networks, using the following XML posted to the APIC REST API:

DETAILED STEPS

Configure the Tenant Cisco, with Cisco-1 and Cisco-2 networks, using the following XML posted to the APIC REST API:

Example:

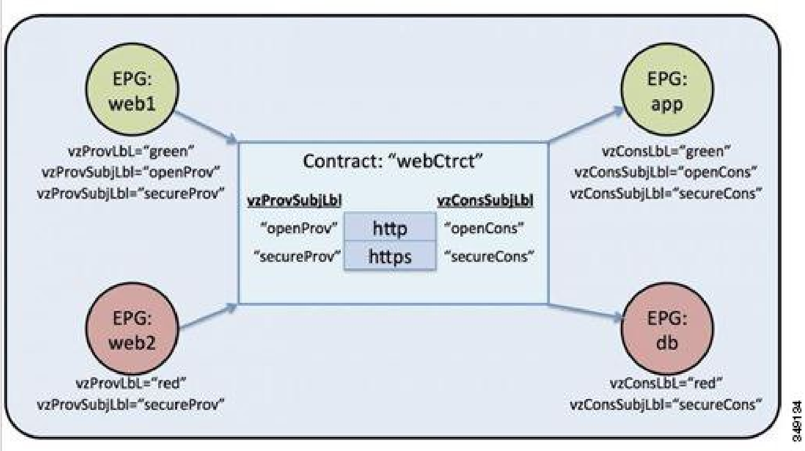
```
<fvTenant dn="uni/tn-Cisco" name="Cisco">
<vzBrCP name="ICMP" scope="tenant">
<vzSubj consMatchT="AtleastOne" name="icmp" provMatchT="AtleastOne"
revFltPorts="yes">
<vzRsSubjFiltAtt tnVzFilterName="icmp"/>
</vzSubj>
</vzBrCP>
<fvCtx knwMcastAct="permit" name="CiscoCtx" pcEnfPref="enforced"/>
<fvCtx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

Tenant Policy Example

Tenant Policy Example Overview

The description of the tenant policy example in this appendix uses XML terminology (http://en.wikipedia.org/wiki/XML#Key_terminology). This example demonstrates how basic APIC policy model constructs are rendered into the XML code. The following figure provides an overview of the tenant policy example.

Figure 3: EPGs and Contract Contained in Tenant Solar



In the figure, according to the contract called webCtrct and the EPG labels, the green-labeled EPG:web1 can communicate with green-labeled EPG:app using both http and https, the red -abeled EPG:web2 can communicate with the red-labeled EPG:db using only https.

Tenant Policy Example XML Code

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

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

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

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

      <vzBrCP name="webCtrct">
          <vzSubj name="http" revFltPorts="true" provmatchT="All">
              <vzRsSubjFiltAtt tnVzFilterName="Http"/>
              <vzRsSubjGraphAtt graphName="G1" termNodeName="TProv"/>
              <vzProvSubjLbl name="openProv"/>
              <vzConsSubjLbl name="openCons"/>
          </vzSubj>
          <vzSubj name="https" revFltPorts="true" provmatchT="All">
              <vzProvSubjLbl name="secureProv"/>
              <vzConsSubjLbl name="secureCons"/>
              < vzRsSubjFiltAtt tnVzFilterName="Https"/>
              <vzRsOutTermGraphAtt graphName="G2" termNodeName="TProv"/>
          </vzSubj>
      </vzBrCP>
      <fvCtx name="solarctx1"/>

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

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

Tenant Policy Example Explanation

This section contains a detailed explanation of the tenant policy example.

Policy Universe

The policy universe contains all the tenant-managed objects where the policy for each tenant is defined.

```
<polUni>
```

This starting tag, `<polUni>`, in the first line indicates the beginning of the policy universe element. This tag is matched with `</polUni>` at the end of the policy. Everything in between is the policy definition.

Tenant Policy Example

The `<fvTenant>` tag identifies the beginning of the tenant element.

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

All of the policies for this tenant are defined in this element. The name of the tenant in this example is solar. The tenant name must be unique in the system. The primary elements that the tenant contains are filters, contracts, outside networks, bridge domains, and application profiles that contain EPGs.

Filters

The filter element starts with a `<vzFilter>` tag and contains elements that are indicated with a `<vzEntry>` tag.

The following example defines "Http" and "Https" filters. The first attribute of the filter is its name and the value of the name attribute is a string that is unique to the tenant. These names can be reused in different tenants. These filters are used in the subject elements within contracts later on in the example.

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

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

This example defines these two filters: Http and Https. The first attribute of the filter is its name and the value of the name attribute is a string that is unique to the tenant, i.e. these names can be reused in different tenants. These filters will be used in the subject elements within contracts later on in the example.

Each filter can have one or more entries where each entry describes a set of Layer 4 TCP or UDP port numbers. Some of the possible attributes of the `<vzEntry>` element are as follows:

- `name`
- `prot`
- `dFromPort`
- `dToPort`
- `sFromPort`
- `sToPort`
- `etherT`
- `ipFlags`
- `arpOpc`

- tcpRules

In this example, each entry's name attribute is specified. The name is an ASCII string that must be unique within the filter but can be reused in other filters. Because this example does not refer to a specific entry later on, it is given a simple name of "e1".

The EtherType attribute, `etherT`, is next. It is assigned the value of ipv4 to specify that the filter is for IPv4 packets. There are many other possible values for this attribute. Common ones include `ARP`, `RARP`, and`IPv6`. The default is `unspecified` so it is important to assign it a value.

Following the EtherType attribute is the `prot` attribute that is set to `tcp` to indicate that this filter is for TCP traffic. Alternate protocol attributes include `udp`, `icmp`, and `unspecified` (default).

After the protocol, the destination TCP port number is assigned to be in the range from 80 to 80 (exactly TCP port 80) with the `dFromPort` and `dToPort` attributes. If the from and to are different, they specify a range of port numbers.

In this example, these destination port numbers are specified with the attributes `dFromPort` and `dToPort`. However, when they are used in the contract, they should be used for the destination port from the TCP client to the server and as the source port for the return traffic. See the attribute `revFltPorts` later in this example for more information.

The second filter does essentially the same thing, but for port 443 instead.

Filters are referred to by subjects within contracts by their target distinguished name, `tDn`. The `tDn` name is constructed as follows:

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

For example, the `tDn` of the first filter above is `uni/tn-coke/flt-Http`. The second filter has the name `uni/tn-coke/flt-Https`. In both cases, `solar` comes from the tenant name.

Contracts

The contract element is tagged `vzBrCP` and it has a `name` attribute.

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

Contracts are the policy elements between EPGs. They contain all of the filters that are applied between EPGs that produce and consume the contract. The contract element is tagged `vzBrCP` and it has a name attribute. Refer to the object model reference documentation for other attributes that can be used in the contract element. This example has one contract named `webCtrct`.

The contract contains multiple subject elements where each subject contains a set of filters. In this example, the two subjects are `http` and `https`.

The contract is later referenced by EPGs that either provide or consume it. They reference it by its name in in the following manner:

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

`tenant-name` is the name of the tenant, “solar” in this example, and the contract-name is the name of the contract. For this example, the `tDn` name of the contract is `uni/tn-solar/brc-webCtrct`.

Subjects

The subject element starts with the tag `vzSubj` and has three attributes: `name`, `revFltPorts`, and `matchT`. The `name` is simply the ASCII name of the subject.

`revFltPorts` is a flag that indicates that the Layer 4 source and destination ports in the filters of this subject should be used as specified in the filter description in the forward direction (that is, in the direction of from consumer to producer EPG), and should be used in the opposite manner for the reverse direction. In this example, the “http” subject contains the “Http” filter that defined TCP destination port 80 and did not specify the source port. Because the `revFltPorts` flag is set to true, the policy will be TCP **destination port 80** and any source port for traffic from the consumer to the producer, and it will be TCP destination port any and **source port 80** for traffic from the producer to the consumer. The assumption is that the consumer initiates the TCP connection to the producer (the consumer is the client and the producer is the server).

The default value for the `revFltPrts` attribute is `false` if it is not specified.

Labels

The match type attribute, `provmatchT` (for provider matching) and `consmatchT` (for consumer matching) determines how the subject labels are compared to determine if the subject applies for a given pair of consumers and producers. The following match type values are available:

- `All`
- `AtLeastOne` (default)
- `None`
- `ExactlyOne`

When deciding whether a subject applies to the traffic between a producer and consumer EPG, the match attribute determines how the subject labels that are defined (or not) in those EPGs should be compared to the labels in the subject. If the match attribute value is set to `All`, it only applies to the providers whose provider subject labels, `vzProvSubjLbl`, match all of the `vzProvSubjLbl` labels that are defined in the subject. If two labels are defined, both must also be in the provider. If a provider EPG has 10 labels, as long as all of the provider labels in the subject are present, a match is confirmed. A similar criteria is used for the consumers that use the `vzConsSubjLbl`. If the `matchT` attribute value is `AtLeastOne`, only one of the labels must match. If the `matchT` attribute is `None`, the match only occurs if none of the provider labels in the subject match the provider labels of the provider EPGs and similarly for the consumer.

If the producer or consumer EPGs do not have any subject labels and the subject does not have any labels, a match occurs for `All`, `AtLeastOne`, and `None` (if you do not use labels, the subject is used and the `matchT` attribute does not matter).

An optional attribute of the subject not shown in the example is `prio` where the priority of the traffic that matches the filter is specified. Possible values are `gold`, `silver`, `bronze`, or `unspecified` (default).

In the example, the subject element contains references to filter elements, subject label elements, and graph elements. `<vzRsSubjFiltAtt tDn=“uni/tn-coke/flt-Http”/>` is a reference to a previously defined filter. This element is identified by the `vzRsSubjFiltAtt` tag.

`<vzRsSubjGraphAtt graphName=“G1” termNodeName=“TProv”/>` defines a terminal connection.

`<vzProvSubjLbl name=“openProv”/>` defines a provider label named “openProv”. The label is used to qualify or filter which subjects get applied to which EPGs. This particular one is a provider label and the corresponding consumer label is identified by the tag `vzConsSubjLbl`. These labels are matched with the corresponding label of the provider or consumer EPG that is associated with the current contract. If a match occurs according to the `matchT` criteria described above, a particular subject applies to the EPG. If no match occurs, the subject is ignored.

Multiple provider and consumer subject labels can be added to a subject to allow more complicated matching criteria. In this example, there is just one label of each type on each subject. However, the labels on the first subject are different from the labels on the second subject, which allows these two subjects to be handled differently depending on the labels of the corresponding EPGs. The order of the elements within the subject element does not matter.

VRF

The Virtual Routing and Forwarding (VRF) (also known as a context or private network) is identified by the `fvCtx` tag and contains a name attribute.

A tenant can contain multiple VRFs. For this example, the tenant uses one VRF named “solartx1”. The name must be unique within the tenant.

The VRF defines a Layer 3 address domain. All of the endpoints within the Layer 3 domain must have unique IPv4 or IPv6 addresses, because it is possible to directly forward packets between these devices if the policy allows it.

Although a VRF defines a unique IP address space, the corresponding subnets are defined within bridge domains. Each bridge domain is then associated with the VRF.

Bridge Domains

The bridge domain element is identified with the `fvBD` tag and has a name attribute.

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

Within the bridge domain element, subnets are defined and a reference is made to the corresponding Virtual Routing and Forwarding (VRF) instance (also known as a context or private network). Each bridge domain must be linked to a VRF and have at least one subnet.

This example uses one bridge domain named “solarBD1”. In this example, the “solarctx1” VRF is referenced by using the element tagged `fvRsCtx` and the `tnFvCtxName` attribute is given the value “solarctx1”. This name comes from the VRF defined above.

The subnets are contained within the bridge domain and a bridge domain can contain multiple subnets. This example defines two subnets. All of the addresses used within a bridge domain must fall into one of the address ranges that is defined by the subnets. However, the subnet can also be a supernet which is a very large subnet that includes many addresses that might never be used. Specifying one giant subnet that covers all current future addresses can simplify the bridge domain specification. However, different subnets must not overlap within a bridge domain or with subnets defined in other bridge domains that are associated with the same VRF. Subnets can overlap with other subnets that are associated with other VRFs.

The subnets described above are 11.22.22.xx/24 and 11.22.23.xx/24. However, the full 32 bits of the address is given even though the mask says that only 24 are used, because this IP attribute also identifies the full IP address for the router in that subnet. In the first case, the router IP address (default gateway) is 11.22.22.20 and for the second subnet, it is 11.22.23.211.

The entry 11.22.22.20/24 is equivalent to the following, but in compact form:

- Subnet: 11.22.22.00
- Subnet Mask: 255.255.255.0
- Default gateway: 11.22.22.20

Application Profiles

The start of the application profile is indicated by the `fvAp` tag and has a name attribute.

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

This example has one application network profile and it is named “sap.”

The application profile is a container that holds the EPGs. EPGs can communicate with other EPGs in the same application profile and with EPGs in other application profiles. The application profile is simply a convenient container that is used to hold multiple EPGs that are logically related to one another. They can be organized by the application they provide such as “sap,” by the function they provide such as “infrastructure,” by where they are in the structure of the data center such as “DMZ,” or whatever organizing principle the administrator chooses to use.

The primary object that the application profile contains is an endpoint group (EPG). In this example, the “sap” application profile contains 4 EPGs: web1, web2, app, and db.

Endpoints and Endpoint Groups (EPGs)

EPGs begin with the tag `fvAEPg` and have a name attribute.

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

The EPG is the most important fundamental object in the policy model. It represents a collection of endpoints that are treated in the same fashion from a policy perspective. Rather than configure and manage those endpoints individually, they are placed within an EPG and are managed as a collection or group.

The EPG object is where labels are defined that govern what policies are applied and which other EPGs can communicate with this EPG. It also contains a reference to the bridge domain that the endpoints within the EPG are associated with as well as which virtual machine manager (VMM) domain they are associated with. VMM allows virtual machine mobility between two VM servers instantaneously with no application downtime.

The first EPG in the example is named "web1." The `fvRsBd` element within the EPG defines which bridge domain that it is associated with. The bridge domain is identified by the value of the `tnFxBDName` attribute. This EPG is associated with the "solarBD1" bridge domain named in the "Bridge Domain" section above. The binding to the bridge domain is used by the system to understand what the default gateway address should be for the endpoints in this EPG. It does not imply that the endpoints are all in the same subnet or that they can only communicate through bridging. Whether an endpoint's packets are bridged or routed is determined by whether the source endpoint sends the packet to its default gateway or the final destination desired. If it sends the packet to the default gateway, the packet is routed.

The VMM domain used by this EPG is identified by the `fvRsDomAtt` tag. This element references the VMM domain object defined elsewhere. The VMM domain object is identified by its `tDn` name attribute. This example shows only one VMM domain called "uni/vmmp-VMware/dom-mininet."

The next element in the "web1" EPG defines which contract this EPG provides and is identified by the `fvRsProv` tag. If "web1" were to provide multiple contracts, there would be multiple `fvRsProv` elements. Similarly, if it were to consume one or more contracts, there would be `fvRsCons` elements as well.

The `fvRsProv` element has a required attribute that is the name of the contract that is being provided. "web1" is providing the contract "webCtrct" that was defined earlier that was called `tDn="uni/tn-coke/brc-webCtrct"`.

The next attribute is the `matchT` attribute, which has the same semantics for matching provider or consumer labels as it did in the contract for subject labels (it can take on the values of `All`, `AtLeastOne`, or `None`). This criteria applies to the provider labels as they are compared to the corresponding consumer labels. A match of the labels implies that the consumer and provider can communicate if the contract between them allows it. In other words, the contract has to allow communication and the consumer and provider labels have to match using the match criteria specified at the provider.

The consumer has no corresponding match criteria. The match type used is always determined by the provider.

Inside the provider element, `fvRsProv`, an administrator needs to specify the labels that are to be used. There are two kinds of labels, provider labels and provider subject labels. The provider labels, `vzProvLbl`, are used to match consumer labels in other EPGs that use the `matchT` criteria described earlier. The provider subject labels, `vzProvSubjLbl`, are used to match the subject labels that are specified in the contract. The only attribute of the label is its name attribute.

In the "web1" EPG, two provider subject labels, `openProv` and `secureProv`, are specified to match with the "http" and "https" subjects of the "webCtrct" contract. One provider label, "green" is specified with a match criteria of `All` that will match with the same label in the "App" EPG.

The next EPG in the example, "web2," is very similar to "web1" except that there is only one `vzProvSubjLbl` and the labels themselves are different.

The third EPG is one called "app" and it is defined as follows:

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

```
                    <vzConsSubjLbl name="secureCons"/>
                    <vzConsLbl name="green"/>
                 </fvRsCons>
            </fvAEPg>
```

The first part is nearly the same as the "web1" EPG. The major difference is that this EPG is a consumer of the "webCtrct" and has the corresponding consumer labels and consumer subject labels. The syntax is nearly the same except that "Prov" is replaced by "Cons" in the tags. There is no match attribute in the `FvRsCons` element because the match type for matching the provider with consumer labels is specified in the provider.

In the last EPG, "db" is very similar to the "app" EPG in that it is purely a consumer.

While in this example, the EPGs were either consumers or producers of a single contract, it is typical for an EPG to be at once a producer of multiple contracts and a consumer of multiple contracts.

Closing

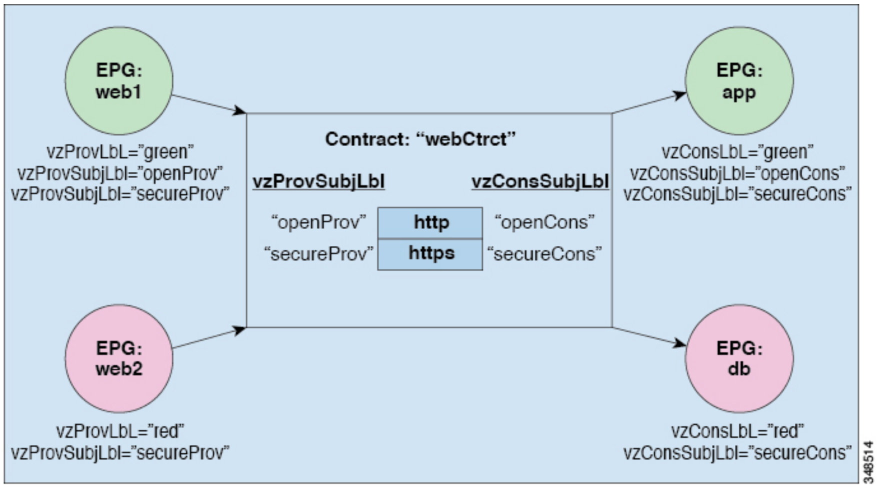
```
        </fvAp>
    </fvTenant>
</polUni>
```

The final few lines complete the policy.

What the Example Tenant Policy Does

The following figure shows how contracts govern endpoint group (EPG) communications.

Figure 4: Labels and Contract Determine EPG to EPG Communications



The four EPGs are named EPG:web1, EPG:web2, EPG:app, and EPG:db. EPG:web1 and EPG:web2 provide a contract called webCtrct. EPG:app and EPG:db consume that same contract.

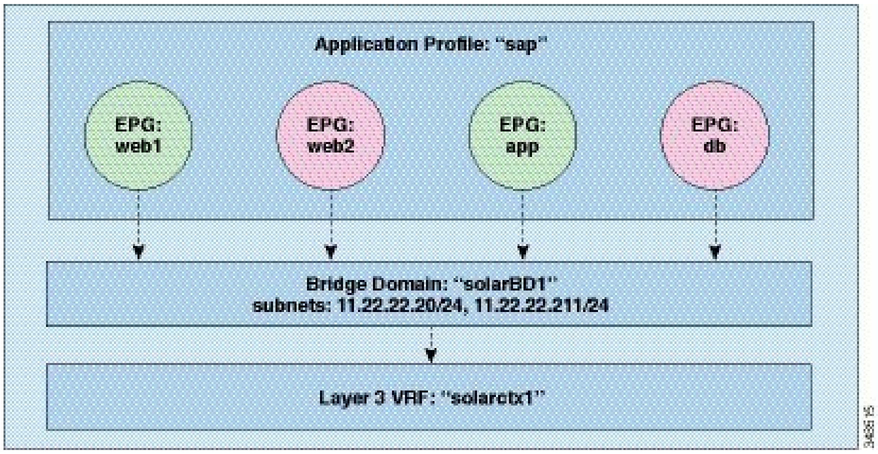
EPG:web1 can only communicate with EPG:app and EPG:web2 can only communicate with EPG:db. This interaction is controlled through the provider and consumer labels "green" and "red".

When EPG:web1 communicates with EPG:app, they use the webCtrct contract. EPG:app can initiate connections to EPG:web1 because it consumes the contract that EPG:web1 provides.

The subjects that EPG:web1 and EPG:app can use to communicate are both http and https because EPG:web1 has the provider subject label "openProv" and the http subject also has it. EPG:web1 has the provider subject label “secureProv” as does the subject https. In a similar fashion, EPG:app has subject labels “openCons” and “secureCons” that subjects http and https have.

When EPG:web2 communicates with EPG:db, they can only use the https subject because only the https subject carries the provider and consumer subject labels. EPG:db can initiate the TCP connection to EPG:web2 because EPG:db consumes the contract provided by EPG:web2.

Figure 5: Bridge Domain, Subnets, and Layer 3 VRF


Application Profile: “sap”
EPG: web1
EPG: web2
EPG: app
EPG: db
Bridge Domain: “solarBD1”
subnets: 11.22.22.20/24, 11.22.22.211/24
Layer 3 VRF: “solarctx1”
348615

The example policy specifies the relationship between EPGs, application profiles, bridge domains, and Layer 3 Virtual Routing and Forwarding (VRF) instances in the following manner: the EPGs EPG:web1, EPG:web2, EPG:app, and EPG:db are all members of the application profile called “sap.”

These EPGs are also linked to the bridge domain “solarBD1.” solarBD1 has two subnets, 11.22.22.XX/24 and 11.22.23.XX/24. The endpoints in the four EPGs must be within these two subnet ranges. The IP address of the default gateway in those two subnets will be 11.22.22.20 and 11.22.23.211. The solarBD1 bridge domain is linked to the “solarctx1” Layer 3 VRF.

All these policy details are contained within a tenant called “solar.”

EPGs

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 on a Specific Port with APIC Using the REST API

Before you begin

The tenant where you deploy the EPG is created.

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>
```

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.

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>
```

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.

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>
```

Intra-EPG Isolation

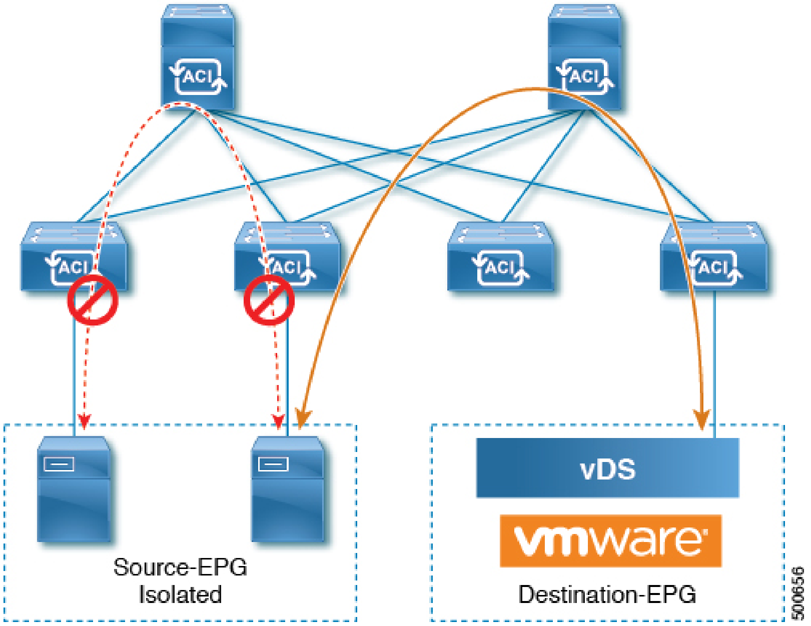
Intra-EPG Isolation for Bare Metal Servers

Intra-EPG endpoint isolation policies can be applied to directly connected endpoints such as bare metal servers.

Examples use cases include the following:

- Backup clients have the same communication requirements for accessing the backup service, buy they don't need to communicate with each other.
- Servers behind a load balancer have the same communication requirements, but isolating them from each other protects against a server that is compromised or infected.

Figure 6: Intra-EPG Isolation for Bare Metal Servers



Bare metal EPG isolation is enforced at the leaf switch. Bare metal servers use VLAN encapsulation. All unicast, multicast and broadcast traffic is dropped (denied) within isolation enforced EPGs. ACI bridge-domains can have a mix of isolated and regular EPGs. Each Isolated EPG can have multiple VLANs where intra-vlan traffic is denied.

Configuring Intra-EPG Isolation for Bare Metal Servers Using the REST API

Before you begin

The port the EPG uses must be associated with a bare metal server interface in the physical domain.

SUMMARY STEPS

1. Send this HTTP POST message to deploy the application using the XML API.
2. Include this XML structure in the body of the POST message.

DETAILED STEPS

Step 1 Send this HTTP POST message to deploy the application using the XML API.

Example:

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

Step 2 Include this XML structure in the body of the POST message.

Example:

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

Intra-EPG Isolation for VMware VDS or Microsoft Hyper-V Virtual Switch

Intra-EPG Isolation is an option to prevent physical or virtual endpoint devices that are in the same base EPG or uSeg EPG from communicating with each other. By default, endpoint devices included in the same EPG are allowed to communicate with one another. However, conditions exist in which total isolation of the endpoint devices from on another within an EPG is desirable. For example, you may want to enforce intra-EPG isolation if the endpoint VMs in the same EPG belong to multiple tenants, or to prevent the possible spread of a virus.

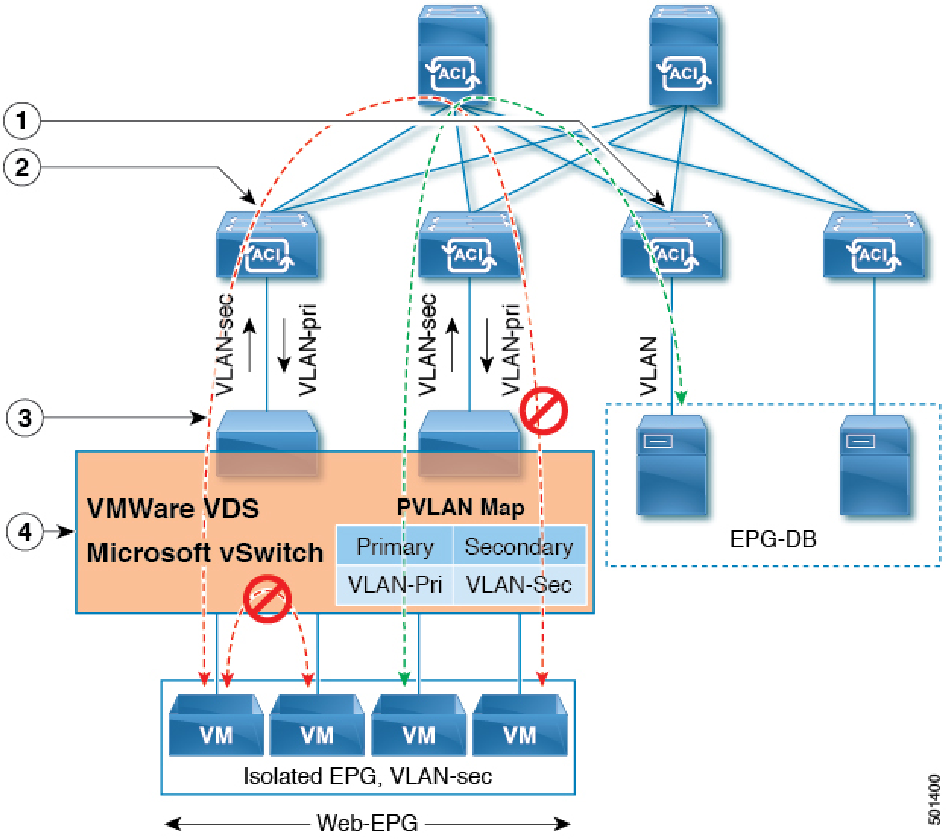
A Cisco ACI virtual machine manager (VMM) domain creates an isolated PVLAN port group at the VMware VDS or Microsoft Hyper-V Virtual Switch for each EPG that has intra-EPG isolation enabled. A fabric administrator specifies primary encapsulation or the fabric dynamically specifies primary encapsulation at the time of EPG-to-VMM domain association. When the fabric administrator selects the VLAN-pri and VLAN-sec values statically, the VMM domain validates that the VLAN-pri and VLAN-sec are part of a static block in the domain pool.

Note When intra-EPG isolation is not enforced, the VLAN-pri value is ignored even if it is specified in the configuration.

VLAN-pri/VLAN-sec pairs for the VMware VDS or Microsoft Hyper-V Virtual Switch are selected per VMM domain during the EPG-to-domain association. The port group created for the intra-EPG isolation EPGs uses the VLAN-sec tagged with type set to `PVLAN`. The VMware VDS or the Microsoft Hyper-V Virtual Switch and fabric swap the VLAN-pri/VLAN-sec encapsulation:

- Communication from the Cisco ACI fabric to the VMware VDS or Microsoft Hyper-V Virtual Switch uses VLAN-pri.
- Communication from the VMware VDS or Microsoft Hyper-V Virtual Switch to the Cisco ACI fabric uses VLAN-sec.

Figure 7: Intra-EPG Isolation for VMware VDS or Microsoft Hyper-V Virtual Switch



Note these details regarding this illustration:

1. EPG-DB sends VLAN traffic to the Cisco ACI leaf switch. The Cisco ACI egress leaf switch encapsulates traffic with a primary VLAN (PVLAN) tag and forwards it to the Web-EPG endpoint.
2. The VMware VDS or Microsoft Hyper-V Virtual Switch sends traffic to the Cisco ACI leaf switch using VLAN-sec. The Cisco ACI leaf switch drops all intra-EPG traffic because isolation is enforced for all intra VLAN-sec traffic within the Web-EPG.
3. The VMware VDS or Microsoft Hyper-V Virtual Switch VLAN-sec uplink to the Cisco ACI Leaf is in isolated trunk mode. The Cisco ACI leaf switch uses VLAN-pri for downlink traffic to the VMware VDS or Microsoft Hyper-V Virtual Switch.
4. The PVLAN map is configured in the VMware VDS or Microsoft Hyper-V Virtual Switch and Cisco ACI leaf switches. VM traffic from WEB-EPG is encapsulated in VLAN-sec. The VMware VDS or Microsoft Hyper-V Virtual Switch denies local intra-WEB EPG VM traffic according to the PVLAN tag. All intra-ESXi host or Microsoft Hyper-V host VM traffic is sent to the Cisco ACI leaf using VLAN-Sec.

Related Topics

For information on configuring intra-EPG isolation in a Cisco AVS environment, see Intra-EPG Isolation Enforcement for Cisco AVS, on page 24.

Configuring Intra-EPG Isolation for VMware VDS or Microsoft Hyper-V Virtual Switch using the REST API

SUMMARY STEPS

1. Send this HTTP POST message to deploy the application using the XML API.
2. For a VMware VDS or Microsoft Hyper-V Virtual Switch deployment, include one of the following XML structures in the body of the POST message.

DETAILED STEPS

Step 1 Send this HTTP POST message to deploy the application using the XML API.

Example:

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

Step 2 For a VMware VDS or Microsoft Hyper-V Virtual Switch deployment, include one of the following XML structures in the body of the POST message.

Example:

The following example is for VMware VDS:

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

Example:

The following example is for Microsoft Hyper-V Virtual Switch:

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

Intra-EPG Isolation Enforcement for Cisco AVS

By default, endpoints with an EPG can communicate with each other without any contracts in place. However, you can isolate endpoints within an EPG from each other. In some instances, you might want to enforce endpoint isolation within an EPG to prevent a VM with a virus or other problem from affecting other VMs in the EPG.

You can configure isolation on all or none of the endpoints within an application EPG; you cannot configure isolation on some endpoints but not on others.

Isolating endpoints within an EPG does not affect any contracts that enable the endpoints to communicate with endpoints in another EPG.

Isolating endpoints within an EPG will trigger a fault when the EPG is associated with Cisco AVS domains in VLAN mode.

Note Using intra-EPG isolation on a Cisco AVS microsegment (uSeg) EPG is not currently supported. Communication is possible between two endpoints that reside in separate uSeg EPGs if either has intra-EPG isolation enforced, regardless of any contract that exists between the two EPGs.

Configuring Intra-EPG Isolation for Cisco AVS Using the REST API

Before you begin

Make sure that Cisco AVS is in VXLAN mode.

Step 1 Send this HTTP POST message to deploy the application using the XML API.

Example:

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

Step 2 For a VMM deployment, include the XML structure in the following example in the body of the POST message.

Example:

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

What to do next

You can select statistics and view them to help diagnose problems involving the endpoint. See the sections "Choosing Statistics to View for Isolated Endpoints" and "Viewing Statistics for Isolated Endpoints" in the *Cisco AVS Configuration Guide* or the *Cisco APIC Layer 2 Configuration Guide*.

Microsegmentation

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 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.

SUMMARY STEPS

1. 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."

DETAILED STEPS

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>
```

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.

SUMMARY STEPS

1. Log in to the Cisco APIC.
2. Post the policy to https://*apic-ip-address*/api/node/mo/.xml.

DETAILED STEPS

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>
```

Configuring Microsegmentation on Virtual Switches

Microsegmentation with the Cisco Application Centric Infrastructure (ACI) provides the ability to automatically assign endpoints to logical security zones called endpoint groups (EPGs) based on various network-based or virtual machine (VM)-based attributes. This section contains instructions for configuring microsegment (uSeg) EPGs on virtual switches.

Microsegmentation with Cisco ACI provides support for virtual endpoints attached to the following:

- VMware vSphere Distributed Switch (VDS)
- Cisco Application Virtual Switch (AVS)
- Microsoft vSwitch

See the *Cisco ACI Virtualization Guide* for information about how Microsegmentation with Cisco ACI works, prerequisites, guidelines, and scenarios.

Configuring Microsegmentation with Cisco ACI Using the REST API

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

Step 1 Log in to the Cisco APIC.

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

Example:

This example configures a uSeg EPG with the attributes VM Name containing "vm" and Operating System attributes containing values of "CentOS" and "Linux," with matching for all attributes and with an EPG Match Precedence of 1.

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

Example:

This example is for an application EPG with microsegmentation enabled.

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

In the example above, the string `<fvRsDomAtt tDn="uni/vmmp-VMware/dom-cli-vmm1" classPref=“useg”/>` is relevant only for VMware VDS and not for Cisco ACI Virtual Edge, Cisco AVS, or Microsoft Hyper-V Virtual Switch.

Example:

This example attaches a uSeg EPG to a Cisco ACI Virtual Edge VMM domain to add the switching mode.

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

Application Profiles

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.

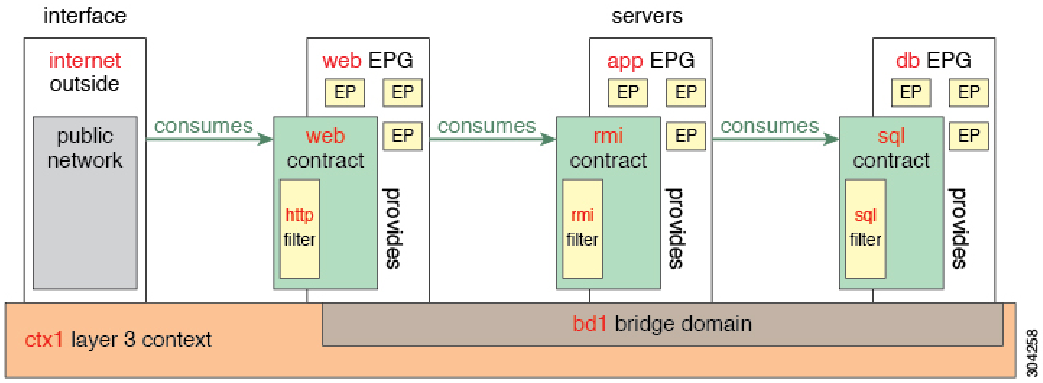
Contracts are policies that enable inter-End Point Group (inter-EPG) communication. These policies are the rules that specify communication between application tiers. If no contract is attached to the EPG, inter-EPG communication is disabled by default. No contract is required for intra-EPG communication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

Parameter Name	Filter for http
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
Destination Port	1099	1521

Deploying an Application Profile Using the REST API

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

SUMMARY STEPS

1. Send this HTTP POST message to deploy the application using the XML API.
2. Include this XML structure in the body of the POST message.

DETAILED STEPS

Step 1 Send this HTTP POST message to deploy the application using the XML API.

Example:

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

Step 2 Include this XML structure in the body of the POST message.

Example:

```
<fvTenant name="ExampleCorp">

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

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

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

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

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

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

In the string **fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"delimiter=@/**, **delimiter=@** is optional. If you do not enter a delimiter, the system will use the default | delimiter.

In the XML structure, the first line modifies, or creates if necessary, the tenant named ExampleCorp.

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

This line creates an application network profile named OnlineStore.

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

The elements within the application network profile create three endpoint groups, one for each of the three servers. The following lines create an endpoint group named web and associate it with an existing bridge domain named bd1. This endpoint group is a consumer, or destination, of the traffic allowed by the binary contract named rmi and is a provider, or source, of the traffic allowed by the binary contract named web. The endpoint group is associated with the VMM domain named datacenter.

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

The remaining two endpoint groups, for the application server and the database server, are created in a similar way.

The following lines define a traffic filter named http that specifies TCP traffic of types HTTP (port 80) and HTTPS (port 443).

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

The remaining two filters, for application data and database (sql) data, are created in a similar way.

The following lines create a binary contract named web that incorporates the filter named http:

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

The remaining two contracts, for rmi and sql data protocols, are created in a similar way.

The final line closes the structure:

```
</fvTenant>
```

Contracts, Taboo Contracts, and Preferred Groups

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 and Taboo Contracts

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.

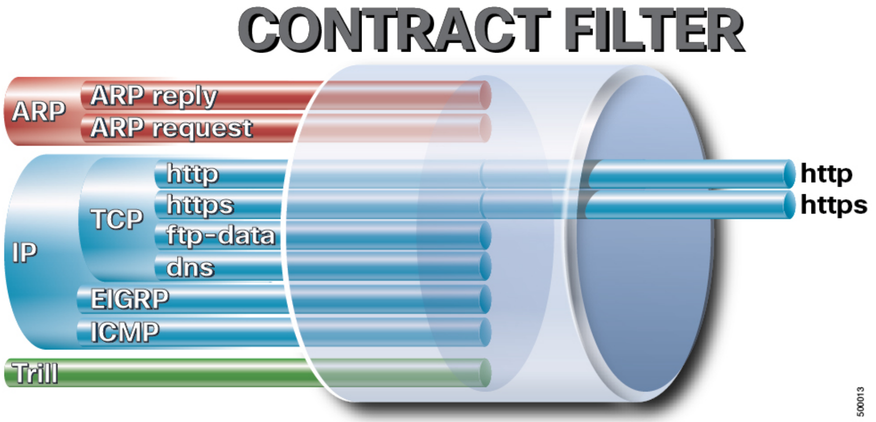
There is also directionality in the relationship between EPGs and contracts. EPGs can either provide or consume a contract. An EPG that provides a contract is typically a set of endpoints that provide a service to a set of client devices. The protocols used by that service are defined in the contract. 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

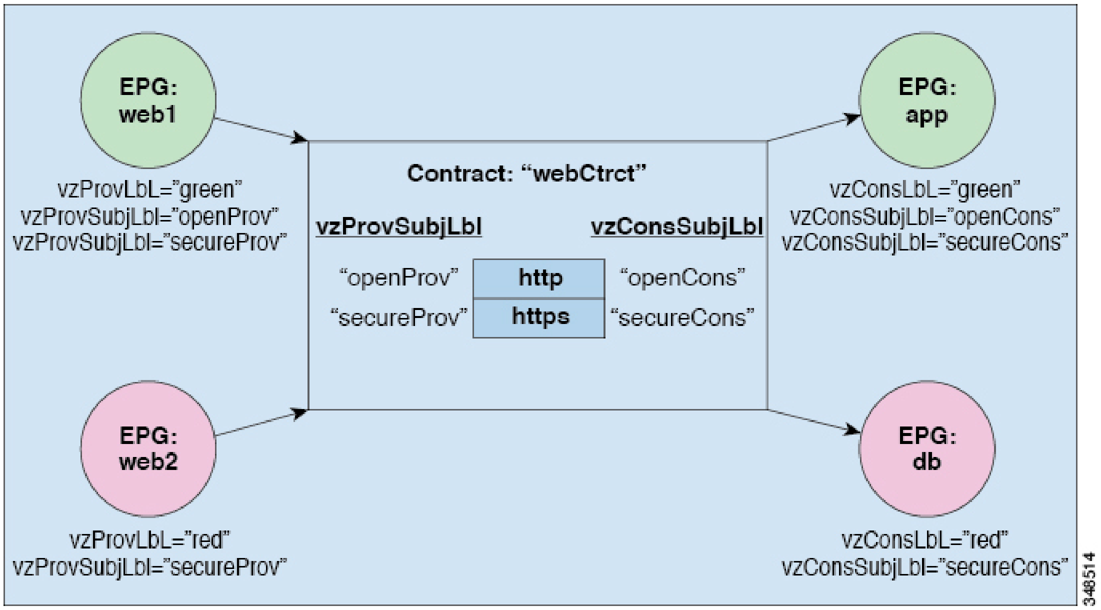
The contract is constructed in a hierarchical manner. It consists of one or more subjects, each subject contains one or more filters, and each filter can define one or more protocols.

Figure 9: Contract Filters



The following figure shows how contracts govern EPG communications.

Figure 10: 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)

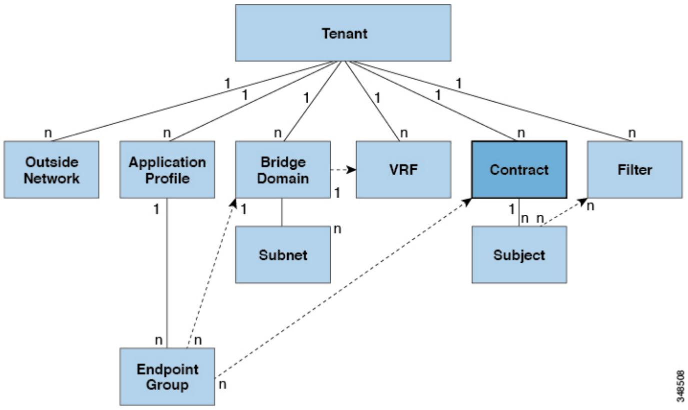
      With Cisco APIC Release 3.2(x) and switches with names that end in EX or FX, you can alternatively use a subject Deny action or Contract or Subject Exception in a standard contract to block traffic with specified patterns.
    - Log the traffic (taboo contracts and regular contracts)
- Aliases—(Optional)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

Contracts

In addition to EPGs, contracts (`vzBrCP`) are key objects in the policy model. EPGs can only communicate with other EPGs according to contract rules. The following figure shows the location of contracts in the management information tree (MIT) and their relation to other objects in the tenant.

Figure 11: Contracts



An administrator uses a contract to select the type(s) of traffic that can pass between EPGs, including the protocols and ports allowed. If there is no contract, inter-EPG communication is disabled by default. There is no contract required for intra-EPG communication; intra-EPG communication is always implicitly allowed.

You can also configure contract preferred groups that enable 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 communication precisely.

Contracts govern the following types of endpoint group communications:

- Between ACI fabric application EPGs (`fvAEPg`), both intra-tenant and inter-tenant

Note In the case of a shared service mode, a contract is required for inter-tenant communication. A contract is used to specify static routes across VRFs, even though the tenant VRF does not enforce a policy.

- Between ACI fabric application EPGs and Layer 2 external outside network instance EPGs (`l2extInstP`)
- Between ACI fabric application EPGs and Layer 3 external outside network instance EPGs (`l3extInstP`)
- Between ACI fabric out-of-band (`mgmtOoB`) or in-band (`mgmtInB`) management EPGs

Contracts govern the communication between EPGs that are labeled providers, consumers, or both. EPG providers expose contracts with which a would-be consumer EPG must comply. The relationship between an EPG and a contract can be either a provider or consumer. When an EPG provides a contract, communication with that EPG can be initiated from other EPGs as long as the communication complies with the provided contract. When an EPG consumes a contract, the endpoints in the consuming EPG may initiate communication with any endpoint in an EPG that is providing that contract.

Note An EPG can both provide and consume the same contract. An EPG can also provide and consume multiple contracts simultaneously.

Configuring a Contract Using the REST API

SUMMARY STEPS

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

DETAILED STEPS

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

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>
```

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`.

Exception criteria exclude these objects as defined in the Consumer Regex and Provider Regex fields	Example	Description
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 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.

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>
```

Configuring EPG Contract Inheritance Using the REST API

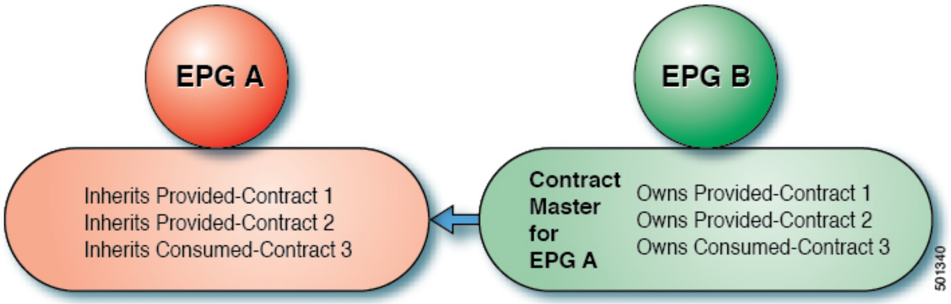
About Contract Inheritance

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 12: Contract Inheritance


EPG A
EPG B
Inherits Provided-Contract 1
Inherits Provided-Contract 2
Inherits Consumed-Contract 3
Contract Master for EPG A
Owns Provided-Contract 1
Owns Provided-Contract 2
Owns Consumed-Contract 3
501340

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.

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

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.

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.

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.

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.

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

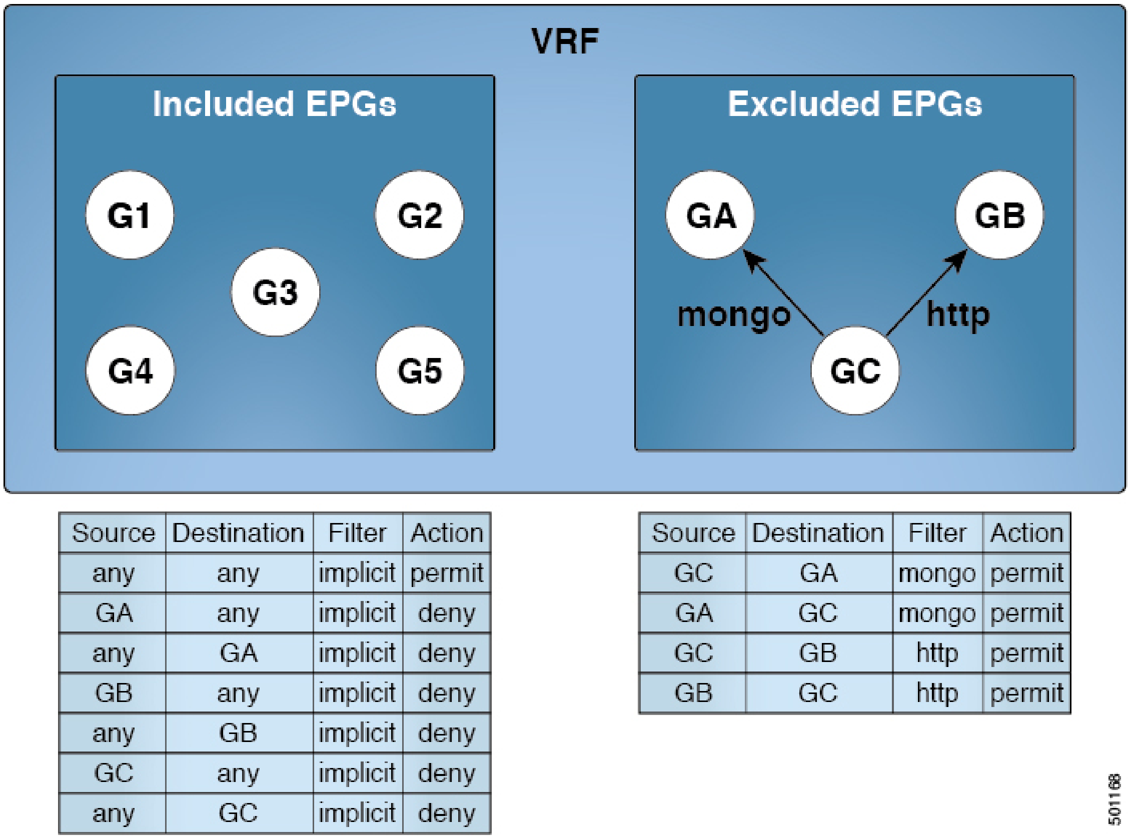
There are two types of policy enforcements available for EPGs in a VRF with a contract preferred group configured:

- 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 13: 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.

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.

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>
```

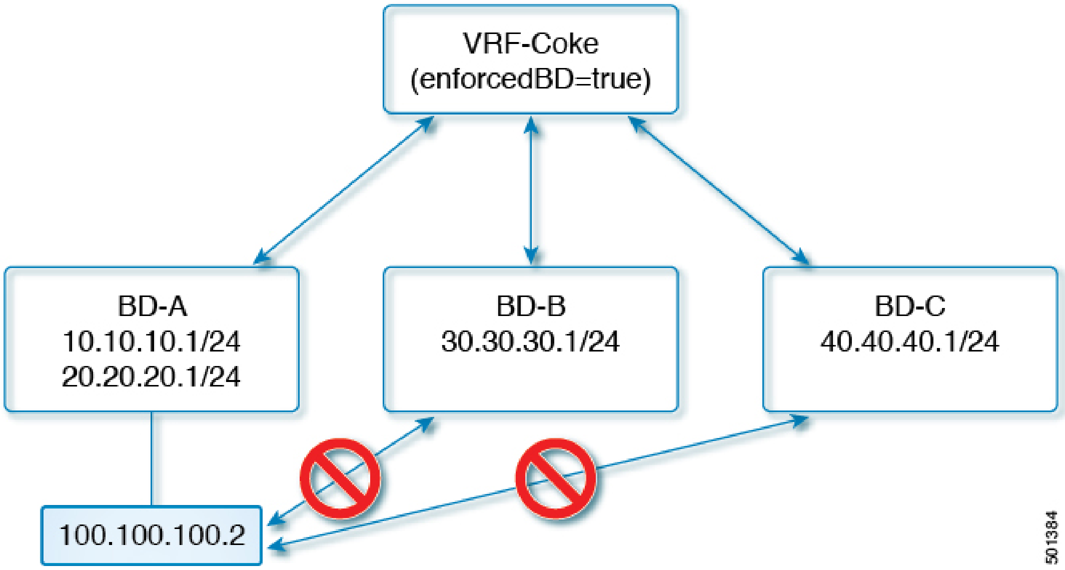
What to do next

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

Configuring an Enforced Bridge Domain

An enforced bridge domain configuration entails creating an endpoint in a subject endpoint group (EPG) that can only ping subnet gateways within the associated bridge domain. With this configuration, you can then create a global exception list of IP addresses that can ping any subnet gateway.

Figure 14: Enforced Bridge Domain



Note

- The exception IP addresses can ping all of the bridge domain gateways across all of your VRF instances.
- A loopback interface configured for an L3Out does not enforce reachability to the IP address that is configured for the subject loopback interface.
- When an eBGP peer IP address exists in a different subnet than the subnet of the L3Out interface, you must add the peer subnet to the allowed exception subnets. Otherwise, eBGP traffic is blocked because the source IP address exists in a different subnet than the L3Out interface subnet.
- For a BGP prefixed-based peer, you must add the peer subnet to the list of allowed exception subnets. For example, if 20.1.1.0/24 is configured as BGP prefixed-based peer, you must add 20.1.1.0/24 to the list of allowed exception subnets.
- An enforced bridge domain is not supported with the Management tenant, regardless if the VRF instances are in-band or out-of-band, and any rules to control the traffic to these VRF instances should be configured using regular contracts.

Configuring an Enforced Bridge Domain Using the REST API

SUMMARY STEPS

1. Create a tenant.
2. Create a VRF and bridge domain.

DETAILED STEPS

	Command or Action	Purpose

Step 1	Create a tenant.
	Example:
	`POST https://apic-ip-address/api/mo/uni.xml`
	`<fvTenant name="ExampleCorp"/>`
		When the POST succeeds, you see the object that you created in the output.

Step 2	Create a VRF and bridge domain.
	Example:
	`URL for POST:`
	`https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml`
	`<fvTenant name="ExampleCorp">`
	`   <fvCtx name="pvn1"/>`
	`   <fvBD name="bd1">`
	`      <fvRsCtx tnFvCtxName="pvn1"`
	`bdEnforceEnable="yes"/>`
	`      <fvSubnet ip="10.10.100.1/24"/>`
	`   </fvBD>`
	`</fvTenant>`
	For adding an exception IP, use the following post:
	https://apic-ip-address/api/node/mo/uni/infra.xml
	`<bdEnforceExceptionCont>`
	`<bdEnforceExceptIp ip="11.0.1.0/24"/>`
	`</bdEnforceExceptionCont>`
	Note If you have a public subnet when you configure the routed outside, you must associate the bridge domain with the outside configuration.
		Note The Gateway Address can be an IPv4 or an IPv6 address. For more about details IPv6 gateway address, see the related KB article, *KB: Creating a Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery* .