

# Troubleshoot ACI Security Policies - Contracts

## Contents

[Introduction](#)

[Background Information](#)

[Overview](#)

[Methods to program zoning-rules](#)

[Comparison between zoning-rule methodologies](#)

[Reading a zoning-rule entry](#)

[Policy Content-Addressable Memory \(CAM\)](#)

[VRF leaking, global pcTags and policy enforcement directionality of shared L3Outs](#)

[VRF policy control enforcement direction](#)

[Where is policy enforced?](#)

[Ingress enforcement and egress enforcement](#)

[Tools](#)

[Zoning-rule validation](#)

['show zoning-rules'](#)

['show zoning-filter'](#)

['show system internal policy-mgr stats'](#)

['show logging ip access-list internal packet-log deny'](#)

[contract\\_parser](#)

[Packet classification validation](#)

[ELAM](#)

[fTriage](#)

[ELAM Assistant App](#)

[Policy CAM usage](#)

[The 'Leaf Capacity' view of Capacity Dashboard](#)

['show platform internal hal health-stats'](#)

[EPG to EPG](#)

[Generic policy drop considerations](#)

[Methodology](#)

[Example troubleshooting scenario EPG to EPG](#)

[Topology](#)

[Identify the source and destination leaf switches involved in the packet drop](#)

[Visibility & Troubleshooting](#)

[Configuration of Visibility & Troubleshooting](#)

[Drop identification](#)

[Drop details](#)

[Contract details](#)

[Contract visualization](#)

[Tenant resource ID to find EPG pcTag and scope](#)

[Verify the policy applied to the traffic flow being troubleshot](#)

[iBash](#)

[ELAM Capture](#)

[ELAM Assistant:](#)

[Configuration](#)

[Elam Assistant Express report](#)

[Elam Assistant Express report \(cont.\)](#)

[Preferred group](#)

[About contract preferred groups](#)

[Contract Preferred Group programming](#)

[Preferred Group Troubleshooting Scenario](#)

[Topology](#)

[Workflow](#)

[vzAny to EPG](#)

[About vzAny](#)

[Example Use Case](#)

[Troubleshooting Scenario - Traffic drops if there is no contract](#)

[Workflow](#)

[Zoning-rules allowing traffic to/from EPG NTP from other EPGs in the VRF present](#)

[Shared L3Out to EPG](#)

[About Shared L3Out](#)

[Troubleshooting a Shared L3out](#)

[Workflow](#)

## Introduction

This document describes steps to understand and troubleshoot ACI Security Policies, known as Contracts.

## Background Information

The material from this document was extracted from the [Troubleshooting Cisco Application Centric Infrastructure, Second Edition](#) book, specifically the **Security Policies - Overview**, **Security Policies - Tools**, **Security Policies - EPG to EPG**, **Security Policies - Preferred group** and **Security Policies - vzAny to EPG** chapters.

## Overview

The fundamental security architecture of the ACI solution follows a permitlist model. Unless a VRF is configured in **unenforced** mode, all EPG to EPG traffic flows are implicitly dropped. As implied by the out-of-the-box permitlist model, the default VRF setting is in **enforced** mode. Traffic flows can be allowed or explicitly denied by implementing zoning-rules on the switch nodes. These zoning-rules can be programmed in a variety of different configurations depending on the desired communication flow between endpoint groups (EPG) and the method used to define them. Note that zoning-rule entries are not stateful and will typically allow/deny based on port/socket given two EPGs once the rule has been programmed.

## Methods to program zoning-rules

The main methods to program zoning-rules within ACI are as follows:

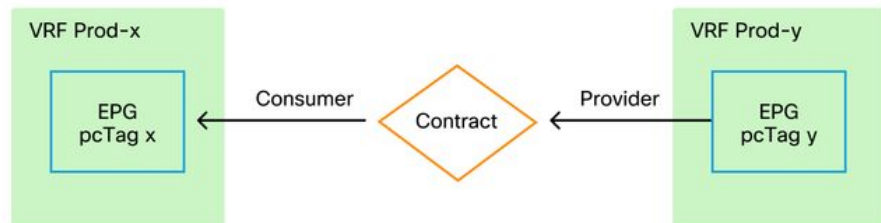
- **EPG-to-EPG Contracts:** Typically requires at least one consumer and one provider to program zoning-rules across two or more distinct endpoint groups.
- **Preferred Groups:** Requires enabling grouping at the VRF level; only one group can exist per VRF. All members of the group can communicate freely. Non-members require contracts to allow flows to the preferred group.
- **vzAny:** An 'EPG Collection' that is defined under a given VRF. vzAny represents all EPGs in the VRF. Usage of vzAny allows flows between one EPG and all EPGs within the VRF via one contract connection.

The following diagram can be used to reference the granularity of zoning-rule that each of the above methods allows for control:

## Comparison between zoning-rule methodologies

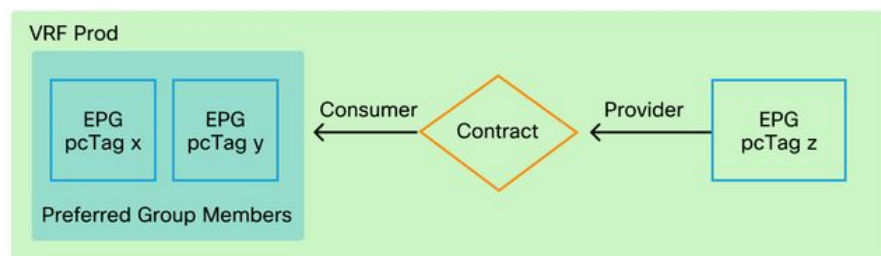
### Contract

- EPG to EPG granularity
- Requires at least 1 consumer and 1 provider
- Can scope across VRFs/Tenants



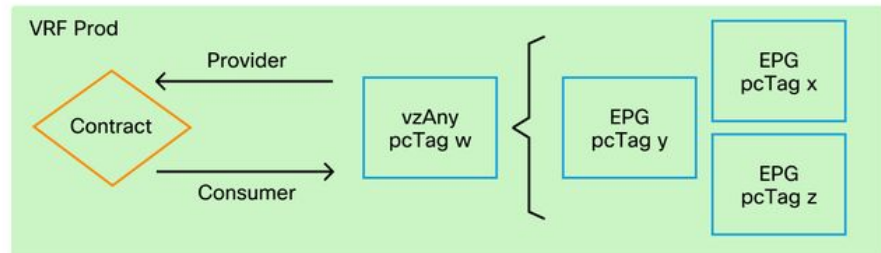
### Preferred Groups

- Must be enabled per VRF
- Only one group per VRF
- EPGs must be explicitly added
- All members communicate freely
- Non-Members require contracts to communicate with members



### vzAny

- Exists within a VRF
- Requires contracts to allow flows
- Zoning-rules apply to all EPGs within the VRF



While utilizing the contract method of programming zoning-rules, there is an option for defining the contract scope. This option must be given careful consideration if any route leaking/shared service design is required. If the wish is to get from one VRF to another within the ACI fabric, contracts are the method to do so.

The scope values can be the following:

- **Application:** a contract consumer/provider relationship will only program rules between EPGs that are defined within the same Application Profile. Re-using the same contract across other

Application Profile EPGs will not allow for crosstalk between them.

- **VRF (default):** a contract consumer/provider relationship will program rules between EPGs that are defined within the same VRF. Re-using the same contract across other Application Profile EPGs will allow for crosstalk between them. Take care to ensure that only desired flows are allowed, otherwise a new contract should be defined to prevent unintentional crosstalk.
- **Tenant:** a contract consumer/provider relationship will program rules between EPGs that are defined within the same tenant. If there are EPGs tied to multiple VRFs within a single tenant and they consume/provide the same contract, this scope can be used to induce route leaking to allow for inter-VRF communication.
- **Global:** a contract consumer/provider relationship will program rules between EPGs across any tenant within an ACI fabric. This is the highest possible scope of the definition, and great care should be taken when this is enabled on previously defined contracts as to prevent unintentional flow leakage.

## Reading a zoning-rule entry

Once the zoning-rule is programmed, it will appear as the following on a leaf:

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name |
Action | Priority |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

- **Rule ID:** the ID of the rule entry. No real significance other than to act as a unique identifier.
- **Src EPG:** a unique ID per VRF (pcTag) of the source endpoint group.
- **Dst EPG:** a unique ID per VRF (pcTag) of the destination endpoint group.
- **FilterID:** the ID of the filter that the rule is attempting to match against. The Filter contains the protocol info that the rule will match against.
- **Dir:** the directionality of the zoning-rule.
- **OperSt:** the operating State of the rule.
- **Scope:** a unique ID of the VRF that the rule will match against.
- **Name:** the name of the contract that resulted in that entry being programmed.
- **Action:** what the leaf will do when it matches that entry. Includes: [Drop, Permit, Log, Redirect].
- **Priority:** the order in which the zoning-rules will be validated for action given a matching Scope, SrcEPG, DstEPG, and Filter Entries.

## Policy Content-Addressable Memory (CAM)

As each zoning rule gets programmed, a matrix of the zoning-rule entry mapped against filter entries will begin to consume **Policy CAM** on the switches. While designing allowed flows through an ACI fabric, special care should be taken when re-using contracts, as opposed to creating new ones, depending on the end design. Haphazardly re-using the same contract across multiple EPGs without understanding the resulting zoning-rules can quickly cascade into multiple flows being allowed unexpectedly. At the same time, these unintentional flows will continue to consume Policy CAM. When Policy CAM becomes full, the zoning-rule programming will begin to fail which can result in unexpected and intermittent loss depending on configuration and endpoint behaviors.

## VRF leaking, global pcTags and policy enforcement directionality of shared L3Outs

This is a special callout for the shared services use case which requires contracts to be configured. Shared services typically imply inter-VRF traffic within an ACI fabric which relies on the usage of either a 'tenant' or 'global' scoped contract. To fully understand this, one must first reinforce the idea that the typical pcTag value assigned to EPGs are not globally unique. pcTags are scoped to a VRF and the same pcTag could potentially be re-used within another VRF. When the discussion of route leaking comes up, start to enforce requirements on the ACI fabric including the need for globally unique values including subnets and pcTags.

What makes this a special consideration is the directionality aspect tied to an EPG being a consumer vs a provider. In a shared services scenario, the provider is typically expected to drive a global pcTag to get a fabric unique value. At the same time, the consumer will retain its VRF-scoped pcTag which puts it in a special position to be able to now program and understand the usage of the global pcTag value to enforce policy.

For reference, the pcTag allocation range is as follows:

- System reserved: 1-15.
- Global scoped: 16-16384 for shared services provider EPGs.
- Local scoped: 16385-65535 for VRF scoped EPGs.

## VRF policy control enforcement direction

In each VRF it is possible to define the enforcement direction setting.

- The default setting of enforcement direction is Ingress.
- The other option for enforcement direction is Egress.

Understanding where the policy is enforced depends on several different variables.

The table below helps to understand where the security policy is enforced at leaf level.

## Where is policy enforced?

Scenario	VRF enforcement mode	Consumer	Provider	Policy enforced on
	Ingress/egress	EPG	EPG	<ul style="list-style-type: none"> <li>• If destination endpoint is learned: ingress leaf*</li> <li>• If destination endpoint is not learned: egress leaf</li> </ul>
	Ingress	EPG	L3Out EPG	Consumer leaf (non-border leaf)
Intra-VRF	Ingress	L3Out EPG	EPG	Provider leaf (non-border leaf)
	Egress	EPG	L3Out EPG	Border leaf -> non-border leaf traffic
	Egress	L3Out EPG	EPG	<ul style="list-style-type: none"> <li>• If destination endpoint is learned: border leaf</li> <li>• If destination endpoint is not learned: non-border leaf</li> </ul> Non-border leaf-> border leaf traffic

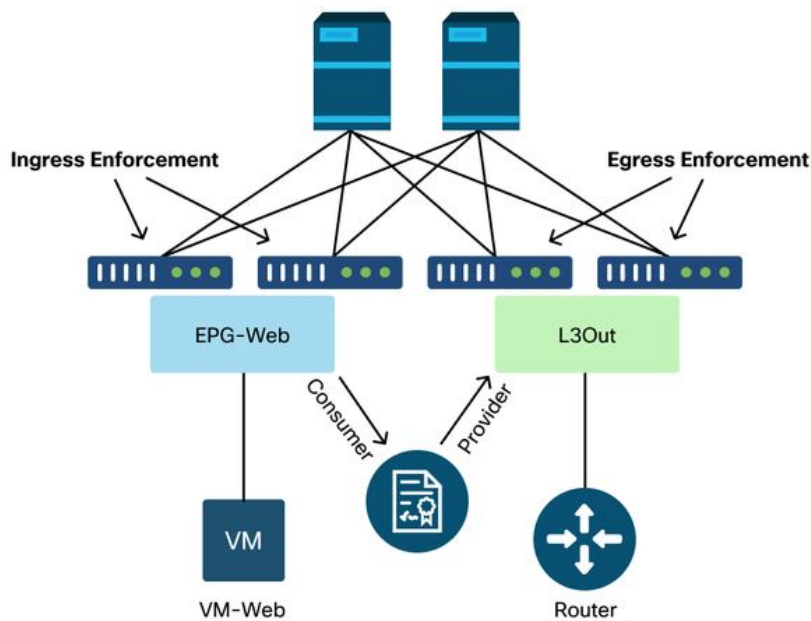
- Border leaf

	Ingress/egress	L3Out	L3Out	Ingress leaf*
	Ingress/egress	EPG	EPG	Consumer leaf
	Ingress/egress	EPG	L3Out	Consumer leaf (Non-border leaf)
Inter-VRF	Ingress/egress	L3Out	EPG	Ingress leaf*
	Ingress/egress	EPG	L3Out	Ingress leaf*

\*Policy enforcement is applied on the first leaf hit by the packet.

The figure below illustrates an example of contract enforcement where EPG-Web as consumer and L3Out EPG as provider have an intra-VRF contract. If VRF is set to Ingress enforcement mode, policy is enforced by the leaf nodes where EPG-Web resides. If VRF is set to Egress enforcement mode, policy is enforced by the border leaf nodes where L3Out resides if VM-Web endpoint is learned on the border leaf.

### Ingress enforcement and egress enforcement



## Tools

There are a variety of tools and commands that can be used to help in the identification of a **policy drop**. A policy drop can be defined as a packet drop due to a contract configuration or lack thereof.

### Zoning-rule validation

The following tools and commands can be used to explicitly validate the zoning-rules that are programmed on leaf switches as a result of completed contract consumer/provider relationships.

## 'show zoning-rules'

A switch level command showing all zoning rules in place.

```
leaf# show zoning-rule
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir      | operSt | Scope  | Name      |
| Action  |         |         |          |          |         |        |           |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 4156   | 25     | 16410  | 425     | uni-dir- | enabled | 2818048 | external_to_ntp |
| permit |         |         |         |          |         |         |           |
| 4131   | 16410  | 25     | 424     | bi-dir   | enabled | 2818048 | external_to_ntp |
| permit |         |         |         |          |         |         |           |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
```

## 'show zoning-filter'

A filter that contains the sport/dport information that the zoning rule is acting on. The filter programming can be verified with this command.

```
leaf# show zoning-filter
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| FilterId | Name      | EtherT | Prot  | ApplyToFrag | Stateful | SFromPort |
| SToPort  | DFromPort | DToPort | Prio  |              |          |            |
+-----+-----+-----+-----+-----+-----+-----+-----+
| implarp  | implarp  | arp     | unspecified | no          | no       | unspecified |
| unspecified | unspecified | unspecified | dport      |             |          |             |
| implicit | implicit | unspecified | unspecified | no          | no       | unspecified |
| unspecified | unspecified | unspecified | implicit   |             |          |             |
| 425     | 425_0    | ip      | tcp     | no          | no       | 123         |
| 123     | unspecified | unspecified | sport    |             |          |             |
| 424     | 424_0    | ip      | tcp     | no          | no       | unspecified  |
| unspecified | 123      | 123     | dport    |             |          |             |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
```

## 'show system internal policy-mgr stats'

This command can be run to verify the number of hits per zoning-rule. This is useful to determine whether an expected rule is being hit as opposed to another, such as an implicit drop rule that may have a higher priority.

```
leaf# show system internal policy-mgr stats
Requested Rule Statistics
Rule (4131) DN (sys/actrl/scope-2818048/rule-2818048-s-16410-d-25-f-424) Ingress: 0, Egress: 0,
Pkts: 0 RevPkts: 0
Rule (4156) DN (sys/actrl/scope-2818048/rule-2818048-s-25-d-16410-f-425) Ingress: 0, Egress: 0,
Pkts: 0 RevPkts: 0
```

## 'show logging ip access-list internal packet-log deny'

A switch level command that can be run at iBash level which reports ACL (contract) related drops

and flow-related information including:

- VRF
- VLAN-ID
- Source MAC/Dest MAC
- Source IP/Dest IP
- Source Port/Dest Port
- Source Interface

```
leaf# show logging ip access-list internal packet-log deny
[ Tue Oct  1 10:34:37 2019 377572 usecs]: CName: Prod1:VRF1(VXLAN: 2654209), VlanType: Unknown,
Vlan-Id: 0, SMac: 0x000c0c0c0c0c, DMac:0x000c0c0c0c0c, SIP: 192.168.21.11, DIP: 192.168.22.11,
SPort: 0, DPort: 0, Src Intf: Tunnel7, Proto: 1, PktLen: 98
[ Tue Oct  1 10:34:36 2019 377731 usecs]: CName: Prod1:VRF1(VXLAN: 2654209), VlanType: Unknown,
Vlan-Id: 0, SMac: 0x000c0c0c0c0c, DMac:0x000c0c0c0c0c, SIP: 192.168.21.11, DIP: 192.168.22.11,
SPort: 0, DPort: 0, Src Intf: Tunnel7, Proto: 1, PktLen: 98
```

## contract\_parser

An on-device Python script which produces an output that correlates the zoning-rules, filters and hit statistics while performing name lookups from IDs. This script is extremely useful in that it takes a multi-step process and turns it into a single command which can be filtered to specific EPGs/VRFs or on other contract related values.

```
leaf# contract_parser.py
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-l4] dst-epg [dst-l4]
[flags][contract:{str}] [hit=count]

[7:4131] [vrf:common:default] permit ip tcp tn-Prod1/ap-Services/epg-NTP(16410) tn-Prod1/l3out-
L3Out1/instP-extEpg(25) eq 123 [contract:uni/tn-Prod1/brc-external_to_ntp] [hit=0]
[7:4156] [vrf:common:default] permit ip tcp tn-Prod1/l3out-L3Out1/instP-extEpg(25) eq 123 tn-
Prod1/ap-Services/epg-NTP(16410) [contract:uni/tn-Prod1/brc-external_to_ntp] [hit=0]
[12:4169] [vrf:common:default] deny,log any tn-Prod1/l3out-L3Out1/instP-extEpg(25) epg:any
[contract:implicit] [hit=0]
[16:4167] [vrf:common:default] permit any epg:any tn-Prod1/bd-Services(32789)
[contract:implicit] [hit=0]
```

## Packet classification validation

### ELAM

An ASIC level report used to check forwarding details which indicates, in the case of a dropped packet, the drop reason. Relevant to this section, the reason can be a SECURITY\_GROUP\_DENY (contract policy drop).

### fTriage

A Python-based utility on the APIC which can track end-to-end packet flow with ELAM.

### ELAM Assistant App

An APIC App that abstracts the complexity of various ASICs to make forwarding decision



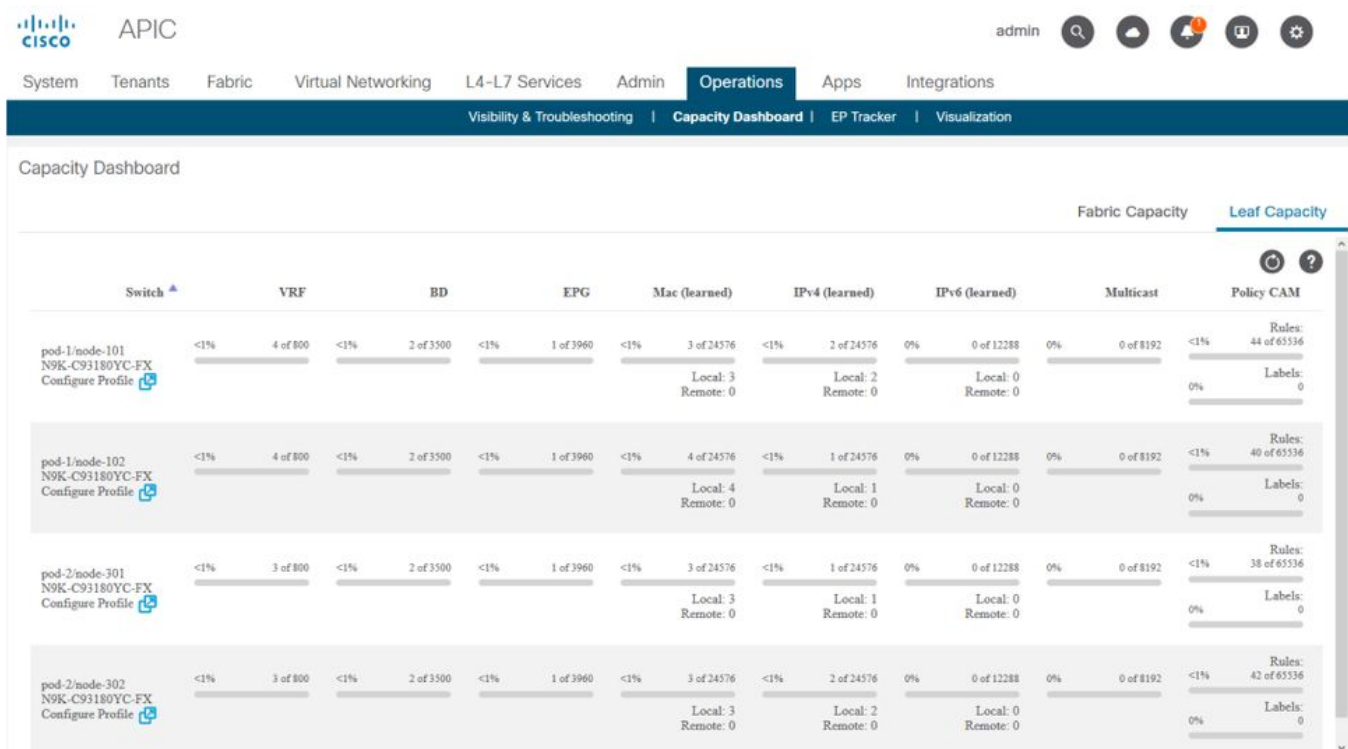
inspection much more convenient and user friendly.

Please refer to the "Intra-Fabric Forwarding" section for additional details on the ELAM, fTriage and ELAM Assistant Tools

## Policy CAM usage

Policy CAM usage on a per leaf basis is an important parameter to monitor to ensure the fabric is in a healthy status. The quickest way to monitor that is to use the 'Capacity Dashboard' within the GUI and explicitly check the 'Policy Cam' column.

## The 'Leaf Capacity' view of Capacity Dashboard



## 'show platform internal hal health-stats'

This command is useful for validating a variety of resource limits and usage, including Policy CAM. Note that this command can only be run in vsh\_lc, so pass it in using the '-c' flag if being run from iBash.

```
leaf8# vsh_lc -c "show platform internal hal health-stats"
|Sandbox_ID: 0 Asic Bitmap: 0x0
|-----
...
Policy stats:
=====
policy_count           : 96
max_policy_count      : 65536
policy_otcam_count    : 175
max_policy_otcam_count : 8192
policy_label_count    : 0
max_policy_label_count : 0
=====
```

# EPG to EPG

## Generic policy drop considerations

There are numerous ways to troubleshoot a connectivity issue between two endpoints. The following methodology provides a good starting point to quickly and effectively isolate whether the connectivity issue is the result of a **policy drop** (contract induced).

Some high-level questions worth asking before diving in:

- Are the endpoints in same or different EPG? Traffic between two endpoints residing in different EPGs (inter-EPG) is implicitly denied and requires a contract to allow communication. Traffic between two endpoints within the same EPG (intra-EPG) is implicitly allowed, unless intra-EPG isolation is in use.
- Is the VRF enforced or unenforced? When a VRF is in **enforced** mode, — within the VRF — contracts are required for endpoints in two different EPGs to communicate. When a VRF is in **unenforced** mode, — within the VRF — all traffic would be allowed by the ACI fabric across multiple EPGs belonging to the unenforced VRF, regardless of the ACI contracts applied.

## Methodology

With the various tools available, there are some that are more appropriate and convenient to start with than others, depending on the level of information already known about the affected flow.

Is the full path of the packet in the ACI fabric known (ingress leaf, egress leaf...)?

- If the answer is yes, ELAM Assistant should be used to identify the drop reason on the source or destination switch.
- If the answer is no, Visibility & Troubleshooting, fTriage, contract\_parser, Operational tab in the Tenant view, and iBash commands will help to narrow down the path of the packet or give more visibility into the drop reasons.

Please note that the fTriage tool will not be discussed in detail in this section. Refer to the chapter "Intra-Fabric Forwarding" for more detail on using this tool.

Consider that while Visibility & Troubleshooting can help to quickly visualize where packets are dropped between two endpoints, fTriage shows more in-depth information for further troubleshooting. i.e. fTriage will help identify interface, drop reason, and other low-level details about the affected flow

This example scenario will show how to troubleshoot a policy drop between two endpoints: 192.168.21.11 and 192.168.23.11

Assuming packet drops are experienced between those two endpoints, the following troubleshooting workflow will be used to identify the root cause of the problem:

Identify the src/dst leaf(s) involved in the traffic flow:

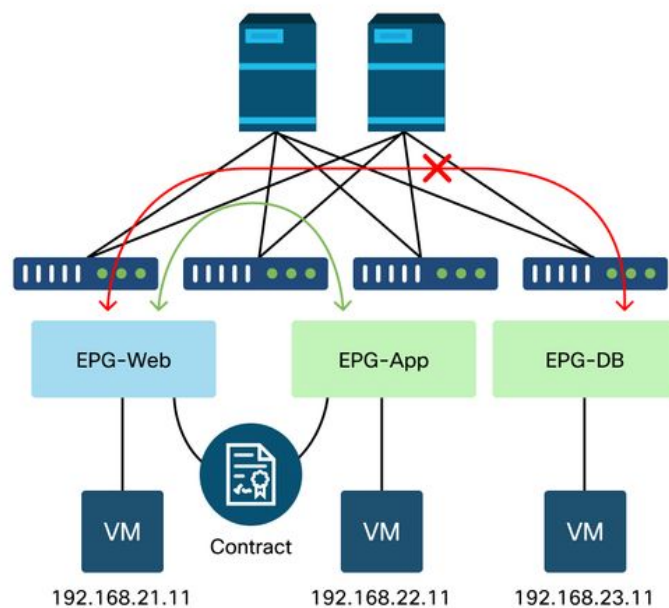
1. Use **Visibility & Troubleshooting** to trace the packet flow and identify which device is dropping the packet.
2. Run the command 'show logging ip access-list internal packet-log deny' on the selected device. If a packet with one of the IP addresses of interest is being denied and logged, the **packet-log** will print the relevant endpoint and contract name on a per hit basis.
3. Use command 'contract\_parser.py --vrf <tenant>:<VRF>' on source and destination leaf to observe hit count for the configured contract: If a packet is hitting the contract on either the source or destination switch, the counter of the relevant contract will increment. This method is less granular than that of IP access-list internal packet-log in situations where many flows could be hitting the same rule (many endpoints/flows between the two EPGs of interest).

The above steps are described further in the next paragraph.

## Example troubleshooting scenario EPG to EPG

This example scenario will show how to troubleshoot a policy drop between two endpoints: 192.168.21.11 in EPG-Web and 192.168.23.11 in EPG-DB.

### Topology



### Identify the source and destination leaf switches involved in the packet drop

#### Visibility & Troubleshooting

The Visibility & Troubleshooting tool will help to visualize the switch where the packet drop occurred for a specific EP-to-EP flow and identify where packets are possibly dropped.

#### Configuration of Visibility & Troubleshooting

**Visibility & Troubleshooting**

**This tool provides:**

1. Location of the specified end points in the fabric and displays the traffic path including any L4-L7 devices. Along the path between these end points, statistics, contracts, faults, events, and audit logs are displayed in scope.
2. Optional triggering of traceroute, and atomic counters for troubleshooting these end points. These debugging steps create and delete corresponding debugging policies as needed.

Session Name:

Session Type:

Description:

**Targets**

Source

Learned At	Tenant	Application	EPG
Pod:1, Leaf:105, Port:eth1/19	Prod1	AppProf	Web

Destination

Learned At	Tenant	Application	EPG
Pod:1, Leaf:105, Port:eth1/19	Prod1	AppProf	DB

Configure a Session Name, Source, and Destination endpoint. Then click 'Submit' or 'Generate Report'.

The tool will automatically find the endpoints in the fabric and provide information about the Tenant, Application Profile and EPG those EP belong to.

In this case, it will discover that the EPs belong to the tenant Prod1, they belong to the same Application Profile 'AppProf' and are assigned to different EPGs: 'Web' and 'DB'.

## Drop identification

**Drop/Stats**

From: latest 240 minutes  
To: now

Session Information

Source	192.168.21.11
Destination	192.168.23.11
Type	Endpoint → Endpoint

Source Endpoint  
IP: 192.168.21.11  
MAC: F6:F2:6C:4E:C8:D0

The tool will automatically visualize the topology of the troubleshooting scenario. In this case, the two endpoints happen to be connected to the same leaf switch.

By navigating to the Drop/Stats submenu, the user can view general drops on the leaf or spine in question. Refer to the "Interface Drops" section in the chapter "Intra-Fabric Forwarding" of this book for more information about understanding which drops are relevant.

Many of these drops are expected behavior and can be ignored.

## Drop details

Statistics - fab3-leaf5



				Drop Stats	Contract Drops	Traffic Stats
<input type="checkbox"/> Show stats with zero values						
Time	Affected Object	Stats	Value			
2019/10/02 03:49:58 - 2019/10/02 03:54:58	topology/pod-1/node-105/sys/ctx-[vxlan-2654209]/bd-[vxlan-16220082]/vlan-[vlan-701]	ingress drop packets periodic	3			
2019/10/02 03:39:48 - 2019/10/02 03:44:58	topology/pod-1/node-105/sys/ctx-[vxlan-2654209]/bd-[vxlan-16121802]/vlan-[vlan-703]	ingress drop packets periodic	3			
2019/10/02 03:29:58 - 2019/10/02 03:44:58	topology/pod-1/node-105/sys/ctx-[vxlan-2654209]/bd-[vxlan-16121802]/vlan-[vlan-703]	ingress drop packets periodic	3			
2019/10/02 03:29:58 - 2019/10/02 03:44:58	topology/pod-1/node-105/sys/ctx-[vxlan-2654209]/bd-[vxlan-16220082]/vlan-[vlan-701]	ingress drop packets periodic	3			
2019/10/02 03:14:58 - 2019/10/02 03:29:58	topology/pod-1/node-105/sys/ctx-[vxlan-2654209]/bd-[vxlan-16121802]/vlan-[vlan-703]	ingress drop packets periodic	3			

By drilling down to drop detail using the yellow 'Packets dropped' button on the switch diagram, the user can view details about the dropped flow.

## Contract details

### S Source Endpoint → Destination Endpoint

Filter ID: implicit							BD Allow (Prod1/DB)	
Info	Protocol	L4 Src	L4 Dest	TCP Flags	Action	Nodes	Hits	
					permit	node-105	0	
Filter ID: implicit							Context Implicit (Prod1/VRF1)	
Info	Protocol	L4 Src	L4 Dest	TCP Flags	Action	Nodes	Hits	
					deny,log	node-105	8636	

### D Destination Endpoint → Source Endpoint

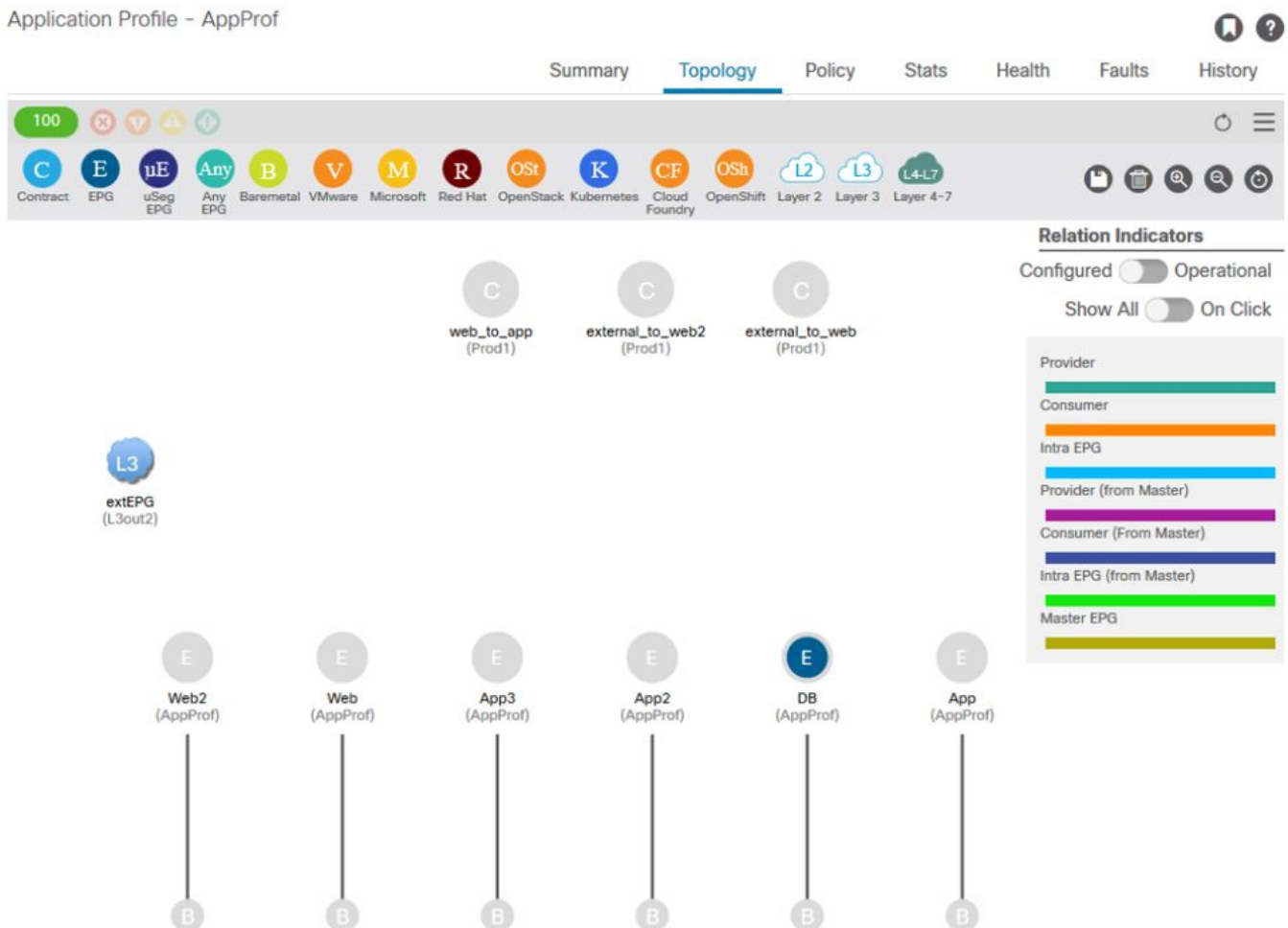
Filter ID: implicit							BD Allow (Prod1/Web)	
Info	Protocol	L4 Src	L4 Dest	TCP Flags	Action	Nodes	Hits	
					permit	node-105	0	
Filter ID: implicit							Context Implicit (Prod1/VRF1)	
Info	Protocol	L4 Src	L4 Dest	TCP Flags	Action	Nodes	Hits	
					deny,log	node-105	8636	

By navigating to the Contracts submenu, the user can identify which contract is causing policy drop off between the EPGs. In the example, it is Implicit to Deny Prod1/VRF1 which shows some hits. This does not necessarily mean the specified flow (192.168.21.11 and 192.168.23.11) is

hitting this implicit deny. If the Hits of Context Implicit deny rule is increasing, it implies there is traffic between Prod1/DB and Prod1/Web that do not hit any of contracts, hence are dropped by the Implicit deny.

In the Application Profile Topology view at Tenant > select the Application Profile name on the left > Topology , it is possible to verify which contracts are applied to the DB EPG. In this case, no contract is assigned to the EPG:

## Contract visualization



Now that the source and destination EPGs are known, it is also possible to identify other relevant information such as the following:

- The src/dst **EPG pcTag** of the affected endpoints. The pcTag is the class ID used to identify an EPG with a zoning-rule.
- The src/dst **VRFVNIID**, also referred to as **scope**, of the affected endpoints.

Class ID and scope can be easily retrieved from the APIC GUI by opening the Tenant > select the Tenant name on the left > Operational > Resource IDs > EPGs

## Tenant resource ID to find EPG pcTag and scope

Tenant - Prod1

Summary Dashboard Policy **Operational** Stats Health Faults History

Flows Packets **Resource IDs**

Bridge Domains VRFs **EPGs** L3Outs External Networks (Bridged)

99

Application Profile Name	AP Alias	EPG Name	Class ID	Scope
AppProf		App	32774	2654209
AppProf		App2	32775	2654209
AppProf		App3	49160	2654209
AppProf		DB	49159	2654209
AppProf		Web	32778	2654209
AppProf		Web2	16388	2097160
Services		NTP	16410	2818048

In this case the Class ID and Scopes are:

- Web EPG pcTag 32778
- Web EPG scope 2654209
- DB EPG pcTag 49159
- DB EPG scope 2654209

## Verify the policy applied to the traffic flow being troubleshot

### iBash

An interesting tool to verify the packet dropped on an ACI leaf is the iBash command line: 'show logging ip access-list internal packet-log deny':

```
leaf5# show logging ip access-list internal packet-log deny | grep 192.168.21.11
[2019-10-01T14:25:44.746528000+09:00]: CName: Prod1:VRF1(VXLAN: 2654209), VlanType: FD_VLAN,
Vlan-Id: 114, SMac: 0xf6f26c4ec8d0, DMac:0x0022bdf819ff, SIP: 192.168.21.11, DIP: 192.168.23.11,
SPort: 0, DPort: 0, Src Intf: Ethernet1/19, Proto: 1, PktLen: 126
[2019-10-01T14:25:44.288653000+09:00]: CName: Prod1:VRF1(VXLAN: 2654209), VlanType: FD_VLAN,
Vlan-Id: 116, SMac: 0x3e2593f0eded, DMac:0x0022bdf819ff, SIP: 192.168.23.11, DIP: 192.168.21.11,
SPort: 0, DPort: 0, Src Intf: Ethernet1/19, Proto: 1, PktLen: 126
```

As per the previous output, it can be seen that on the leaf switch, numerous ICMP packets sourced by EP 192.168.23.11 towards 192.168.21.11 have been dropped.

The contract\_parser tool will help to verify the actual policies applied to the VRF where the Endpoints are associated with:

```
leaf5# contract_parser.py --vrf Prod1:VRF1
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-l4] dst-epg [dst-l4]
[flags][contract:{str}] [hit=count]
```

```
[7:5159] [vrf:Prod1:VRF1] permit ip tcp tn-Prod1/ap-App1/epg-App(32771) eq 5000 tn-Prod1/ap-App1/epg-Web(32772) [contract:uni/tn-Prod1/brc-web_to_app] [hit=0]
[7:5156] [vrf:Prod1:VRF1] permit ip tcp tn-Prod1/ap-App1/epg-Web(32772) tn-Prod1/ap-App1/epg-App(32771) eq 5000 [contract:uni/tn-Prod1/brc-web_to_app] [hit=0]
[16:5152] [vrf:Prod1:VRF1] permit any epg:any tn-Prod1/bd-Web(49154) [contract:implicit] [hit=0]
[16:5154] [vrf:Prod1:VRF1] permit arp epg:any epg:any [contract:implicit] [hit=0]
[21:5155] [vrf:Prod1:VRF1] deny,log any epg:any epg:any [contract:implicit] [hit=38,+10]
[22:5153] [vrf:Prod1:VRF1] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
```

This can also be verified through the zoning rule programmed in the leaf the policies enforced by the switch.

```
leaf5# show zoning-rule scope 2654209
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name |
Action | Priority |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+
| 5155 | 0 | 0 | implicit | uni-dir | enabled | 2654209 |
deny,log | any_any_any(21) |
| 5159 | 32771 | 32772 | 411 | uni-dir-ignore | enabled | 2654209 | web_to_app |
permit | fully_qual(7) |
| 5156 | 32772 | 32771 | 410 | bi-dir | enabled | 2654209 | web_to_app |
permit | fully_qual(7) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+
```

As already seen by the Visibility & Troubleshooting tool, the contract\_parser tool, and the zoning rules, output confirms there is no contract between the source and destination EPGs in troubleshooting. It is easy to assume that the packets dropped are matching the implicit deny rule 5155.

## ELAM Capture

ELAM capture provides an ASIC level report used to check forwarding details which indicates, in the case of a dropped packet, the drop reason. When the reason of a drop is a policy drop, as in this scenario, the output of the ELAM capture will look like the following.

Please note that details of setting up an ELAM capture will not be discussed in this chapter, please see the chapter "Intra-Fabric Forwarding".

```
leaf5# vsh_lc
module-1# debug platform internal tah elam asic 0
module-1(DBG-elam)# trigger init in-select 6 out-select 0
module-1(DBG-elam)# trigger reset
module-1(DBG-elam-insel6)# set outer ipv4 src_ip 192.168.21.11 dst_ip 192.168.23.11
module-1(DBG-elam-insel6)# start
module-1(DBG-elam-insel6)# status
```

```
ELAM STATUS
```

```
=====
```

```
Asic 0 Slice 0 Status Triggered
Asic 0 Slice 1 Status Armed
```

```
module-1(DBG-elam-insel6)# ereport | grep reason
RW drop reason : SECURITY_GROUP_DENY
LU drop reason : SECURITY_GROUP_DENY
pkt.lu_drop_reason: 0x2D
```

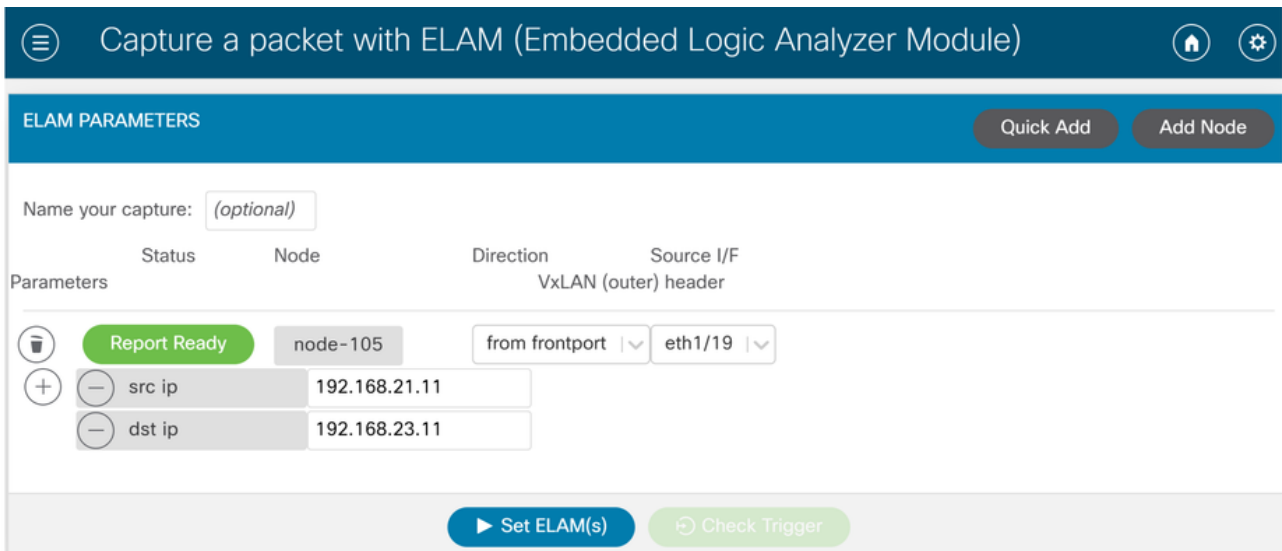


The ELAM report above shows clearly that the packet was dropped due to a policy drop: 'SECURITY\_GROUP\_DENY'

### ELAM Assistant:

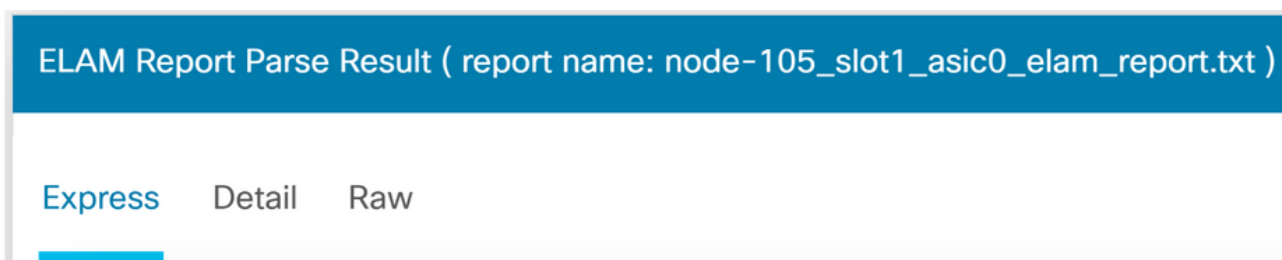
The very same result of the ELAM capture can be shown through the ELAM Assistant App on the APIC GUI.

### Configuration



Typically, the user will configure both source and destination details for the flow of interest. In this example, src IP is used to capture traffic towards endpoint in destination EPG that does not have a contract relationship to the source EPG.

### Elam Assistant Express report



There are three levels of output that can be viewed with ELAM Assistant. These are Express, Detail, and Raw.

### Elam Assistant Express report (cont.)

## Packet Forwarding Information

Forward Result	
Destination Type	To a local port
Destination Logical Port	Eth1/19
Destination Physical Port	packet dropped
Sent to SUP/CPU instead	yes
SUP Redirect Reason (SUP code)	ISTACK_SUP_CODE_ACL_LOG

Contract	
Destination EPG pcTag (dclass)	16387 (Prod1:App1:DB)
Source EPG pcTag (sclass)	10935 (Prod1:App1:Web)
Contract was applied	0 (Contract was not applied on this node)

Drop	
Drop Code	SECURITY_GROUP_DENY

Under the Express Result, the Drop Code reason `SECURITY_GROUP_DENY` indicates that the drop was a result of a contract hit.

## Preferred group

### 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 deny rules between the excluded EPG and any EPG apply.

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, configure a combination of a contract preferred group and contracts with filters to more precisely control inter-EPG communication.

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.

### Contract Preferred Group programming

Essentially, Contract Preferred Groups are an inverse of regular contracts. For regular contracts, explicit permit zoning-rules are programmed with an implicit deny zoning-rule with the VRF Scope. For Preferred Groups, an implicit PERMIT zoning-rule is programmed with the highest numeric

priority value and specific DENY zoning-rules are programmed to disallow traffic from EPGs which are not Preferred Group members. As a result, the deny rules are evaluated first and if the flow isn't matched by these rules, then the flow is implicitly permitted.

There's always a pair of explicit deny zoning-rules for every EPG outside of the preferred group:

- One from the non-Preferred Group member to any pcTag (value 0).
- Another from any pcTag (value 0) to the non-Preferred Group member.

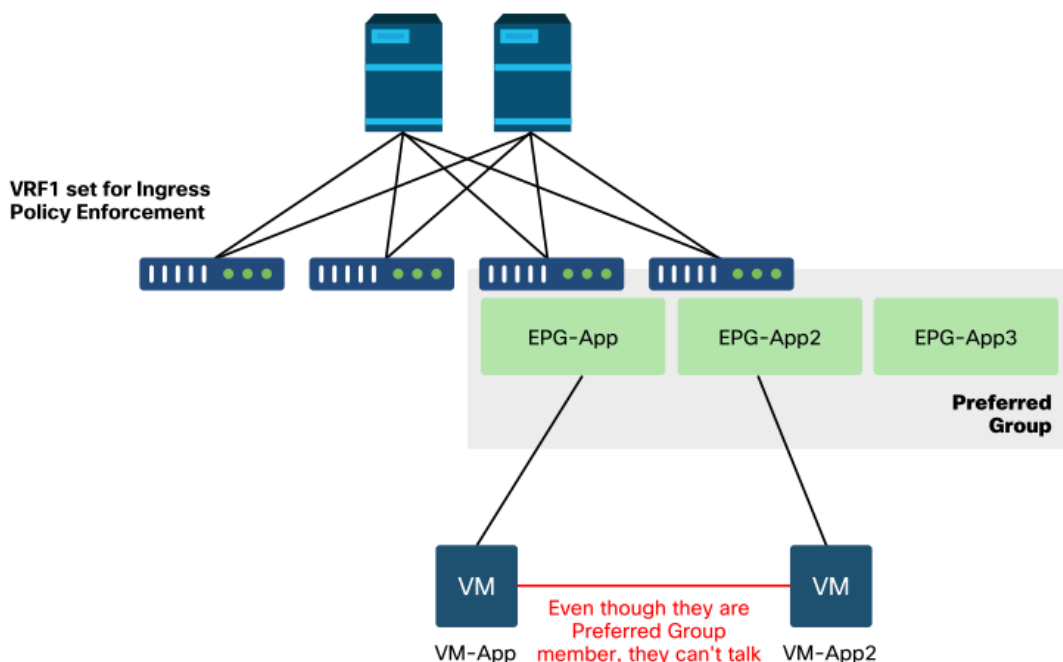
## Preferred Group Troubleshooting Scenario

The figure below shows a logical topology in which EPGs App, App2 and App3 are all configured as Preferred Group Members.

VM-App is part of EPG-App and VM-App2 is part of EPG-App2. Both App and App2 EPG should be part of the preferred and hence communicate freely.

VM-App initiates a traffic flow on TCP port 6000 to VM-App2. Both EPG-App and EPG-App2 are Preferred Group Members as part of VRF1. VM-App2 never receives any packets on TCP port 6000.

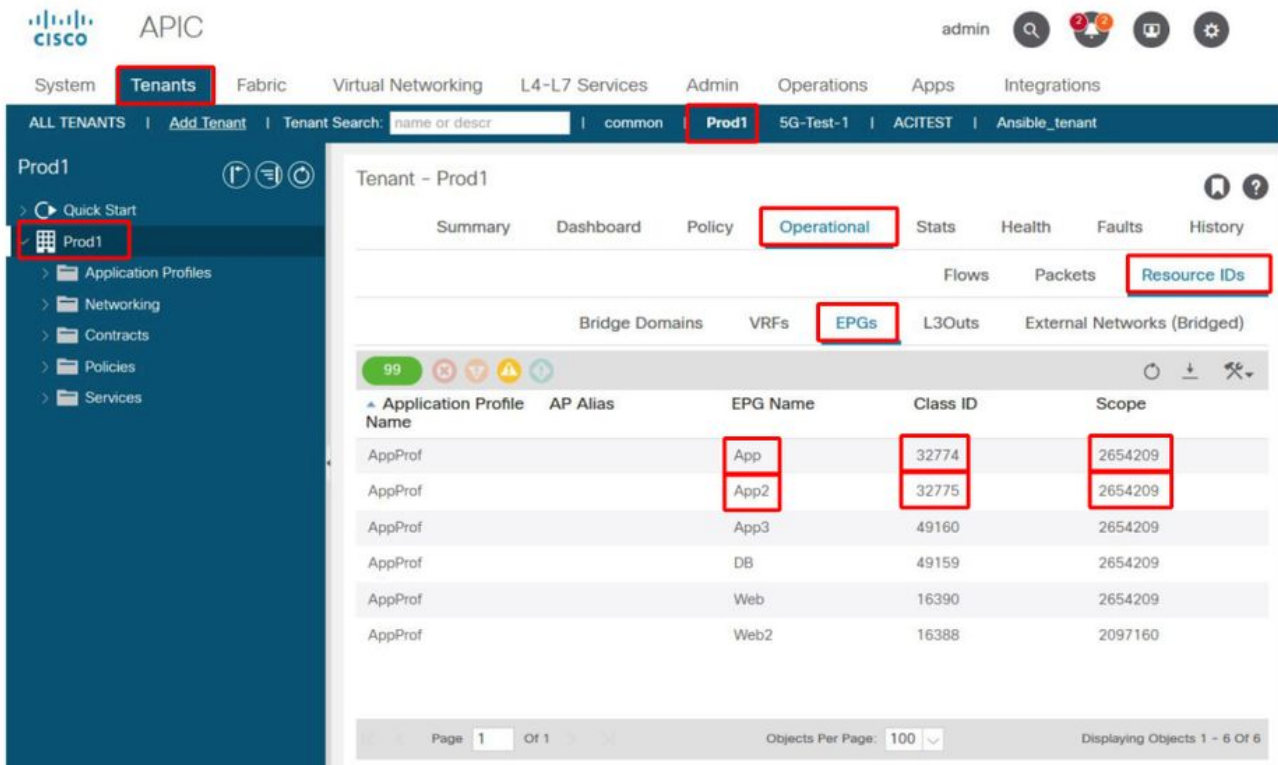
### Topology



### Workflow

1. Look up the pcTag of EPG APP and its VRF VNID/Scope

EPG and VRF pcTags



## 2. Verify contract programming using contract\_parser.py on the ingress leaf

Use contract\_parser.py and/or the 'show zoning-rule' command and specify the VRF

```
fab3-leaf8# show zoning-rule scope 2654209
```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name | Action |
|         | Priority |         |         |     |         |       |      |        |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 4165 | 0 | 0 | implicit | uni-dir | enabled | 2654209 | | permit |
grp_any_any_any_permit(20) |
| 4160 | 0 | 0 | implarp | uni-dir | enabled | 2654209 | | permit |
any_any_filter(17) |
| 4164 | 0 | 15 | implicit | uni-dir | enabled | 2654209 | | deny,log |
grp_any_dest_any_deny(19) |
| 4176 | 0 | 16386 | implicit | uni-dir | enabled | 2654209 | | permit |
any_dest_any(16) |
| 4130 | 32770 | 0 | implicit | uni-dir | enabled | 2654209 | | deny,log |
grp_src_any_any_deny(18) |
| 4175 | 49159 | 0 | implicit | uni-dir | enabled | 2654209 | | deny,log |
grp_src_any_any_deny(18) |
| 4129 | 0 | 49159 | implicit | uni-dir | enabled | 2654209 | | deny,log |
grp_any_dest_any_deny(19) |
| 4177 | 32778 | 0 | implicit | uni-dir | enabled | 2654209 | | deny,log |
grp_src_any_any_deny(18) |
| 4128 | 0 | 32778 | implicit | uni-dir | enabled | 2654209 | | deny,log |
grp_any_dest_any_deny(19) |
| 4178 | 32775 | 0 | implicit | uni-dir | enabled | 2654209 | | deny,log |
grp_src_any_any_deny(18) |
| 4179 | 0 | 32775 | implicit | uni-dir | enabled | 2654209 | | deny,log |
grp_any_dest_any_deny(19) |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```
fab3-leaf8# contract_parser.py --vrf Prod1:VRF1
```

```

Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-l4] dst-epg [dst-l4]
[flags][contract:{str}] [hit=count]
[16:4176] [vrf:Prod1:VRF1] permit any epg:any tn-Prod1/bd-App(16386) [contract:implicit] [hit=0]
[16:4160] [vrf:Prod1:VRF1] permit arp epg:any epg:any [contract:implicit] [hit=0]
[18:4130] [vrf:Prod1:VRF1] deny,log any tn-Prod1/vrf-VRF1(32770) epg:any [contract:implicit]
[hit=?]
[18:4178] [vrf:Prod1:VRF1] deny,log any epg:32775 epg:any [contract:implicit] [hit=?]
[18:4177] [vrf:Prod1:VRF1] deny,log any epg:32778 epg:any [contract:implicit] [hit=?]
[18:4175] [vrf:Prod1:VRF1] deny,log any epg:49159 epg:any [contract:implicit] [hit=?]
[19:4164] [vrf:Prod1:VRF1] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
[19:4179] [vrf:Prod1:VRF1] deny,log any epg:any epg:32775 [contract:implicit] [hit=?]
[19:4128] [vrf:Prod1:VRF1] deny,log any epg:any epg:32778 [contract:implicit] [hit=?]
[19:4129] [vrf:Prod1:VRF1] deny,log any epg:any epg:49159 [contract:implicit] [hit=?]
[20:4165] [vrf:Prod1:VRF1] permit any epg:any epg:any [contract:implicit] [hit=65]

```

Examining the above output, the implicit permit entry — ruleId 4165 — with the highest priority of 20, is observed. This implicit permit rule will cause all traffic flows to be allowed unless there's an explicit deny rule with a lower priority disallowing the traffic flow.

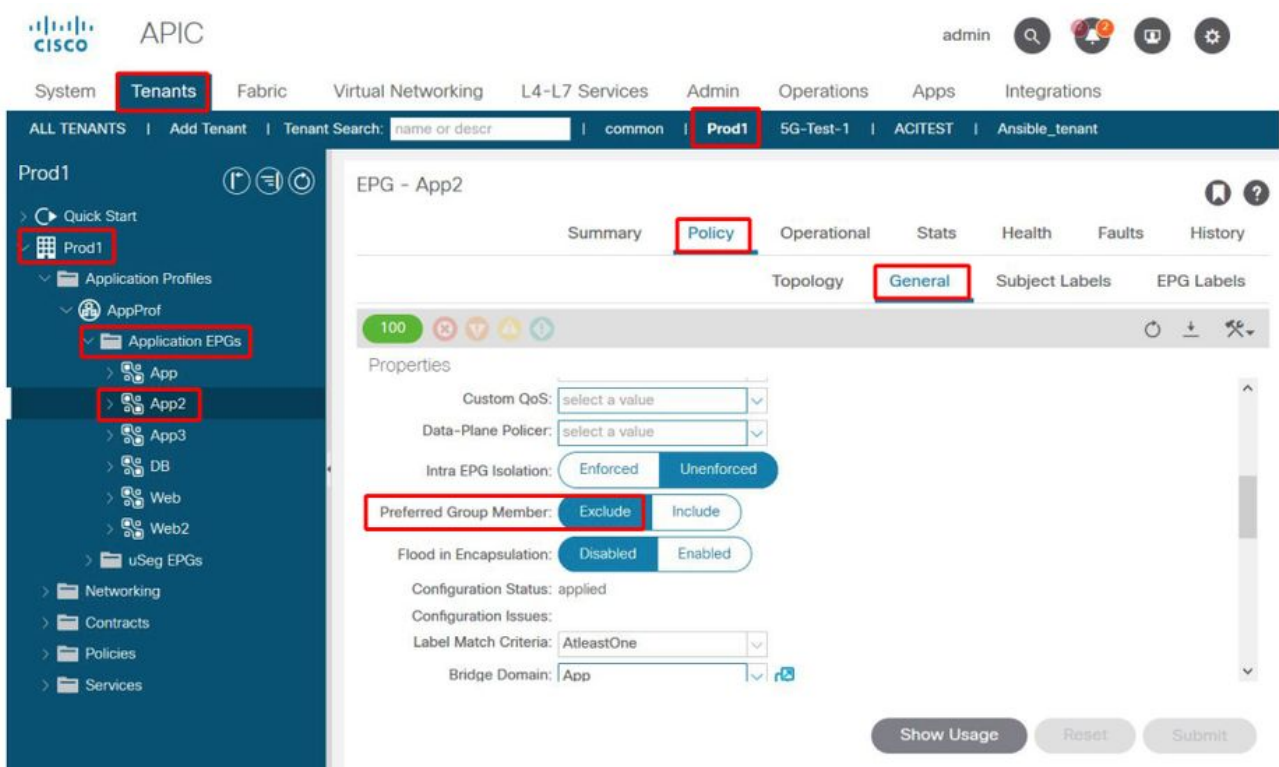
In addition, there are two explicit deny rules observed for pcTag 32775 which is the pcTag of EPG App2. These two explicit deny zoning-rules disallow traffic from any EPG to EPG App2, and vice versa. Those rules have priority 18 and 19, so they will take precedence on the default permit rule.

The conclusion is that EPG App2 is not a Preferred Group Member as the explicit deny rules are observed.

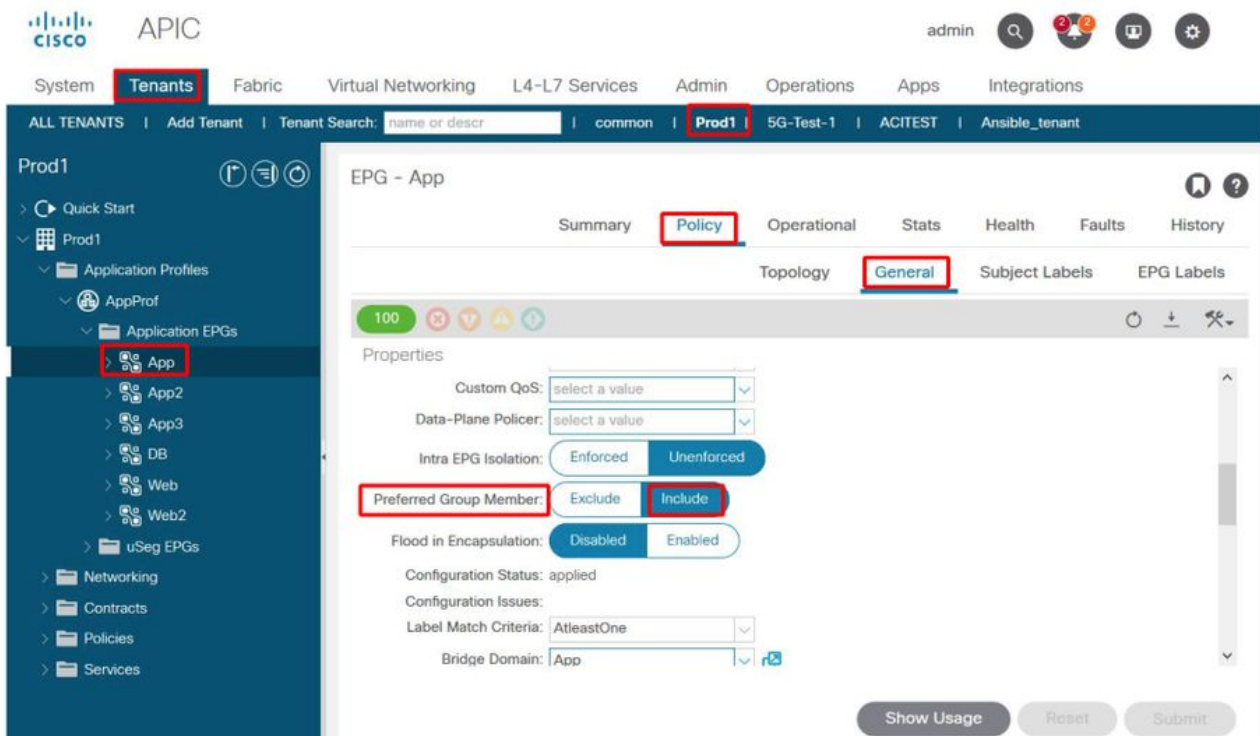
### 3. Verify EPG preferred Group Member Configuration

Navigate the APIC GUI and check EPG App2 and EPG App Preferred Group Member Configuration, In the following figure, see EPG App2 is not configured as a Preferred Group Member.

#### EPG App2 — Preferred Group Member setting excluded



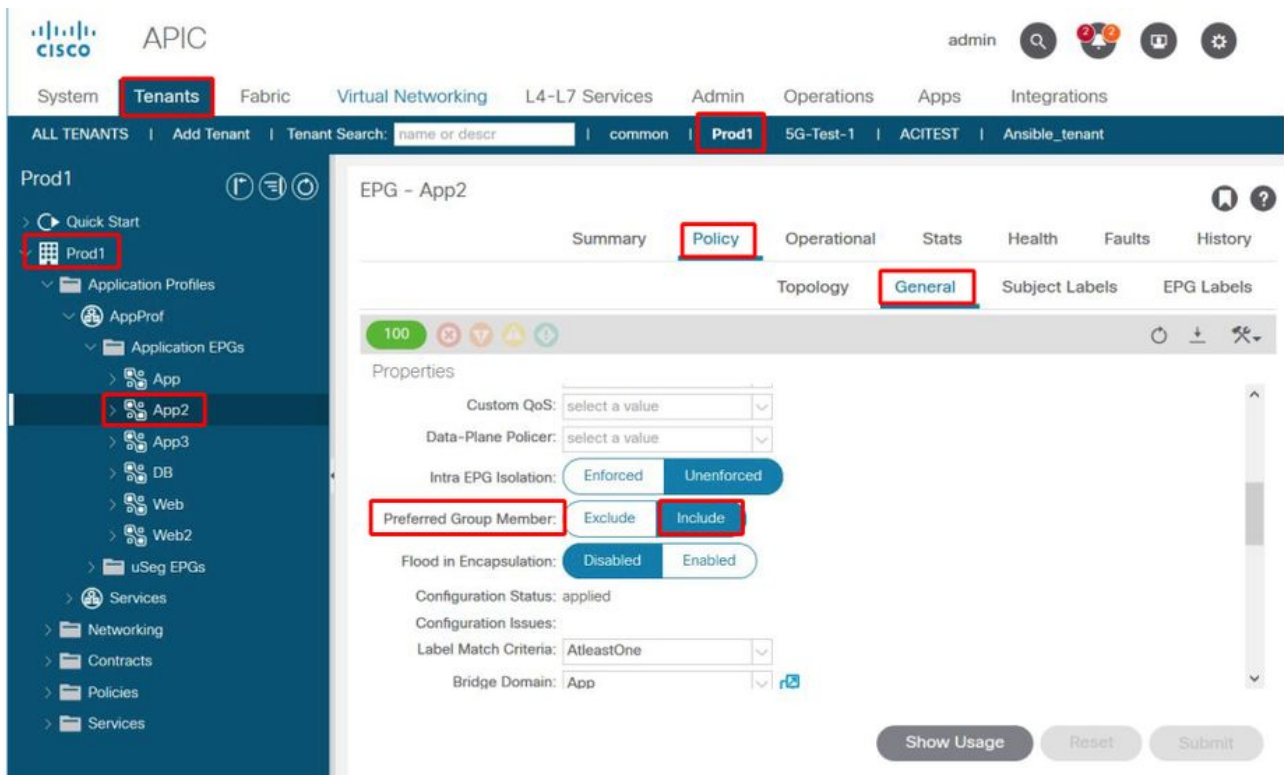
## EPG App — Preferred Group Member setting included



## 4. Set EPG App2 to be a Preferred Group Member

Changing the configuration of App2 EPG enables the preferred group to communicate freely as part of the preferred group.

## EPG App2 — Preferred Group Member setting included



## 5. Re-verify contract programming using contract\_parser.py on the leaf where the src EP resides

Use `contract_parser.py` again and specify the VRF name to verify whether the explicit deny rules for EPG App2 are now gone.

```
fab3-leaf8# contract_parser.py --vrf Prod1:VRF1
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-l4] dst-epg [dst-l4]
[flags][contract:{str}] [hit=count]
[16:4176] [vrf:Prod1:VRF1] permit any epg:any tn-Prod1/bd-App(16386) [contract:implicit] [hit=0]
[16:4160] [vrf:Prod1:VRF1] permit arp epg:any epg:any [contract:implicit] [hit=0]
[18:4175] [vrf:Prod1:VRF1] deny,log any epg:16390 epg:any [contract:implicit] [hit=0]
[18:4167] [vrf:Prod1:VRF1] deny,log any epg:23 epg:any [contract:implicit] [hit=0]
[18:4156] [vrf:Prod1:VRF1] deny,log any tn-Prod1/vrf-VRF1(32770) epg:any [contract:implicit]
[hit=0]
[18:4168] [vrf:Prod1:VRF1] deny,log any epg:49159 epg:any [contract:implicit] [hit=0]
[19:4164] [vrf:Prod1:VRF1] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
[19:4169] [vrf:Prod1:VRF1] deny,log any epg:any epg:16390 [contract:implicit] [hit=0]
[19:4159] [vrf:Prod1:VRF1] deny,log any epg:any epg:23 [contract:implicit] [hit=0]
[19:4174] [vrf:Prod1:VRF1] deny,log any epg:any epg:49159 [contract:implicit] [hit=0]
[20:4165] [vrf:Prod1:VRF1] permit any epg:any epg:any [contract:implicit] [hit=65]
```

The explicit deny rules for EPG App2 and its pcTag 32775 are no longer observed in the above output. This means that traffic between EPs in EPG App and EPG App2 will now match the implicit permit rule — ruleId 4165 — with the highest priority of 20.

## vzAny to EPG

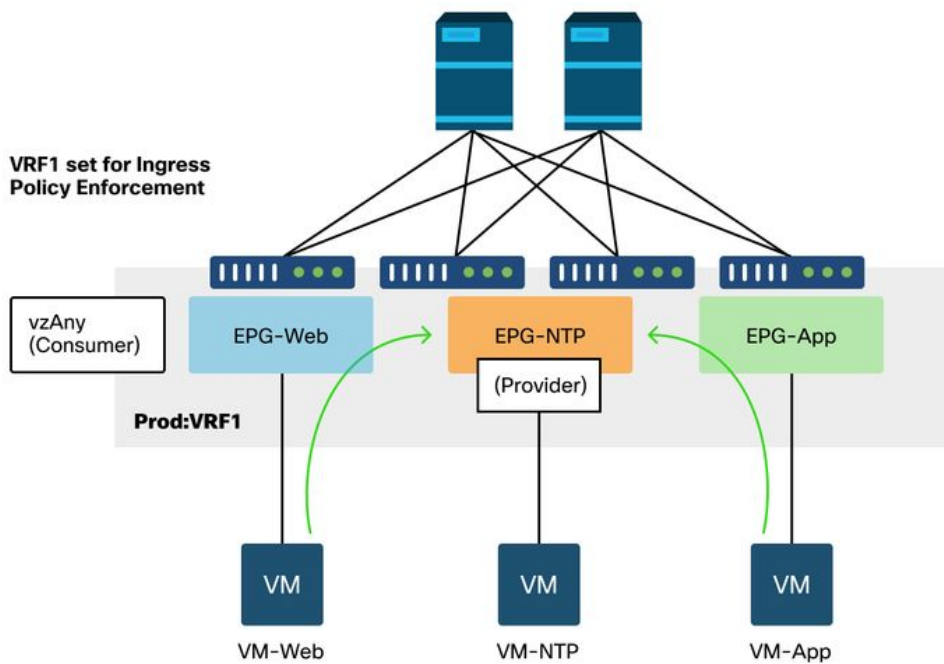
### About vzAny

When configuring contracts between one or multiple EPGs, contracts can either be configured as a consumed or provided relation. When the number of EPGs grows, so can the amount of contract relations between them. Some common use cases require all EPGs to exchange traffic flows with another specific EPG. Such a use case could be an EPG containing EPs providing services that need to be consumed by all other EPGs inside the same VRF (NTP or DNS for example). `vzAny` allows for lower operational overhead in configuring contract relations between all EPGs and specific EPGs providing services to be consumed by all other EPGs. In addition, `vzAny` allows for a much more efficient Security Policy CAM usage on leaf switches as only 2 zoning-rules are added for each `vzAny` contract relation.

### Example Use Case

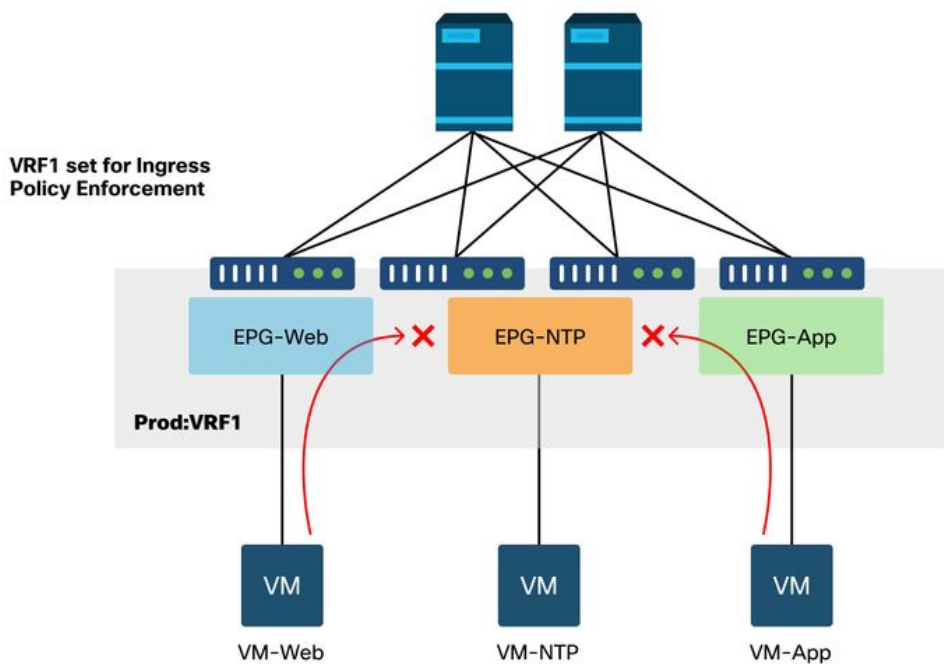
The figure below describes such a use case whereby VM-Web and VM-App in EPGs Web and App respectively need to consume NTP services from VM-NTP in EPG-NTP. Instead of configuring a provided contract on EPG NTP, and subsequently having that same contract as a consumed contract on EPGs Web and App, `vzAny` allows each EPG in VRF Prod:VRF1 to consume NTP services from EPG NTP.

**`vzAny` — Any EPG in VRF Prod:VRF1 can consume NTP services from EPG NTP**



Consider a scenario where drops are observed between EPGs that consume the NTP services when there is no contract between them.

### Troubleshooting Scenario - Traffic drops if there is no contract



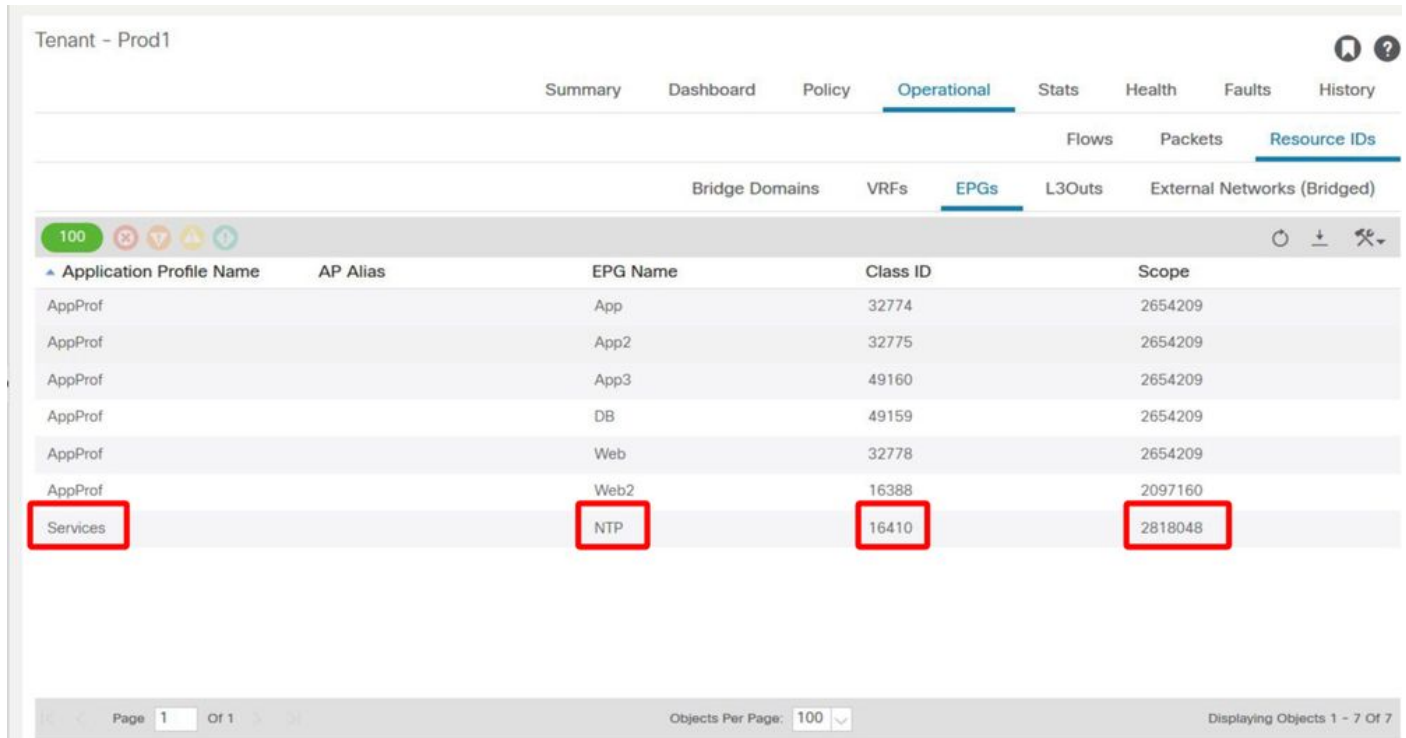
### Workflow



## 1. Look up the pcTag of EPG NTP and its VRF VNID/Scope

'Tenant > Operational > Resource IDs > EPGs' allows finding the pcTag and scope

### EPG NTP pcTag and its VRF VNID/Scope



Tenant - Prod1

Summary Dashboard Policy **Operational** Stats Health Faults History

Flows Packets **Resource IDs**

Bridge Domains VRFs **EPGs** L3Outs External Networks (Bridged)

Application Profile Name	AP Alias	EPG Name	Class ID	Scope
AppProf		App	32774	2654209
AppProf		App2	32775	2654209
AppProf		App3	49160	2654209
AppProf		DB	49159	2654209
AppProf		Web	32778	2654209
AppProf		Web2	16388	2097160
Services		NTP	16410	2818048

Page 1 Of 1 Objects Per Page: 100 Displaying Objects 1 - 7 Of 7

## 2. Verify if a contract is configured as a vzAny consumed contract as part of the VRF

Navigate to the VRF and check if there's a consumed contract configured as vzAny under the 'EPG Collection for VRF'.

### Contract configured as a consumed vzAny contract on the VRF

The screenshot shows the APIC interface for the Prod1 tenant. The left sidebar shows the navigation tree with 'Networking' and 'VRFs' expanded. The main panel shows the configuration for the VRF 'vzAny'. The 'General' tab is selected, and the 'Consumed Contracts' table is visible. The table contains one entry: 'any\_to\_ntp' with Tenant 'Prod1', Type 'Contract', QoS Class 'Unspecified', and State 'formed'.

Name	Tenant	Type	QoS Class	State
any_to_ntp	Prod1	Contract	Unspecified	formed

### 3. Verify if the same contract is applied as a provided contract on EPG NTP

In order to establish a contract relation, the same contract needs to be applied as a provided contract on EPG NTP which is providing NTP services to the other EPGs in its VRF.

The screenshot shows the APIC interface for the Prod1 tenant. The left sidebar shows the navigation tree with 'Contracts' expanded. The main panel shows the configuration for the Contract 'any\_to\_ntp'. The 'Contracts' tab is selected, and the 'Contracts' table is visible. The table contains one entry: 'any\_to\_ntp' with Tenant 'Prod1', Contract Type 'Contract', and a provided contract relationship.

Tenant Name	Tena Alias	Contract Name	Contract Type	Provided / Consumed	QoS Class	State	Label	Subject Label
Prod1		any_to_ntp	Contra...	Provid...	Unspecified	formed		

### 4. Zoning-rule verification on ingress leaf using contract\_parser.py or 'show zoning-rule'

The ingress leaf should have 2 zoning-rules to allow bi-directional traffic flows (if the contract subject is set to allow both directions) between any EPG and EPG NTP. 'Any EPG' is denoted as pcTag 0 in zoning-rule programming.

Using contract\_parser.py or the 'show zoning-rule' commands on the ingress leaf whilst specifying the VRF allows to ensure the zoning-rule are programmed.

### Zoning-rules allowing traffic to/from EPG NTP from other EPGs in the VRF present

Using contract\_parser.py and 'show zoning-rule' to check the presence of the vzAny based zoning-rules.

Here two types of rules are evident:

1. Rule 4156 and Rule 4168 which permit Any to NTP and vice-versa. They have priority 13 and 14: Zoning-rule allowing traffic flows from any EPG (pcTag 0) to EPG NTP (pcTag 49161). Zoning-rule allowing traffic flows from EPG NTP (pcTag 46161) to any other EPG (pcTag 0).
2. Rule 4165 which is the any to any deny rule (default) with priority 21.

Given that lowest priority has precedence, all EPGs of the VRF will have access NTP EPG.

```
fab3-leaf8# contract_parser.py --vrf Prod1:VRF
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-l4] dst-epg [dst-l4]
[flags][contract:{str}] [hit=count]

[13:4156] [vrf:Prod1:VRF1] permit ip tcp tn-Prod1/ap-Services/epg-NTP(49161) eq 123 epg:any
[contract:uni/tn-Prod1/brc-any_to_ntp] [hit=0]
[14:4168] [vrf:Prod1:VRF1] permit ip tcp epg:any tn-Prod1/ap-Services/epg-NTP(49161) eq 123
[contract:uni/tn-Prod1/brc-any_to_ntp] [hit=0]
[16:4176] [vrf:Prod1:VRF1] permit any epg:any tn-Prod1/bd-App(16386) [contract:implicit] [hit=0]
[16:4174] [vrf:Prod1:VRF1] permit any epg:any tn-Prod1/bd-Services(32776) [contract:implicit]
[hit=0]
[16:4160] [vrf:Prod1:VRF1] permit arp epg:any epg:any [contract:implicit] [hit=0]
[21:4165] [vrf:Prod1:VRF1] deny,log any epg:any epg:any [contract:implicit] [hit=65]
[22:4164] [vrf:Prod1:VRF1] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
```

```
fab3-leaf8# show zoning-rule scope 2654209
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Rule ID | SrcEPG | DstEPG | FilterID | Dir | operSt | Scope | Name | Action |
| Priority | | | | | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 4165 | 0 | 0 | implicit | uni-dir | enabled | 2654209 | | deny,log |
any_any_any(21) |
| 4160 | 0 | 0 | implarp | uni-dir | enabled | 2654209 | | permit |
any_any_filter(17) |
| 4164 | 0 | 15 | implicit | uni-dir | enabled | 2654209 | | deny,log |
any_vrf_any_deny(22) |
| 4176 | 0 | 16386 | implicit | uni-dir | enabled | 2654209 | | permit |
any_dest_any(16) |
| 4174 | 0 | 32776 | implicit | uni-dir | enabled | 2654209 | | permit |
any_dest_any(16) |
| 4168 | 0 | 49161 | 424 | uni-dir | enabled | 2654209 | any_to_ntp | permit |
any_dest_filter(14) |
| 4156 | 49161 | 0 | 425 | uni-dir | enabled | 2654209 | any_to_ntp | permit |
```



Tenant - Prod1

Summary Dashboard Policy **Operational** Stats Health Faults History

Flows Packets **Resource IDs**

Bridge Domains VRFs **EPGs** L3Outs External Networks (Bridged)

Application Profile Name	AP Alias	EPG Name	Class ID	Scope
AppProf		App	32774	2654209
AppProf		App2	32775	2654209
AppProf		App3	49160	2654209
AppProf		DB	49159	2654209
AppProf		Web	32778	2654209
AppProf		Web2	16388	2097160
Services		NTP	16410	2818048

Page 1 Of 1 Objects Per Page: 100 Displaying Objects 1 - 7 Of 7

## 2. Verify the pcTag and VRF VNID/Scope for the provider L3Out EPG

As noted in Step 1, the provider L3Out EPG has a global range pcTag as prefixes from L3Out which are leaked into the consumer VRF. As a result, the L3Out EPG pcTag is required to not overlap with pcTags in the consumer VRF, and so it is within the global pcTag range.

### pcTag of provider external EPG

Tenant - Prod1

Summary Dashboard Policy **Operational** Stats Health Faults History

Flows Packets **Resource IDs**

Bridge Domains VRFs EPGs **L3Outs** External Networks (Bridged)

EPG Name	EPG Alias	Class ID	Scope
extEpg		25	2719752

Page 1 Of 1 Objects Per Page: 100 Displaying Objects 1 - 1 Of 1

## 3. Verify the consumer EPG has either an imported tenant scoped contract or global contract configured

The consumer EPG NTP with subnet defined under the EPG/BD is consuming the 'tenant' or 'global' scoped contract

## Contract consumed by EPG

The screenshot shows the Cisco APIC interface. The top navigation bar includes 'System', 'Tenants', 'Fabric', 'Virtual Networking', 'L4-L7 Services', 'Admin', 'Operations', 'Apps', and 'Integrations'. The 'Tenants' tab is selected, and the 'Prod1' tenant is chosen from the breadcrumb navigation. The left sidebar shows the 'Prod1' configuration tree, with 'Contracts' highlighted. The main content area displays the 'Contracts' page for the 'Prod1' tenant. A table lists the contracts, with one contract named 'external\_to\_ntp' under the 'Contract Type: Contract' section. The table columns are: Tenant Name, Tenant Alias, Contract Name, Contract Type, Provided / Consumed, QoS Class, State, Label, and Sub Lab.

Tenant Name	Tenant Alias	Contract Name	Contract Type	Provided / Consumed	QoS Class	State	Label	Sub Lab
Prod1		external_to_ntp	Contract	Consumed	Unspecified	form...		

### 4. Verify whether the BD of the consumer EPG has a subnet configured with its scope set to 'Shared between VRFs'

The subnet of the EPG is configured under the bridge domain but must have the 'shared between VRF' flag (to allow routed leaking) and the 'advertised externally' flag (to allow to advertise to L3Out)

### 5. Verify the provider L3Out EPG has either an imported tenant scoped contract or global contract configured

The L3Out EPG should either have a tenant scoped contract or global contract configured as a provided contract.

## Contract on provider L3Out

The screenshot displays the Cisco APIC interface for configuring an External EPG Instance Profile. The navigation tree on the left shows the path: **Prod1** > **L3Outs** > **L3Out1** > **External EPGs** > **extEpg**. The main configuration page for 'External EPG Instance Profile - extEpg' is shown, with the 'Policy' tab selected. Under the 'Contracts' sub-tab, the 'Provided Contracts' section contains the following table:

Name	Tenant	Type	QoS Class	Match Type	State
external_to_ntp	Prod1	Contract	Unspecified	AtleastOne	formed

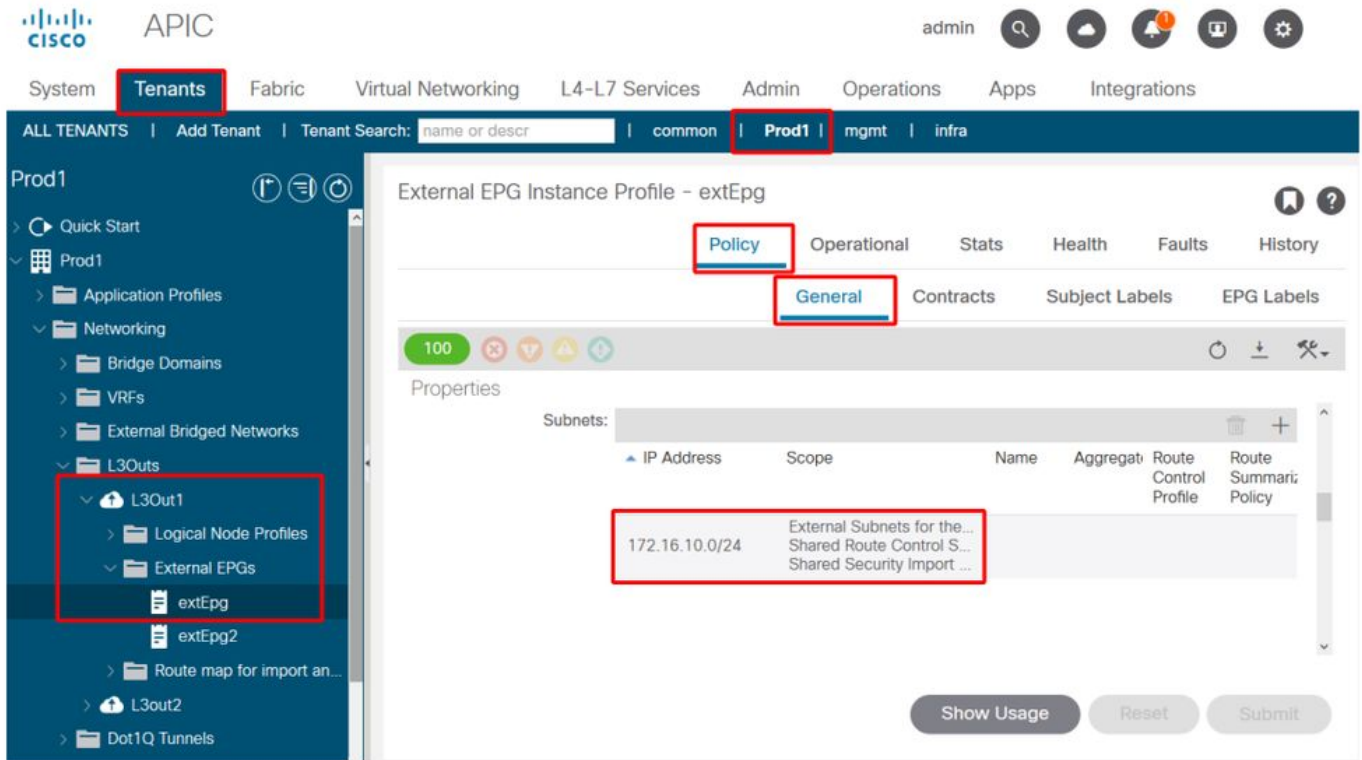
## 6. Verify if the provider L3Out EPG has a subnet configured with the necessary scopes checked

The provider L3Out EPG should have the to-be-leaked prefix configured with the following scopes:

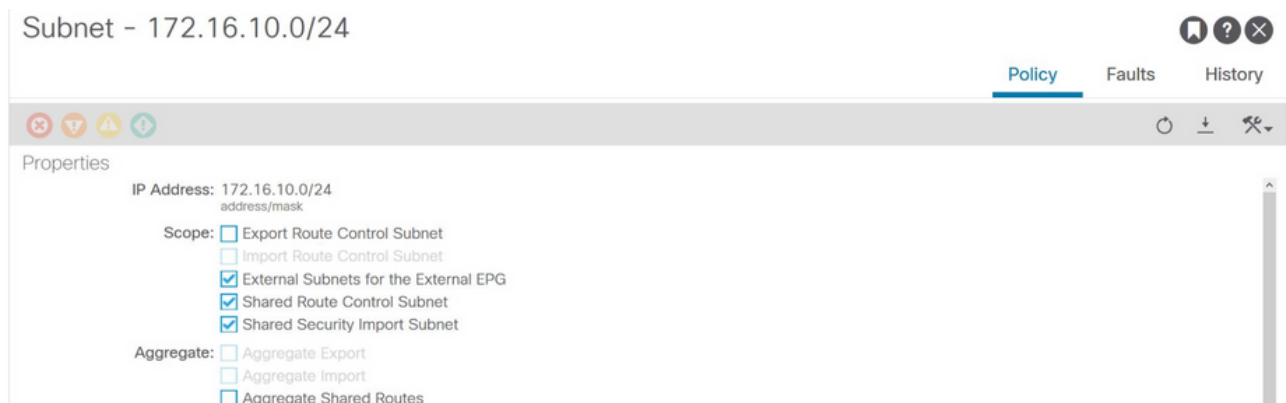
- External subnets for the external EPG.
- Shared route control subnet.
- Shared security import subnet.

For more detail on subnet flag in L3Out EPG refer to the "External forwarding" chapter.

### External EPG subnet settings



## External EPG subnet settings expanded



## 7. Verify the pcTag of L3Out EPG subnet on the non-BL for the consumer VRF

When traffic destined to the external EPG subnet ingresses the non-BL, a lookup is performed against the destination prefix to determine the pcTag. This can be checked using the following command on the non-BL.

Note this output is taken in the scope of the VNI 2818048 which is the consumer VRF VNID. By looking at the table the consumer can find the pcTag of the destination, even though it is not in the same VRF.

```
fab3-leaf8# vsh -c 'show system internal policy-mgr prefix' | egrep 'Vrf-Vni|==|common:default'
Vrf-Vni Vrf-Id Table-Id Table-State VRF-Name
Addr Class Shared Remote Complete
=====
=====
2818048 19 0x13 Up common:default
0.0.0.0/0 15 False False False
2818048 19 0x80000013 Up common:default
```



```

::/0 15 False False False
2818048 19 0x13 Up common:default
172.16.10.0/24 25 True True False

```

The above output shows the combination of the L3Out EPG subnet and its global pcTag 25.

## 8. Verify the programmed zoning-rules on the non-BL for the consumer VRF

Use either 'contract\_parser.py' or the 'show zoning-rule' command and specify the VRF.

Below command outputs display two zoning-rules are installed to allow traffic from the consumer EPG local pcTag 16410 to the L3Out EPG global pcTag 25. This is in the scope 2818048, which is the scope of the consumer VRF.

```
fab3-leaf8# show zoning-rule scope 2818048
```

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name
4174	0	0	implarp	uni-dir	enabled	2818048	
4168	0	15	implicit	uni-dir	enabled	2818048	
4167	0	32789	implicit	uni-dir	enabled	2818048	
4159	0	0	implicit	uni-dir	enabled	2818048	
4169	25	0	implicit	uni-dir	enabled	2818048	
4156	25	16410	425	uni-dir-ignore	enabled	2818048	external_to_ntp
4131	16410	25	424	bi-dir	enabled	2818048	external_to_ntp

```
fab3-leaf8# contract_parser.py --vrf common:default
```

Key:

```
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-14] dst-epg [dst-14]
[flags][contract:{str}] [hit=count]
```

```

[7:4131] [vrf:common:default] permit ip tcp tn-Prod1/ap-Services/epg-NTP(16410) tn-Prod1/l3out-L3Out1/instP-extEpg(25) eq 123 [contract:uni/tn-Prod1/brc-external_to_ntp] [hit=0]
[7:4156] [vrf:common:default] permit ip tcp tn-Prod1/l3out-L3Out1/instP-extEpg(25) eq 123 tn-Prod1/ap-Services/epg-NTP(16410) [contract:uni/tn-Prod1/brc-external_to_ntp] [hit=0]
[12:4169] [vrf:common:default] deny,log any tn-Prod1/l3out-L3Out1/instP-extEpg(25) epg:any [contract:implicit] [hit=0]
[16:4167] [vrf:common:default] permit any epg:any tn-Prod1/bd-Services(32789) [contract:implicit] [hit=0]
[16:4174] [vrf:common:default] permit arp epg:any epg:any [contract:implicit] [hit=0]
[21:4159] [vrf:common:default] deny,log any epg:any epg:any [contract:implicit] [hit=0]
[22:4168] [vrf:common:default] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]

```

## 9. Verify the programmed zoning-rules on the BL for the provider VRF

Use either 'contract\_parser.py' or the 'show zoning-rule' command and specify the VRF. The following command outputs show that there are **NO** specific zoning-rules in the provider VRF as outlined multiple times before.

It is in the scope 2719752 which is the scope of provider VRF.

```
border-leaf# show zoning-rule scope 2719752
```

Rule ID	SrcEPG	DstEPG	FilterID	Dir	operSt	Scope	Name
4134	10937	24	default	uni-dir-ignore	enabled	2719752	vrf1_to_vrf2
4135	24	10937	default	bi-dir	enabled	2719752	vrf1_to_vrf2
4131	0	0	implicit	uni-dir	enabled	2719752	
4130	0	0	implarp	uni-dir	enabled	2719752	
4132	0	15	implicit	uni-dir	enabled	2719752	

```
border-leaf# contract_parser.py --vrf Prod1:VRF3
```

```
Key:
[prio:RuleId] [vrf:{str}] action protocol src-epg [src-l4] dst-epg [dst-l4]
[flags][contract:{str}] [hit=count]

[9:4134] [vrf:Prod1:VRF3] permit any tn-Prod1/l3out-L3Out1/instP-extEpg2(10937) tn-Prod1/l3out-L3Out2/instP-extEpg2(24) [contract:uni/tn-Prod1/brc-vrf1_to_vrf2] [hit=0]
[9:4135] [vrf:Prod1:VRF3] permit any tn-Prod1/l3out-L3Out2/instP-extEpg2(24) tn-Prod1/l3out-L3Out1/instP-extEpg2(10937) [contract:uni/tn-Prod1/brc-vrf1_to_vrf2] [hit=0]
[16:4130] [vrf:Prod1:VRF3] permit arp epg:any epg:any [contract:implicit] [hit=0]
[21:4131] [vrf:Prod1:VRF3] deny,log any epg:any epg:any [contract:implicit] [hit=0]
[22:4132] [vrf:Prod1:VRF3] deny,log any epg:any pfx-0.0.0.0/0(15) [contract:implicit] [hit=0]
```