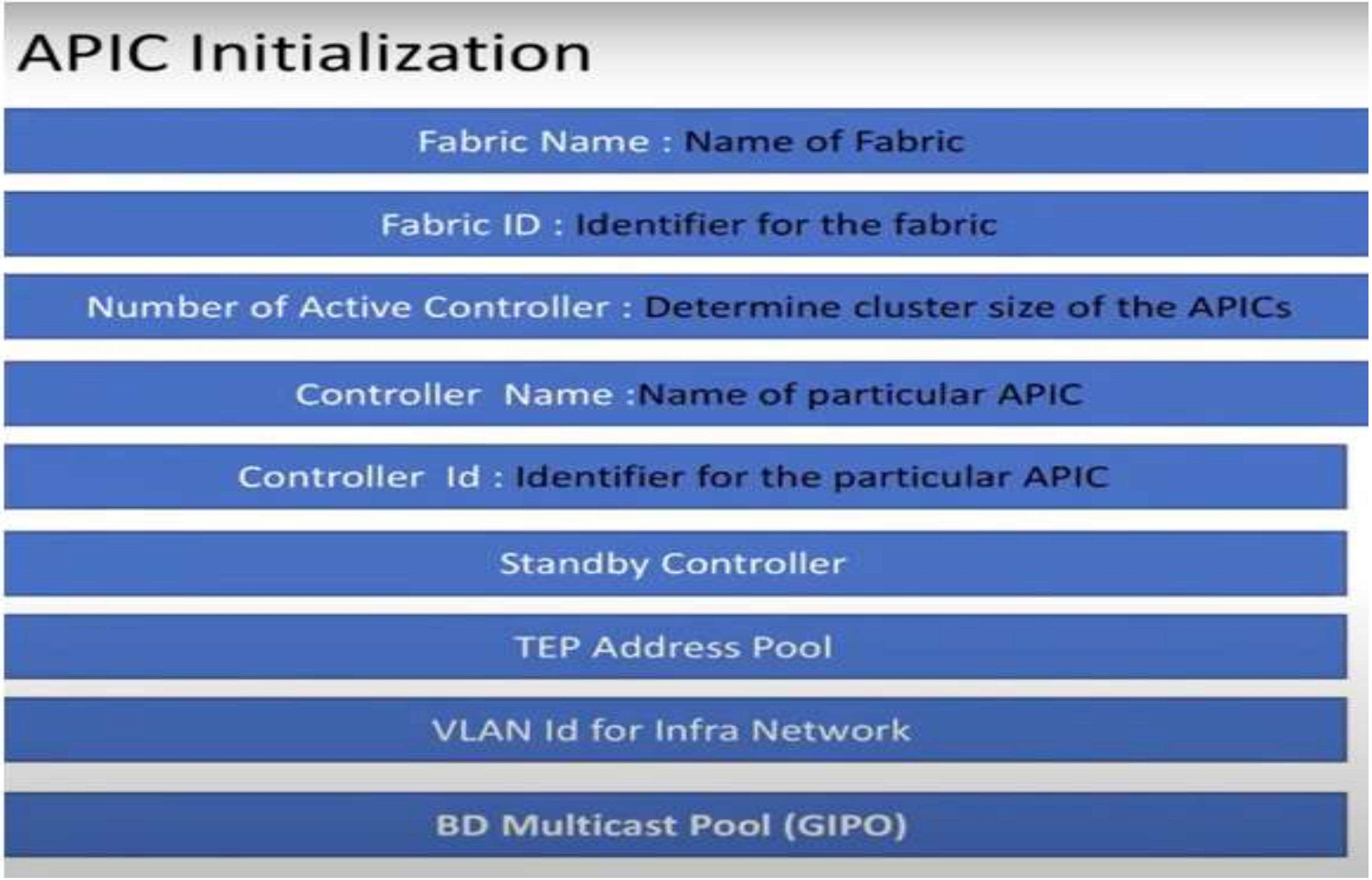


APIC Initialization

Fabric Name : Name of Fabric

Fabric ID : Identifier for the fabric

Number of Active Controller : Determine cluster size of the APICs

Controller Name :Name of particular APIC

Controller Id : Identifier for the particular APIC

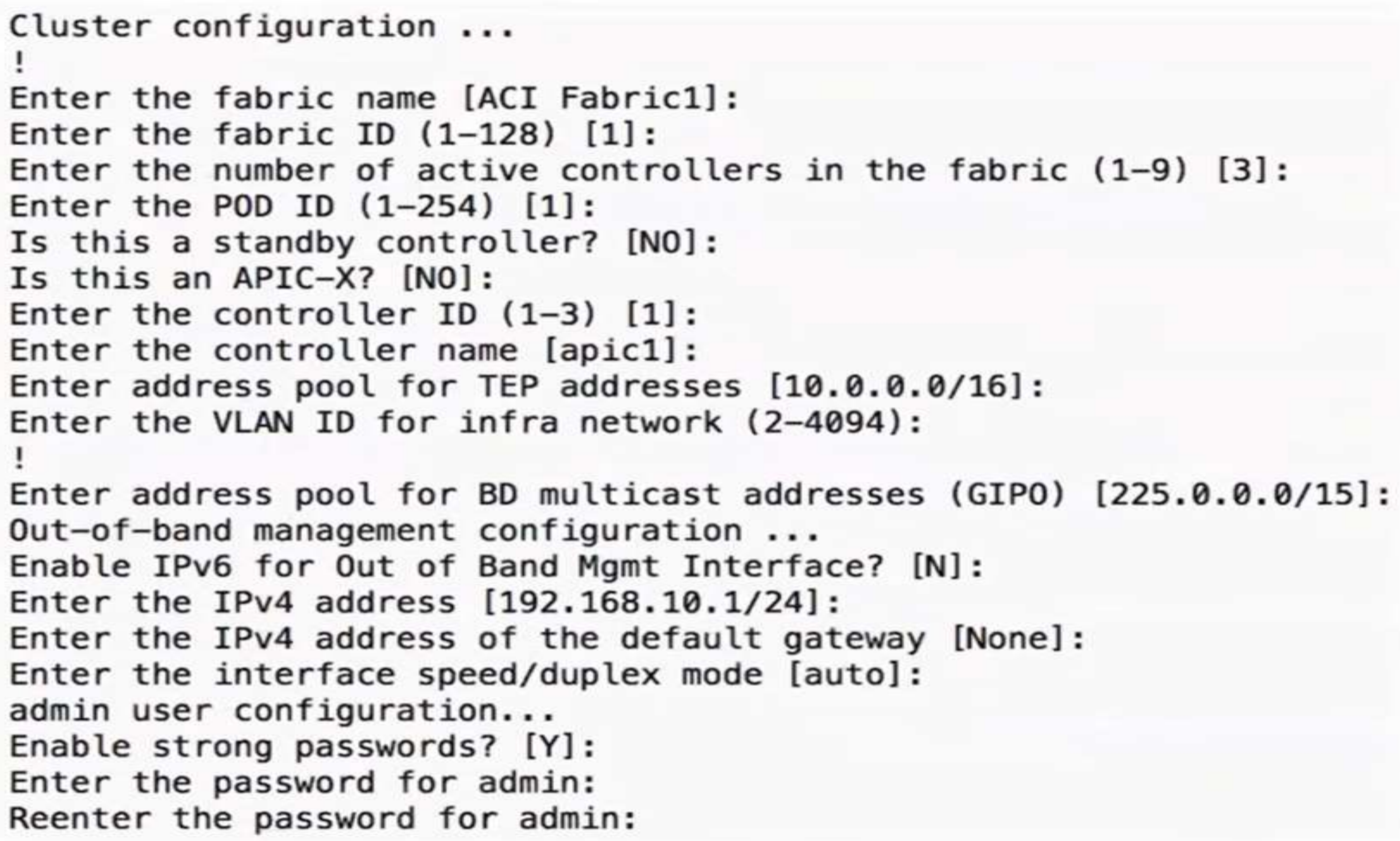
Standby Controller

TEP Address Pool

VLAN Id for Infra Network

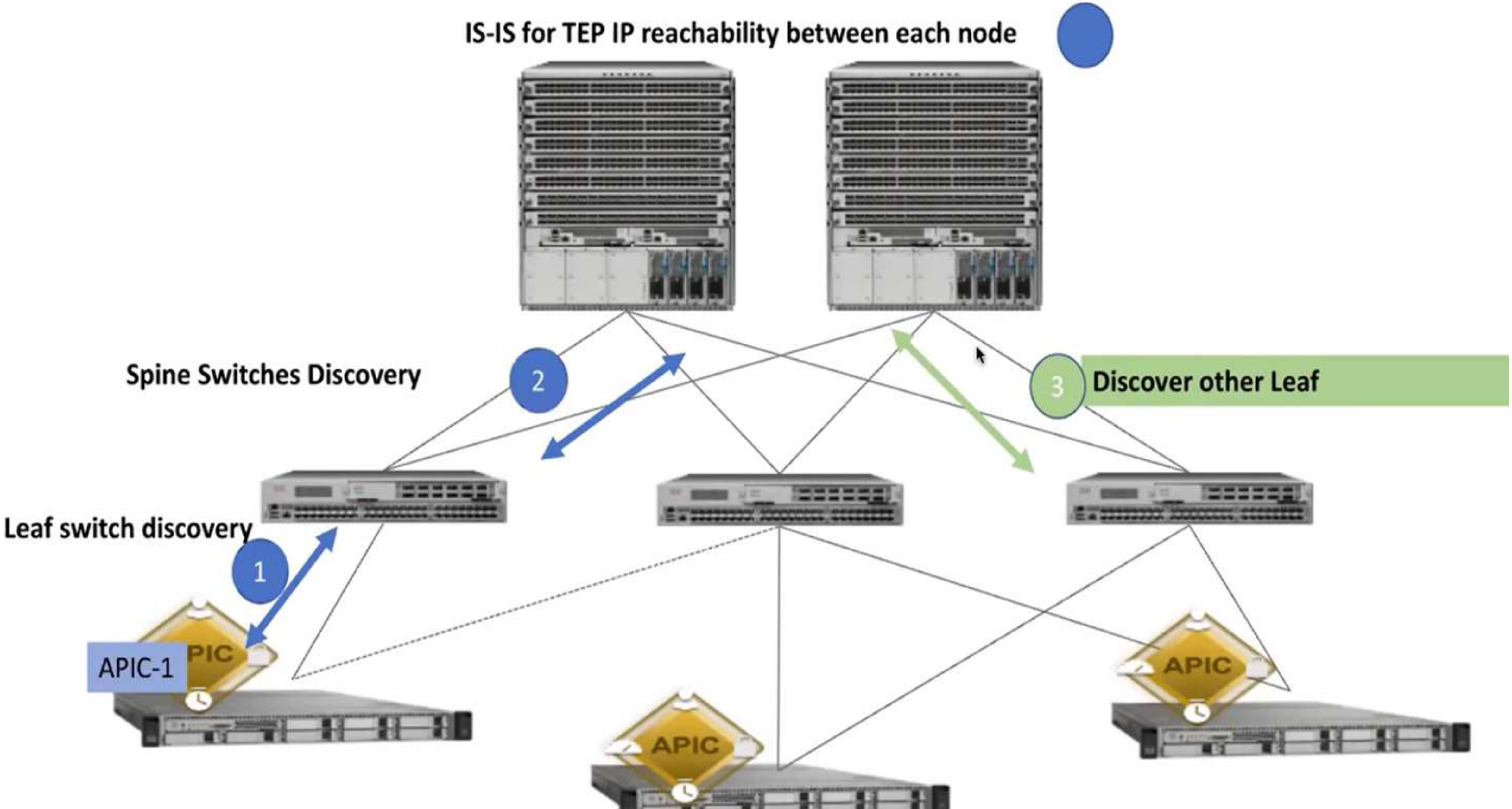
BD Multicast Pool (GIPO)

APIC Initialization

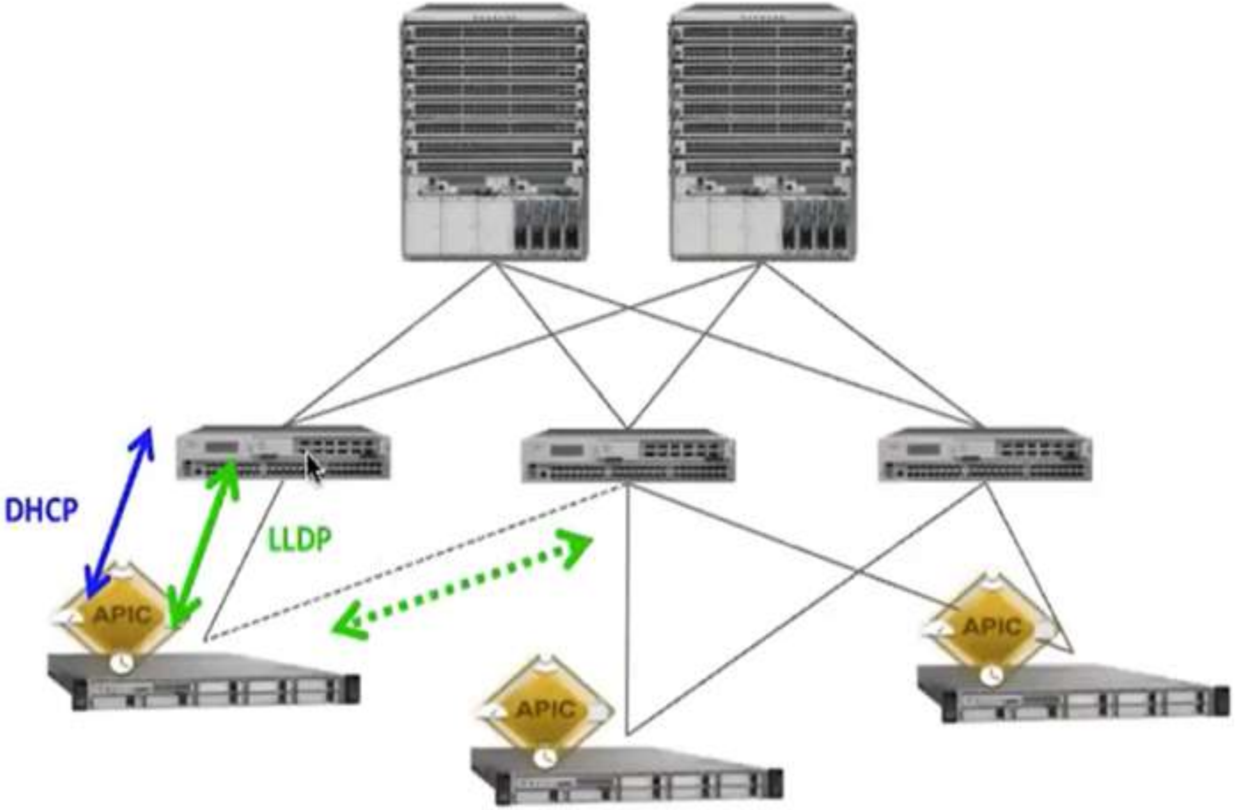


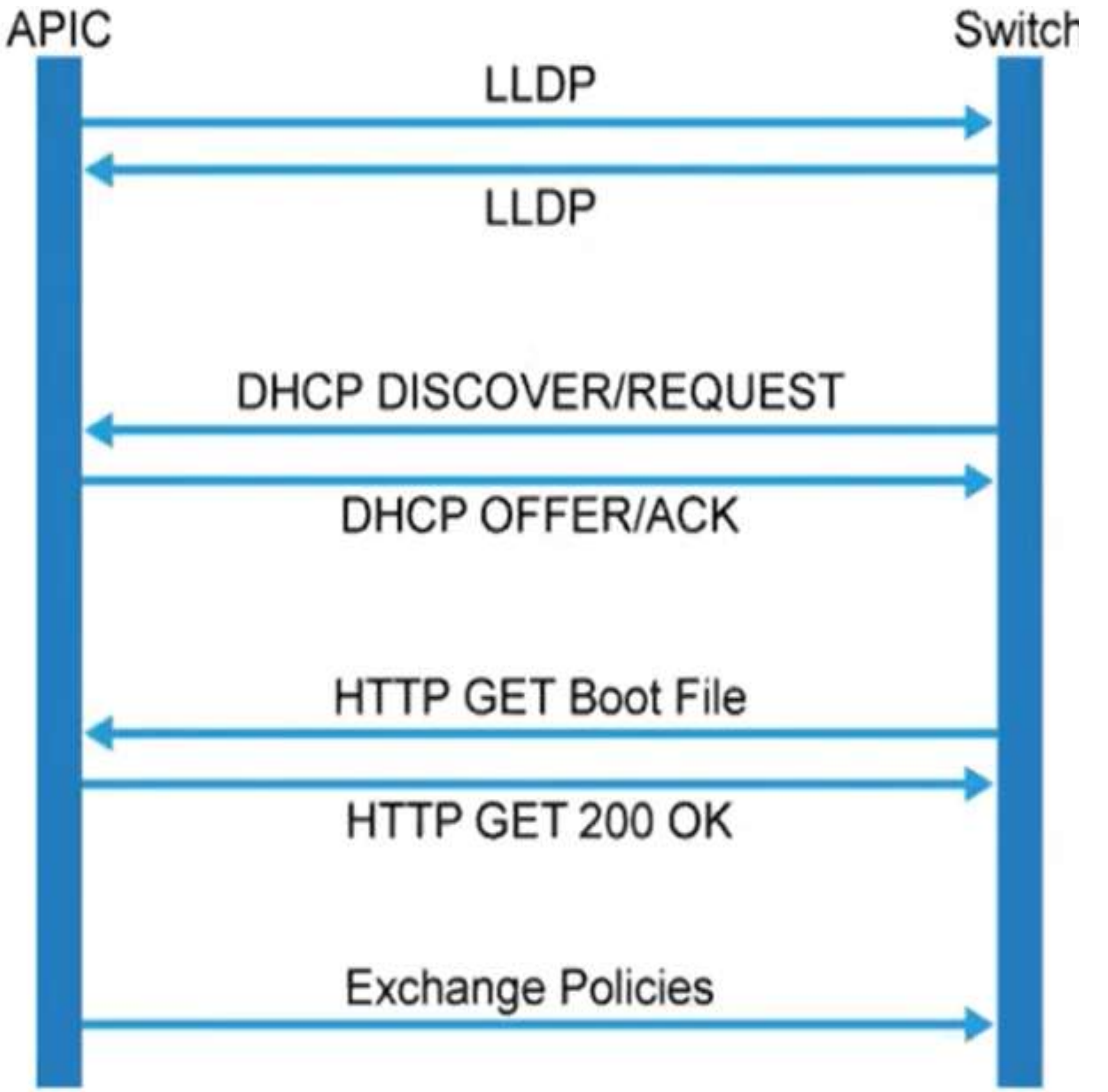
```
Cluster configuration ...
!
Enter the fabric name [ACI Fabric1]:
Enter the fabric ID (1-128) [1]:
Enter the number of active controllers in the fabric (1-9) [3]:
Enter the POD ID (1-254) [1]:
Is this a standby controller? [NO]:
Is this an APIC-X? [NO]:
Enter the controller ID (1-3) [1]:
Enter the controller name [apic1]:
Enter address pool for TEP addresses [10.0.0.0/16]:
Enter the VLAN ID for infra network (2-4094):
!
Enter address pool for BD multicast addresses (GIPO) [225.0.0.0/15]:
Out-of-band management configuration ...
Enable IPv6 for Out of Band Mgmt Interface? [N]:
Enter the IPv4 address [192.168.10.1/24]:
Enter the IPv4 address of the default gateway [None]:
Enter the interface speed/duplex mode [auto]:
admin user configuration...
Enable strong passwords? [Y]:
Enter the password for admin:
Reenter the password for admin:
```

Cisco ACI Fabric Discovery : Leaf and Spine Switches

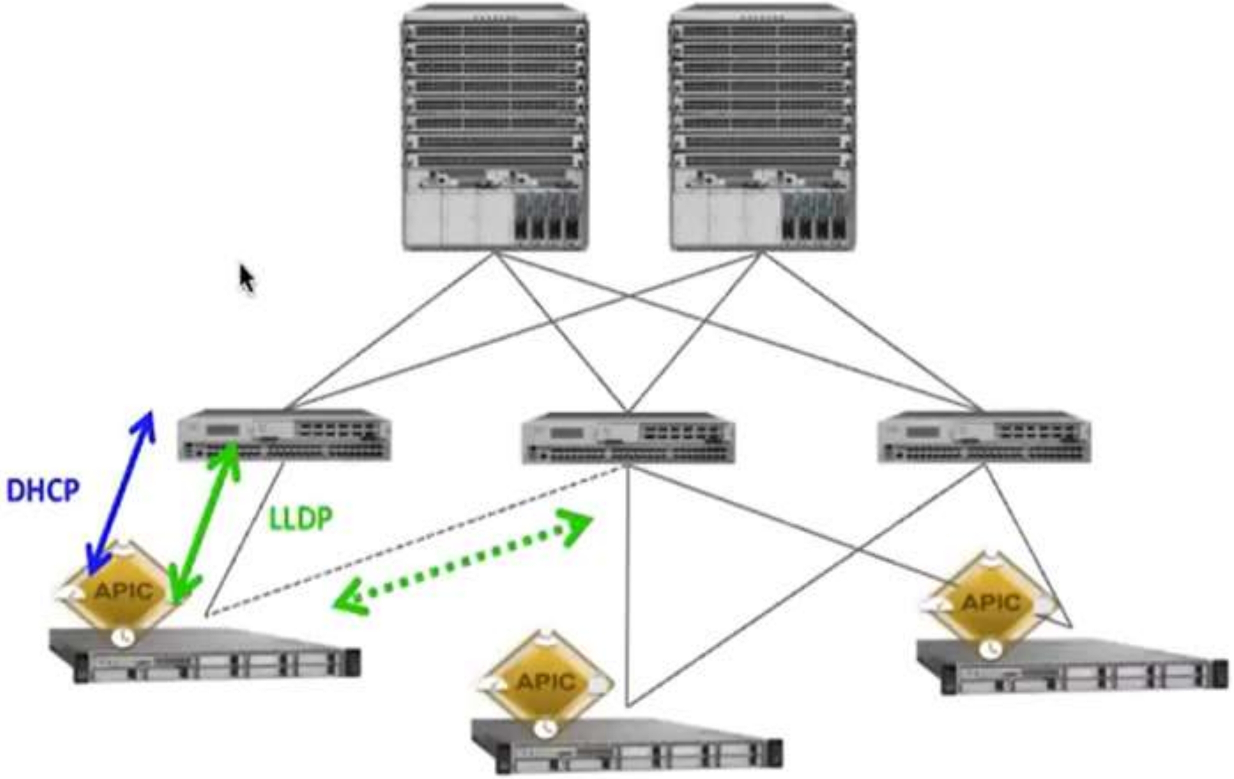


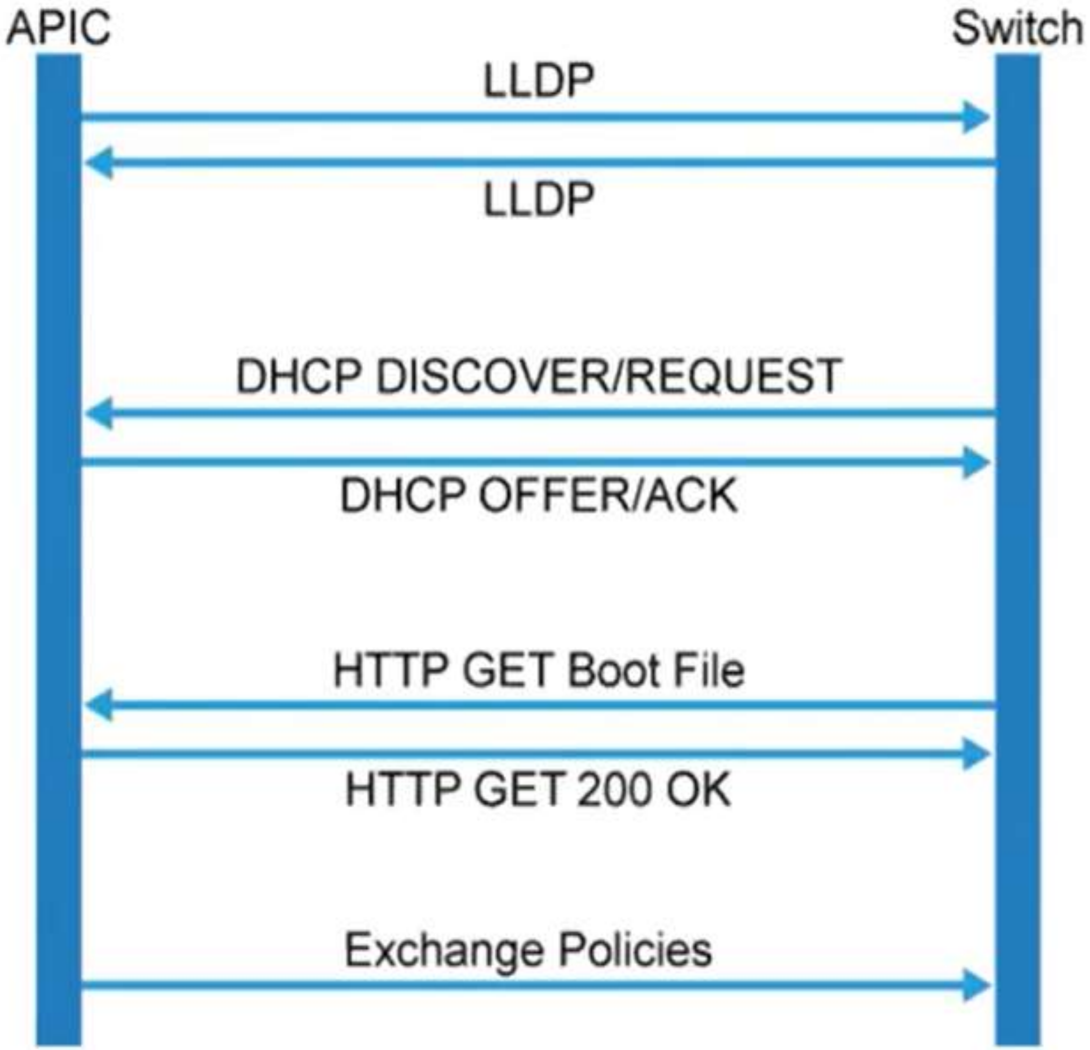
1. First Leaf Switch Discovery





1. First Leaf Switch Discovery





APIC

Switch

LLDP

LLDP

DHCP DISCOVER/REQUEST

DHCP OFFER/ACK

HTTP GET Boot File

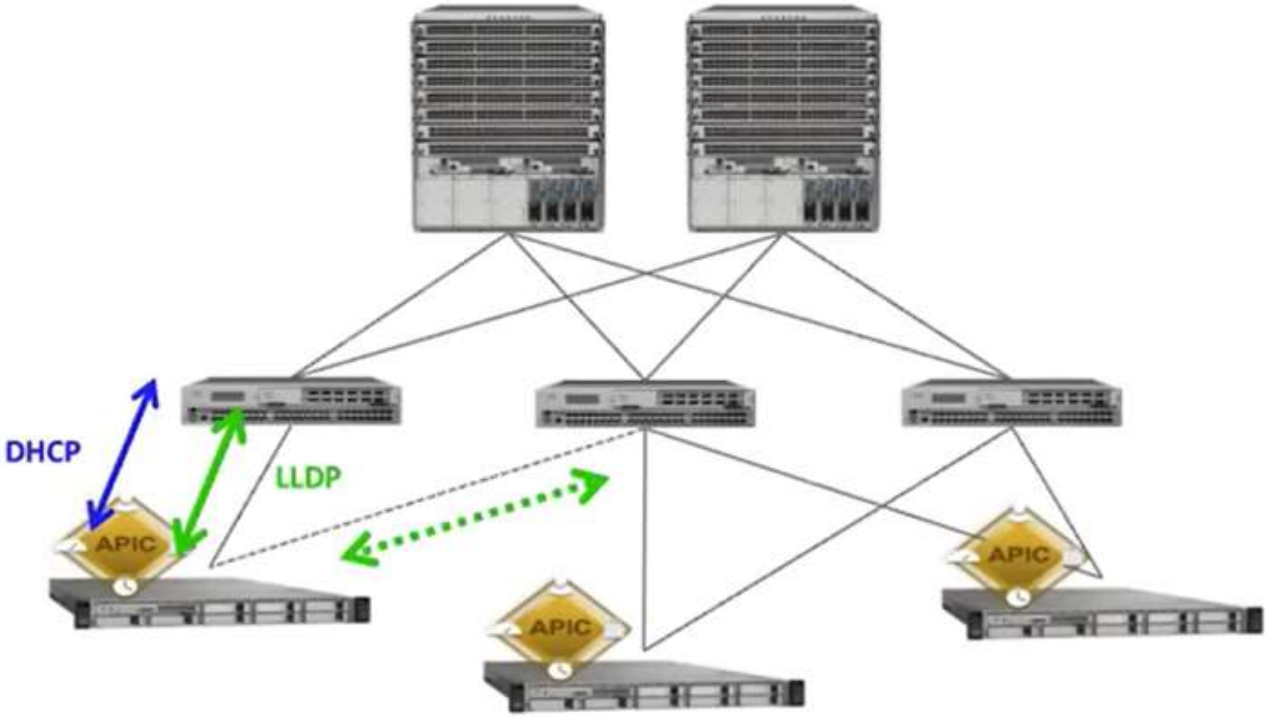
HTTP GET 200 OK

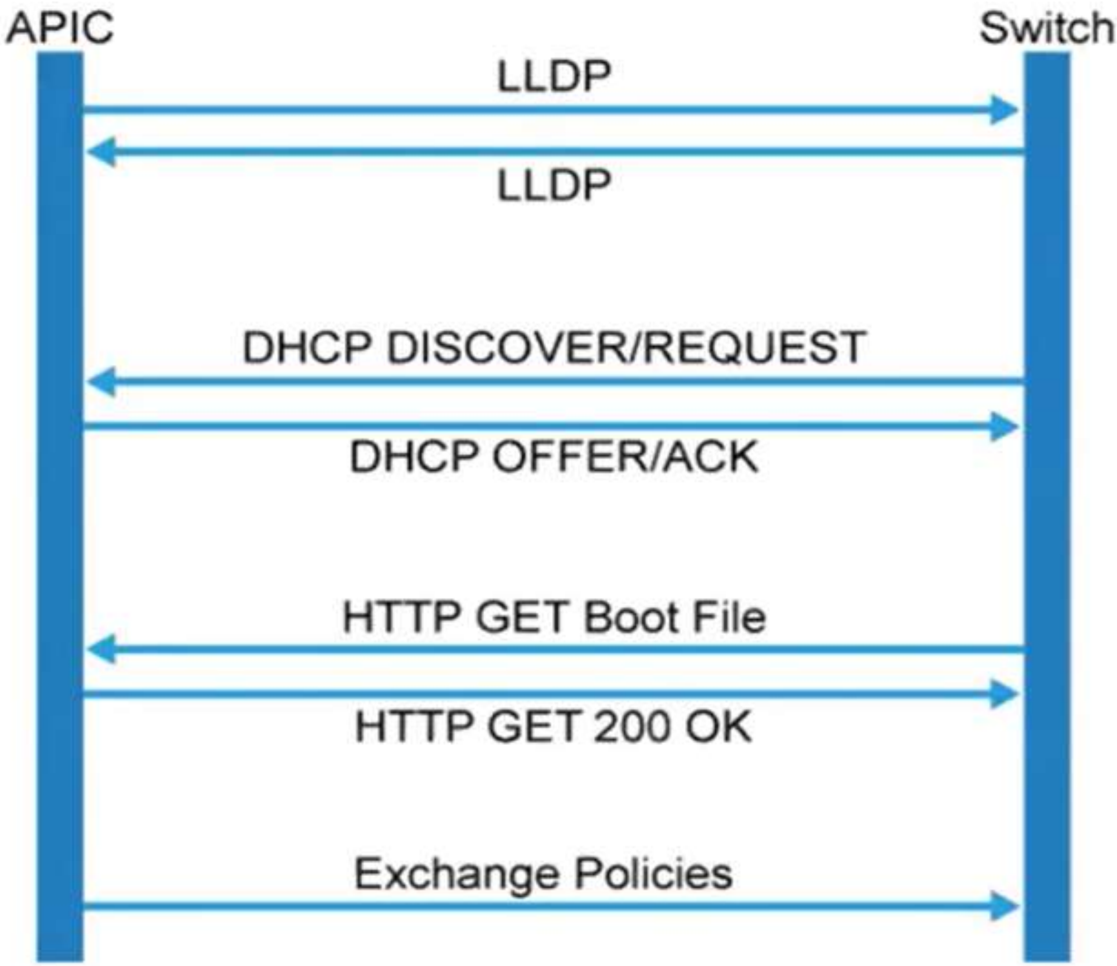
Exchange Policies

1. Cisco APIC uses LLDP neighbor discovery to discover a switch.

2. After a successful discovery, the switch sends a request for an IP address via DHCP.

1. First Leaf Switch Discovery



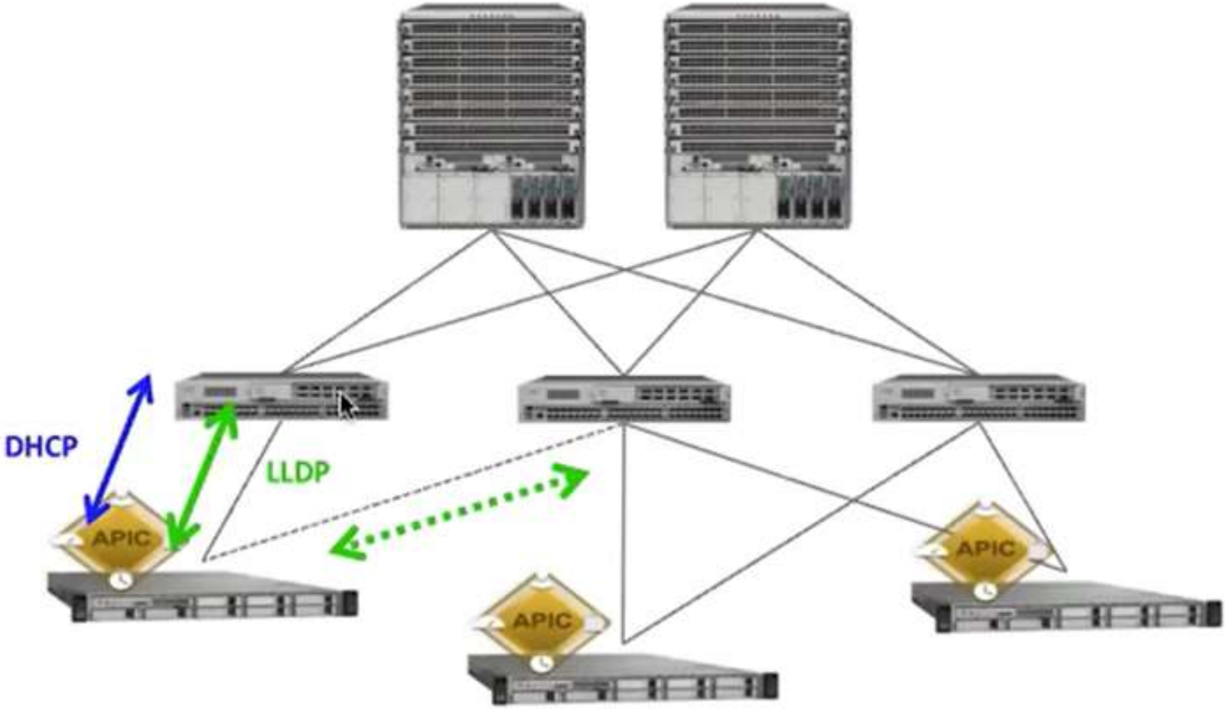


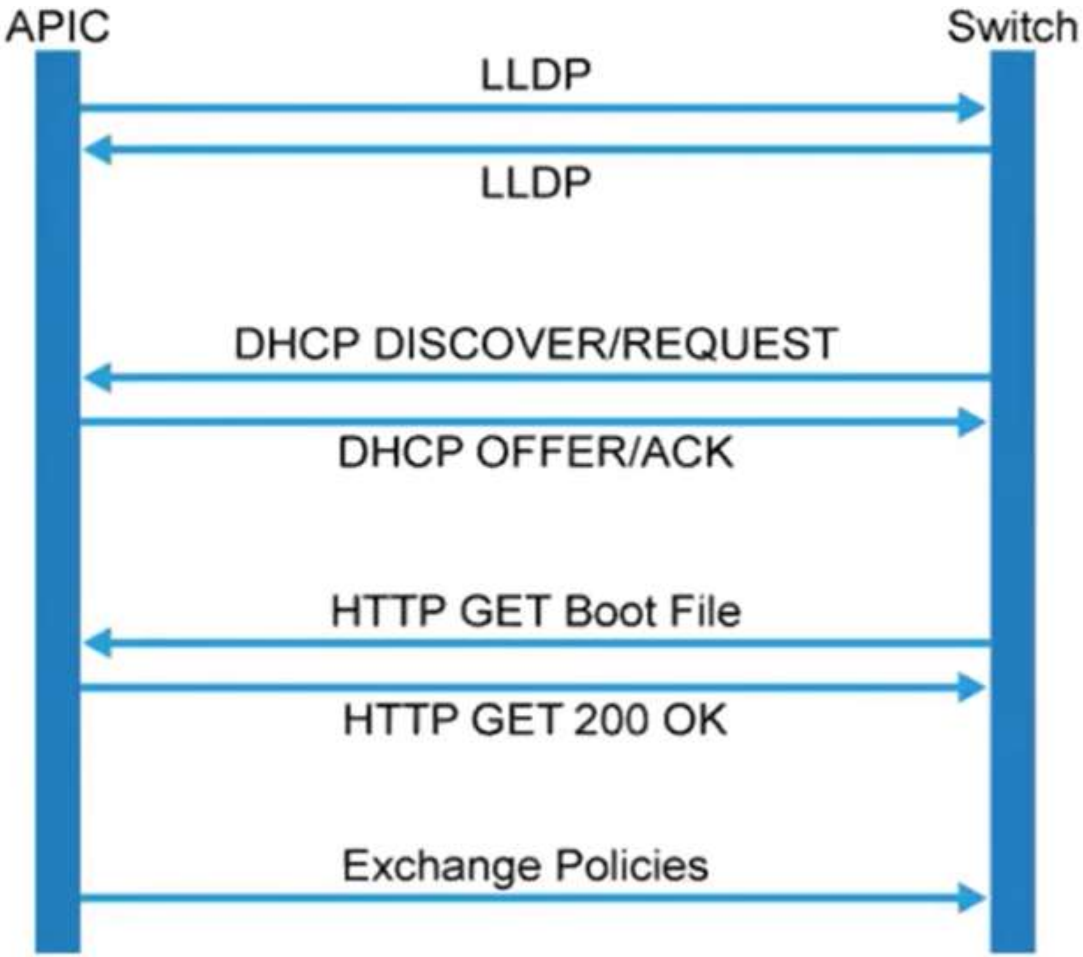
1. Cisco APIC uses LLDP neighbor discovery to discover a switch.

2. After a successful discovery, the switch sends a request for an IP address via DHCP.

3. To register Leaf to the Fabric, **Manual Register** from APIC is required after auto-discovery where we define **Node Id , POD Id, Role, Rack Name and Node Name**.

1. First Leaf Switch Discovery





APIC

Switch

LLDP

LLDP

DHCP DISCOVER/REQUEST

DHCP OFFER/ACK

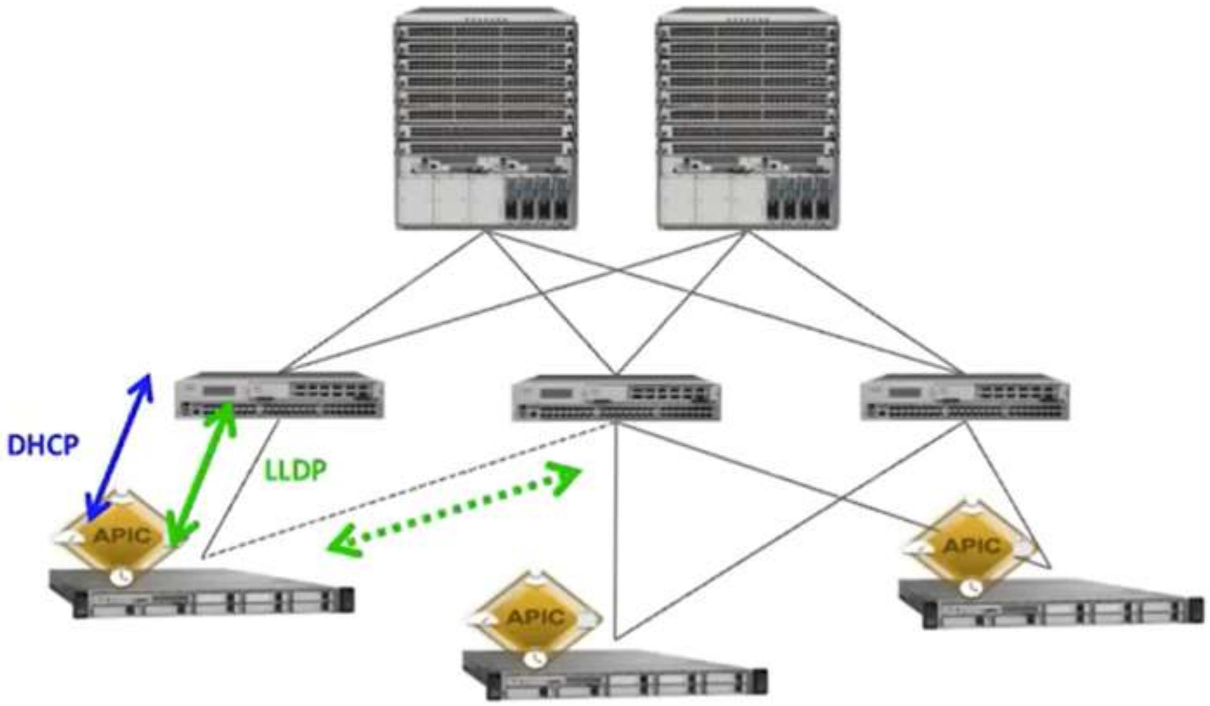
HTTP GET Boot File

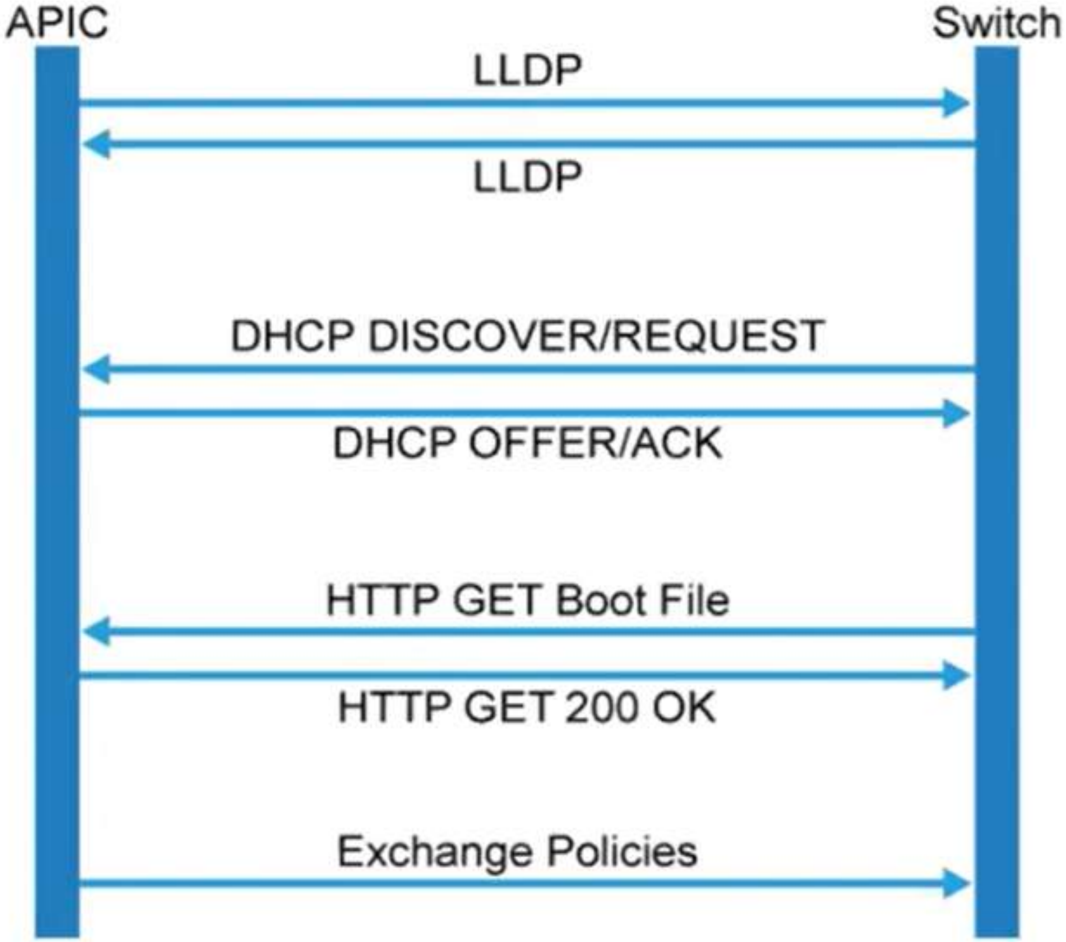
HTTP GET 200 OK

Exchange Policies

1. Cisco APIC uses LLDP neighbor discovery to discover a switch.
2. After a successful discovery, the switch sends a request for an IP address via DHCP.
3. To register Leaf to the Fabric, **Manual Register** from APIC is required after auto-discovery where we define **Node Id , POD Id, Role, Rack Name and Node Name**.
4. After registration, APIC offers to DHCP discovery from Leaf and allocates an address from the DHCP pool, which is essentiall the Address Pool for TEP address.

1. First Leaf Switch Discovery





APIC

Switch

LLDP

LLDP

DHCP DISCOVER/REQUEST

DHCP OFFER/ACK

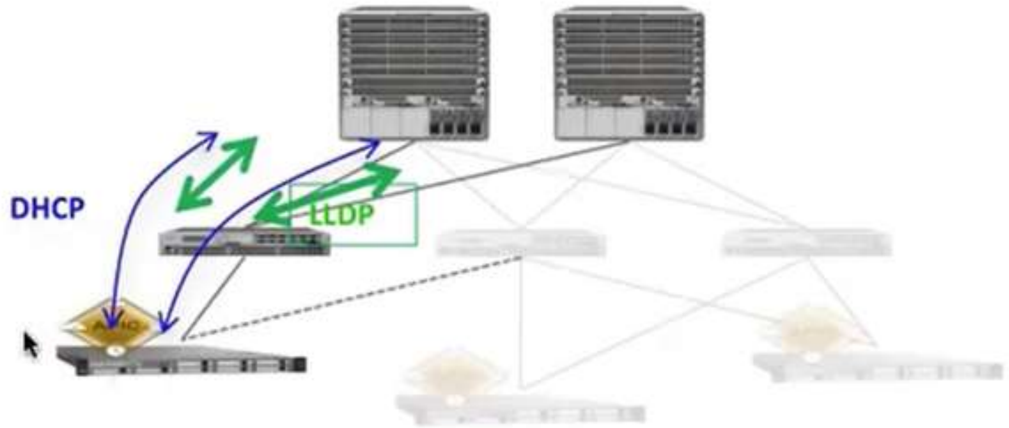
HTTP GET Boot File

HTTP GET 200 OK

Exchange Policies

1. Cisco APIC uses LLDP neighbor discovery to discover a switch.
2. After a successful discovery, the switch sends a request for an IP address via DHCP.
3. To register Leaf to the Fabric, **Manual Register** from APIC is required after auto-discovery where we define **Node Id , POD Id, Role, Rack Name and Node Name**.
4. After registration, APIC offers to DHCP discovery from Leaf and allocates an address from the DHCP pool, which is essentially the Address Pool for TEP address.
5. APIC initiates the encrypted TCP session with the switch to install policies

2. Spine Switches Discovery



1. Spine switches discover the first leaf via LLDP and start sending DHCP discovers.

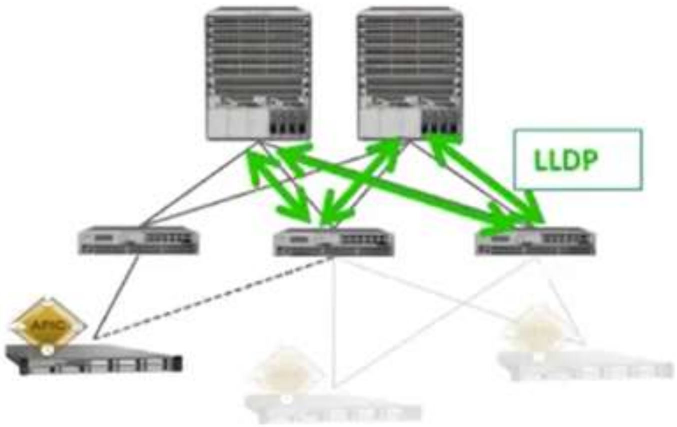
2. Manual Register from APIC is required where we define **Node Id , POD Id, Role, Rack Name and Node Name**.

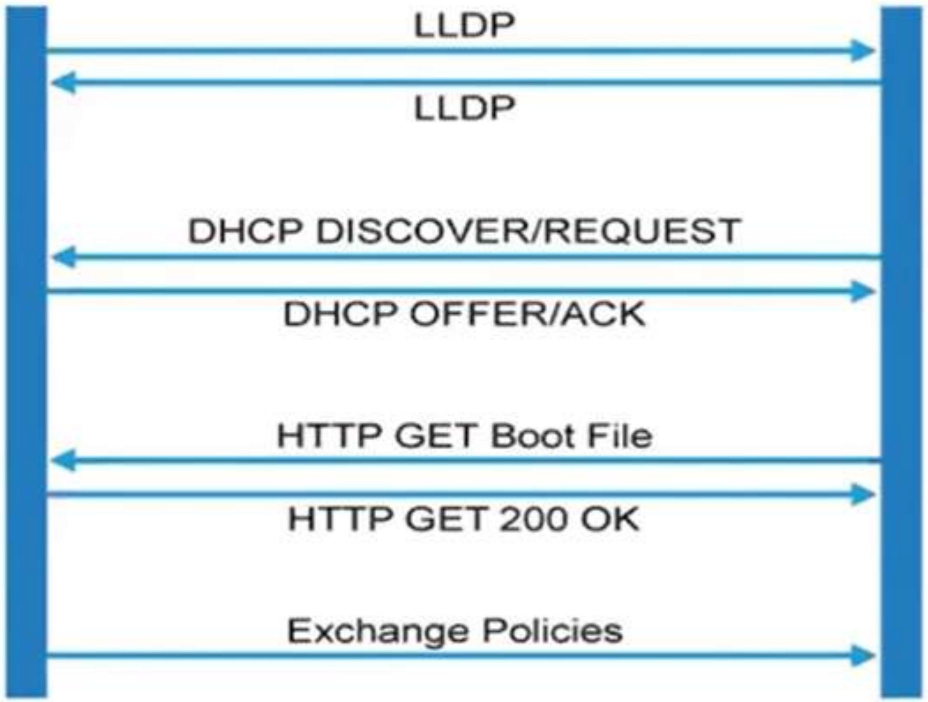
3. After the registration, APIC 1 starts responding to DHCP for the spine switches and assigns TEP IPs to them from TEP Pool.

4. TCP sessions are established between each spine and APIC 1 via TEP IP

3. Discovery other Leaf via Spine

In our example only APIC1 is in active for now
Therefore APIC1 is responding to DHCP




LLDP

LLDP

DHCP DISCOVER/REQUEST

DHCP OFFER/ACK

HTTP GET Boot File

HTTP GET 200 OK

Exchange Policies

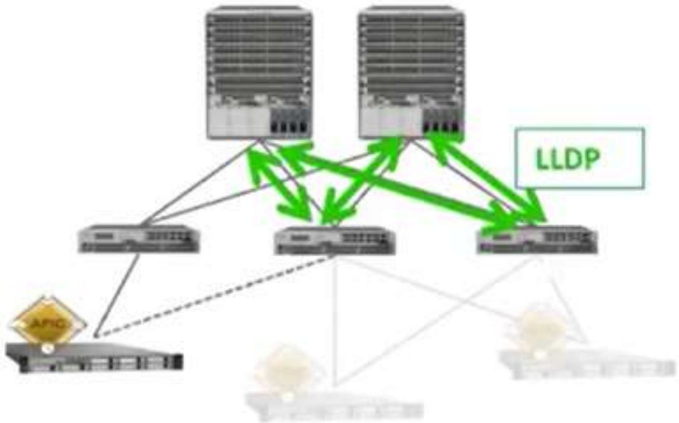
1. The Other leaf switches discover spine switches via LLDP and start sending DHCP discovers.

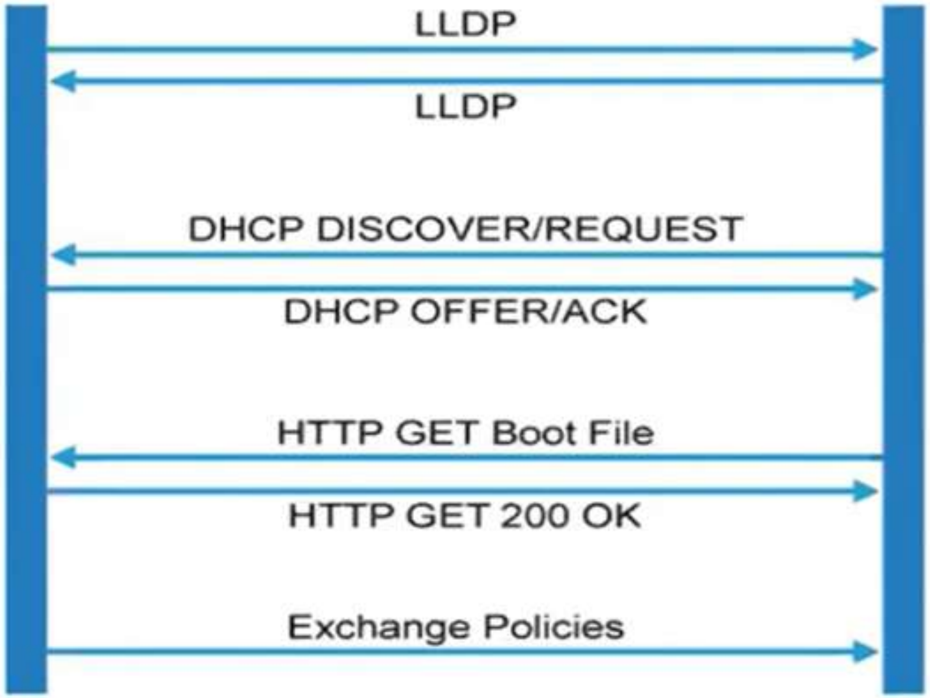
2. Manual Register from APIC is required where we define **Node Id , POD Id, Role, Rack Name and Node Name**.

3. After the registration, **APIC 1** starts responding to DHCP for the spine switches and assigns TEP IPs to them from TEP Pool.

3. Discovery other Leaf via Spine

In our example only APIC1 is in active for now
Therefore APIC1 is responding to DHCP





LLDP

LLDP

DHCP DISCOVER/REQUEST

DHCP OFFER/ACK

HTTP GET Boot File

HTTP GET 200 OK

Exchange Policies

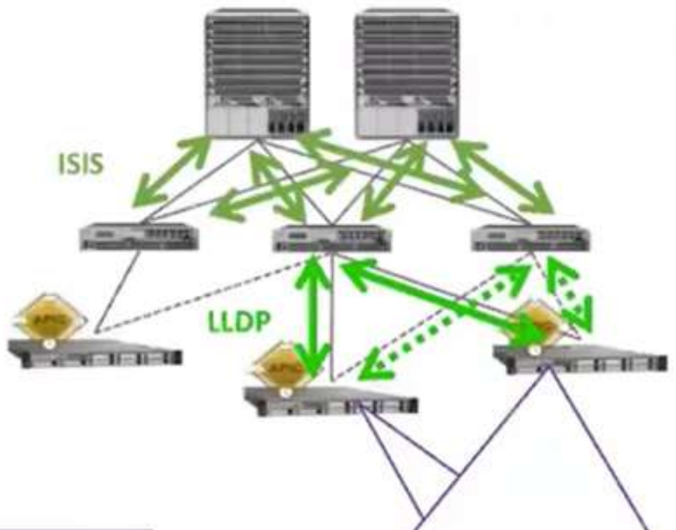
1. The Other leaf switches discover spine switches via LLDP and start sending DHCP discovers.

2. Manual Register from APIC is required where we define **Node Id , POD Id, Role, Rack Name and Node Name**.

3. After the registration, **APIC 1** starts responding to DHCP for the spine switches and assigns TEP IPs to them from TEP Pool.

Fabric Discovery

5. APIC2,3 join Fabric as well



The important information stored in AV (Appliance Vector) is the followings:

Information of the entire fabric

- Fabric Name
- TEP Address Range APIC Cluster Size

Information of each APIC

- APIC ID
- APIC TEP IP
- APIC Universally Unique ID (UUID) or chassis ID

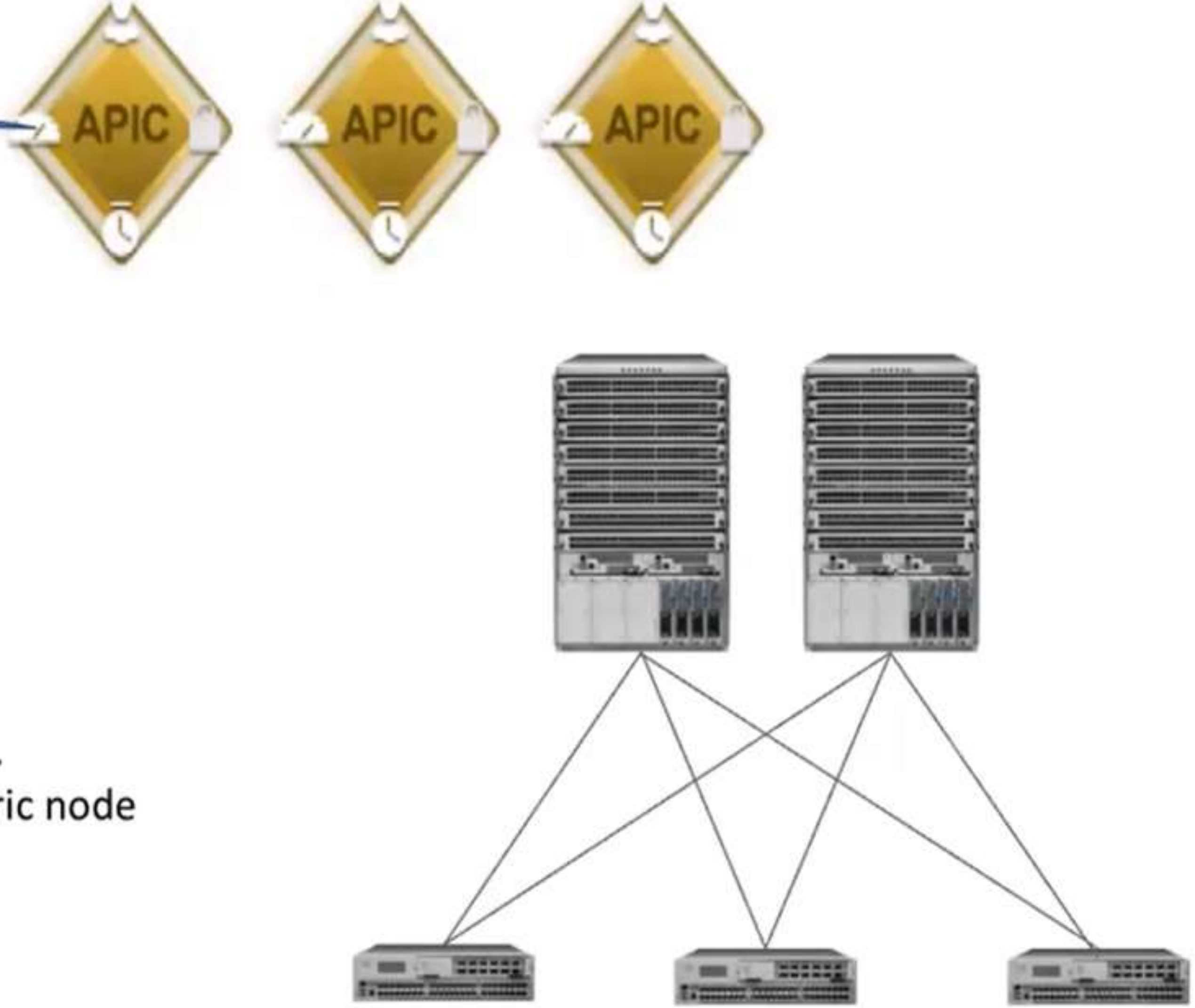
APIC2,3 Cluster Join

- No Manual Registration. It's automatic.
- Password is synced from APIC1 and user is able to log in APIC2,3 from this point of time
- At the same time, Database distribution/replication are automatically done.
- Possible Reasons for Join failure
  - Wrong Fabric Domain Name
  - UUID & APIC number pair stored on Leaf is different from the one on APIC

Fabric Discovery

TEP Address Pool : X.X.X.X/X
Infrastructure VLAN : X
Other Parameters like Fabric Id , Name...

LLDP
DHCP



Cisco APIC automatically discovers all spine and leaf switches. However, we have to manually register those switches as fabric node members for them to participate in the Cisco ACI fabric.

ACI OBJECTS

The Tenant Object

A tenant is a logical container for application policies that enable an administrator to exercise domain-based access control. A tenant represents a unit of isolation from a policy perspective. Tenants can represent a customer in a service provider setting, an organization or domain in an enterprise setting, or just a convenient grouping of policies.

Tenant

Class:	fvTenant
DN Prefix:	tn-
Parent Object:	uni (Universe)

Default Tenants

infra

Infrastructure tenant: Networking for Inter-Pod Network (IPN), Inter-Site Network (ISN), and Remote Leaf.

Recommendation: Don’t use for other networks

mgmt

Management tenant: Out-of-band and in-band management for ACI controllers, leaves, and spines.

Recommendation: Don’t use for other networks

common

Common tenant: Common services to be consumed across multiple tenants.

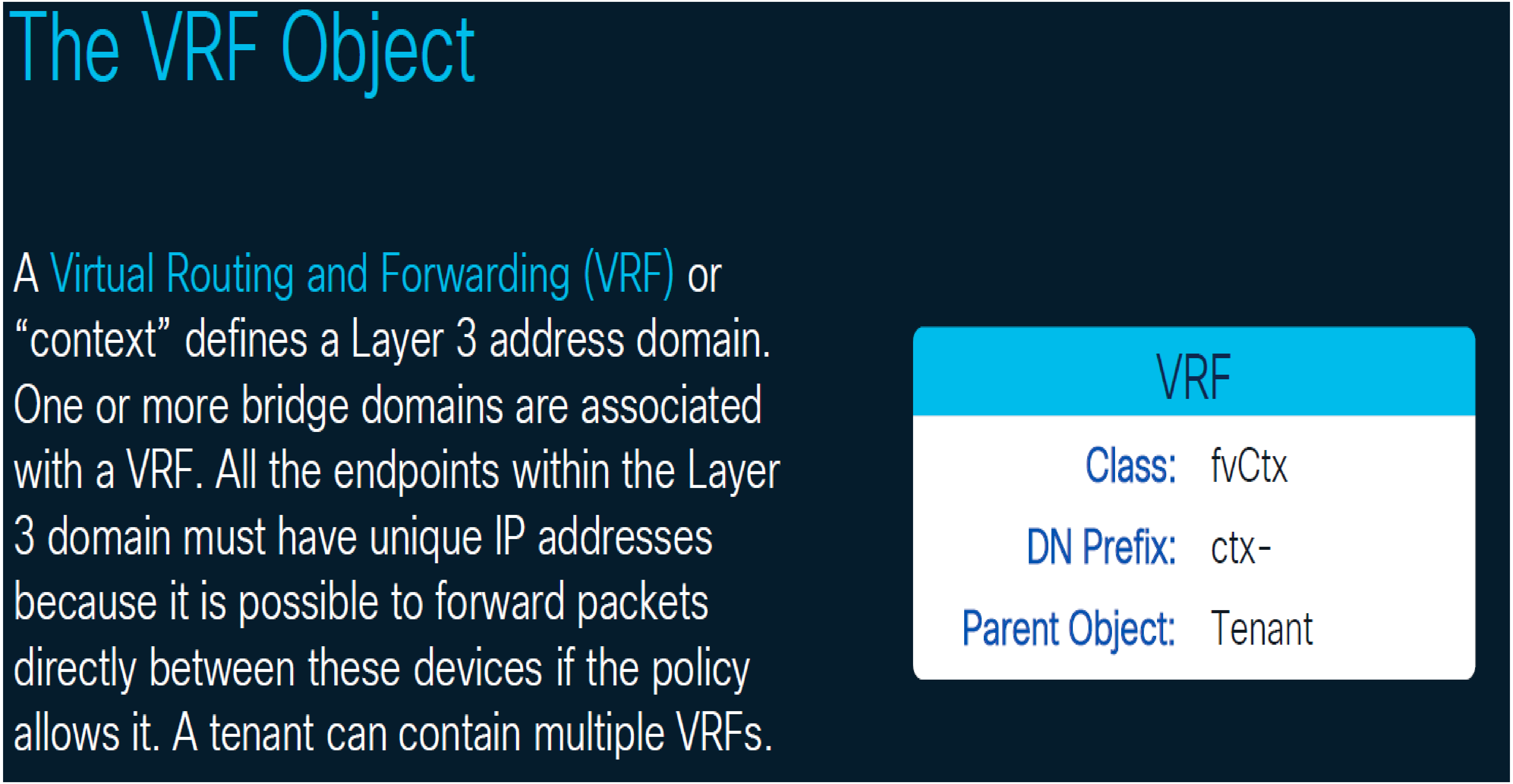
Recommendation: Use for shared services

Default Tenants – Common

Common Tenant – Dedicated tenant for common services and shared objects.

- For shared services hosted in Common and consumed in other tenants.
- Route-leaking is simplified, but still required.
- Not recommended to split the object model between Tenants. (e.g. using a VRF or an L3Out out of Common from another tenant.)
  - Exception: It is a normal practice to use Filters from the Common tenant across the fabric

Example: Active Directory services hosted in Common and consumed in “Prod” and “Dev” tenants.



The VRF Object

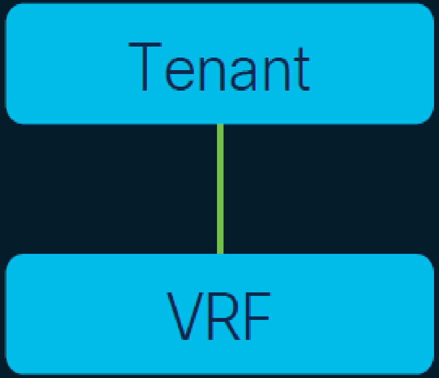
A Virtual Routing and Forwarding (VRF) or "context" defines a Layer 3 address domain. One or more bridge domains are associated with a VRF. All the endpoints within the Layer 3 domain must have unique IP addresses because it is possible to forward packets directly between these devices if the policy allows it. A tenant can contain multiple VRFs.

VRF	
Class:	fvCtx
DN Prefix:	ctx-
Parent Object:	Tenant

Tenant Object Model - Tenant and VRF

Preferred Design

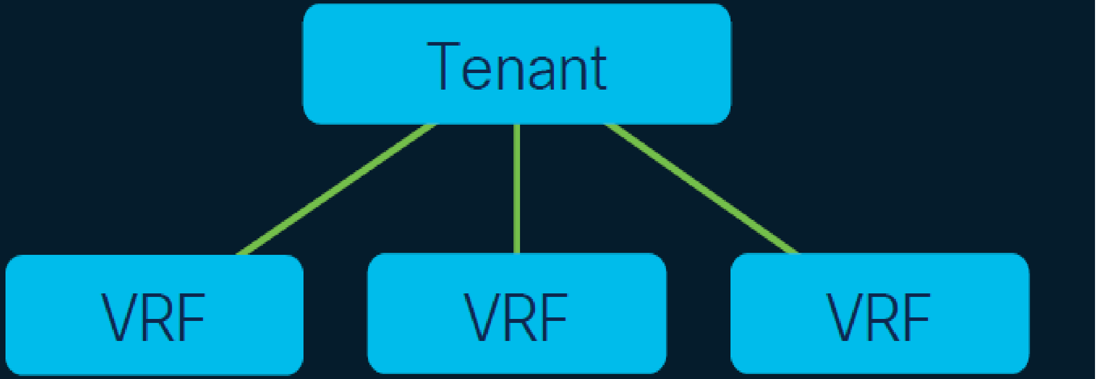
Single VRF per Tenant



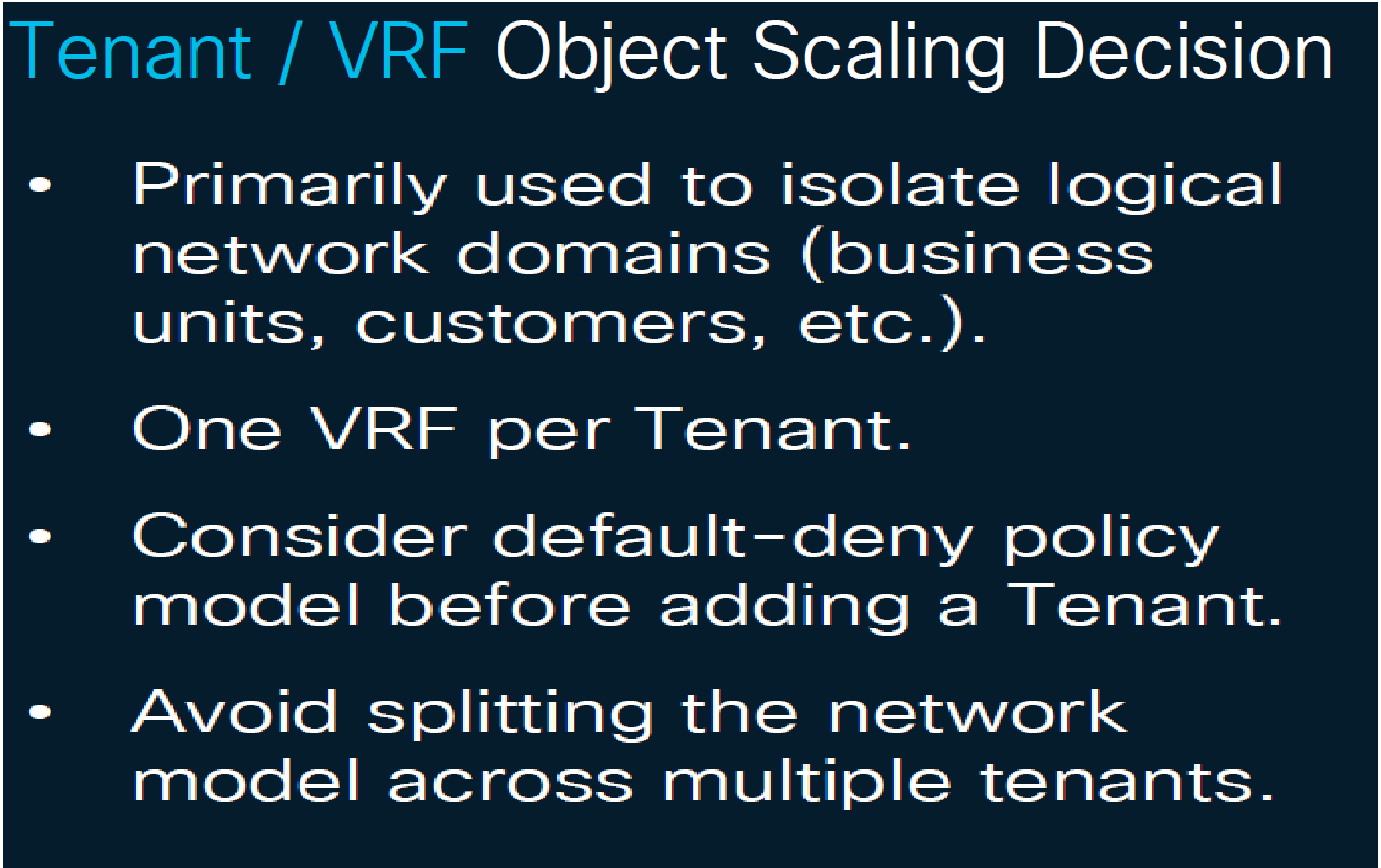
- Object relationships easier to track
- Simplified L3Out
- No route-leaking

Not Recommended

Multiple VRFs per Tenant



- Confuses object relationships
- Complicates L3Outs
- May encourage route–leaking



Tenant / VRF Object Scaling Decision

- Primarily used to isolate logical network domains (business units, customers, etc.).
- One VRF per Tenant.
- Consider default-deny policy model before adding a Tenant.
- Avoid splitting the network model across multiple tenants.

The Bridge Domain Object

A bridge domain defines the unique Layer 2 MAC address space and a Layer 2 flood domain. With connections to Subnets and L3Outs, the BD provides a critical connection point between Endpoints and all things at Layer 3.

Uses:

- Define L2 flooding domain
- Connect to ACI L3 Object (Subnet, L3Out, etc.)

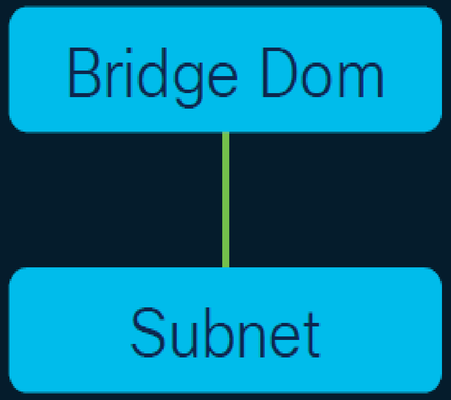
Bridge Domain

Class:	fvBD
DN Prefix:	BD-
Parent Object:	Tenant

Tenant Object Model - BD and Subnet

Preferred Design

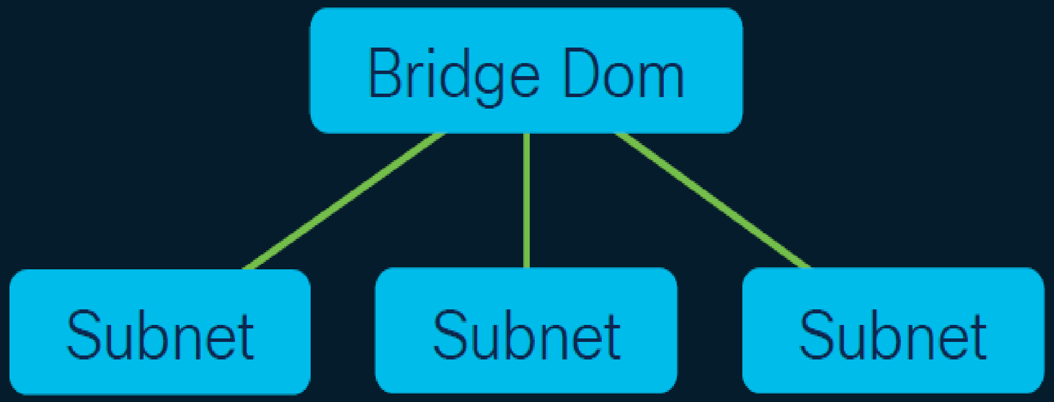
Single Subnet per Bridge Domain



- Endpoint addressing easier to track
- Simplified Routing

Not Recommended

Multiple Subnets per Bridge Domain



- Confuses Endpoint Addressing
- Complicates Routing

The Subnet Object

The subnet object brings the layer 3 address space to a bridge domain where it can ultimately be consumed by endpoints in an endpoint group (EPG). The address defined in a subnet will be programmed into leaves as an anycast gateway for the associated endpoints.

Uses:

- Subnet and Anycast Gateway for endpoints

Subnet	
Class:	fvSubnet
DN Prefix:	subnet-
Parent Object:	Bridge Domain

The Application Profile Object

The application profile contains as many (or as few) EPGs as necessary that are logically related to providing the capabilities of an application.

Uses:

- Organization of EPGs
- Contract scope boundary (not recommended)

Application Profile

Class:	fvAp
DN Prefix:	ap-
Parent Object:	Tenant

Application Profile General

Generally, just an EPG/ESG container

The Endpoint Group Object

An EPG is a managed object that contains a collection of endpoints. Endpoints are devices that are connected to the network directly or indirectly. EPGs are fully decoupled from the physical and logical topology.

Uses:

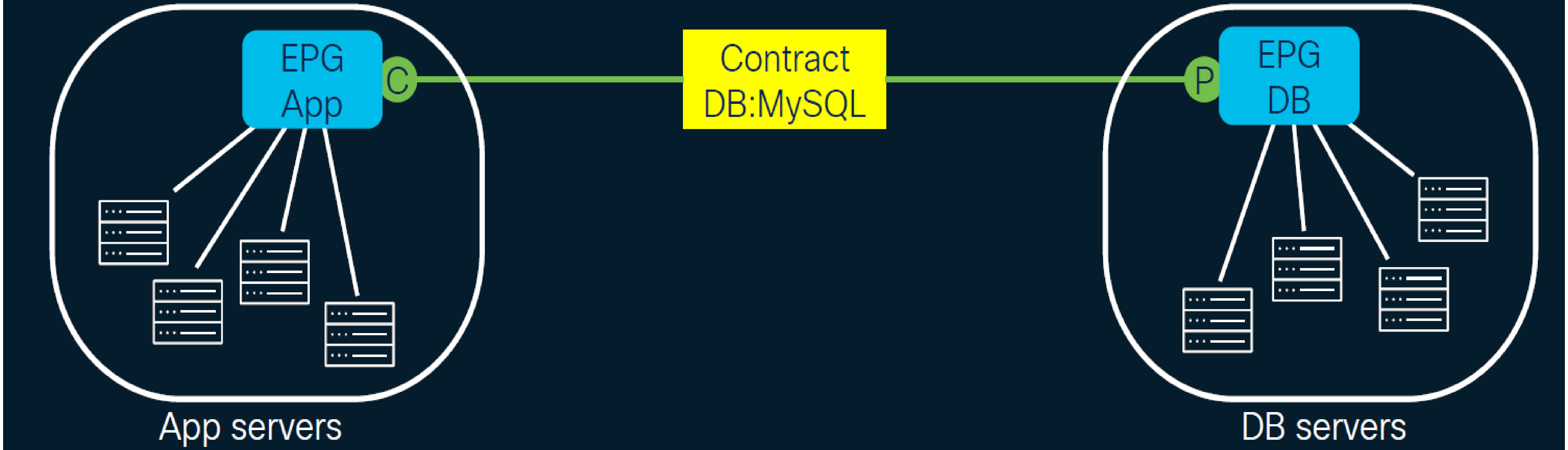
- Segmentation
- Association to BD/Subnet
- Define endpoint connectivity

Endpoint Group

Class:	fvAEpg
DN Prefix:	epg-
Parent Object:	App Profile

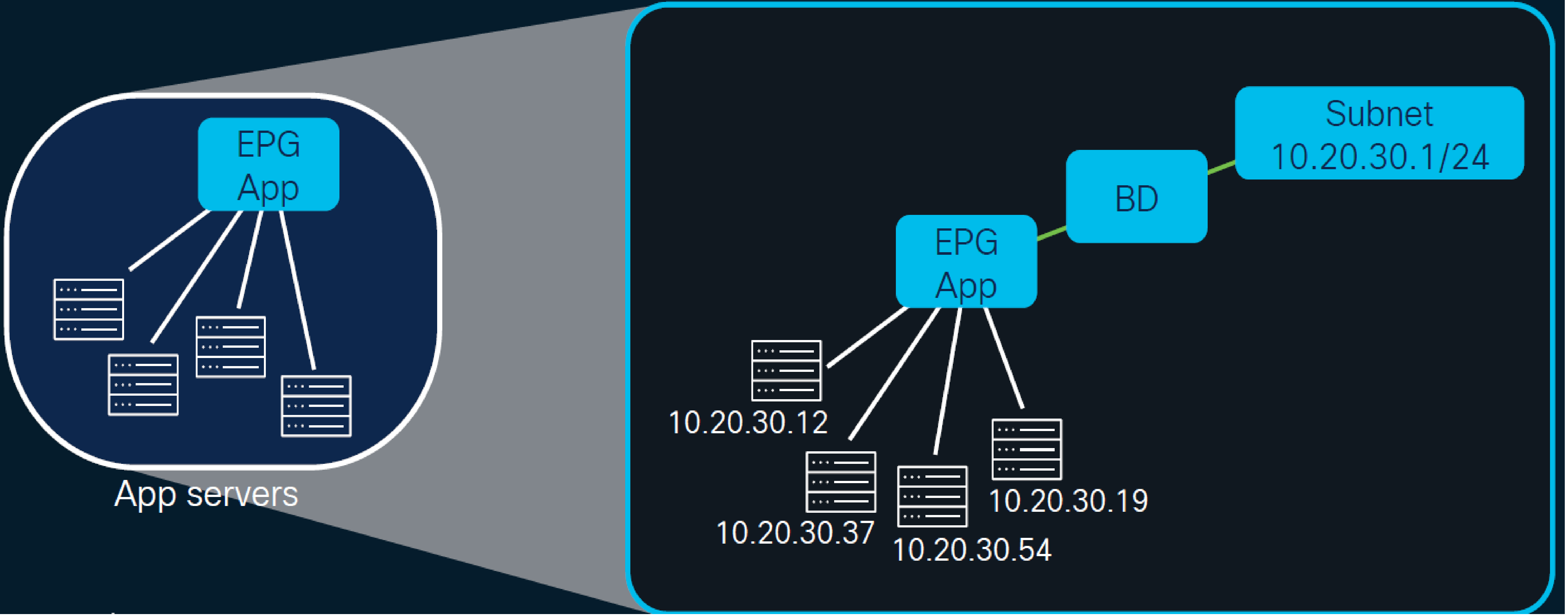
Endpoint Group

Primary Role: Group Endpoints with like policy requirements



Endpoint Group

Secondary Role: Connect Endpoints to a BD and Subnet.



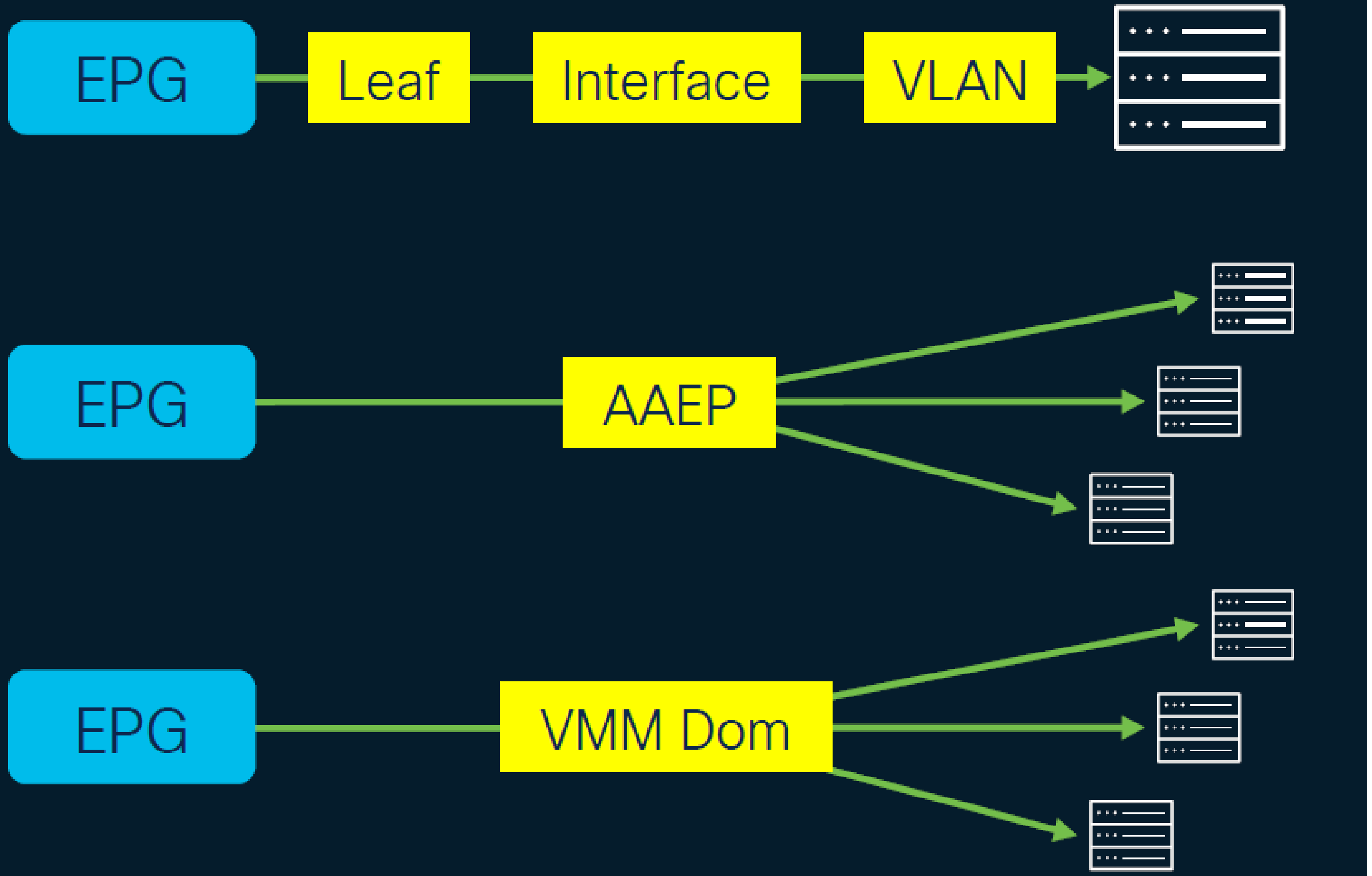
Endpoint Group

Tertiary Role: Connect endpoints to the fabric.

Static Port Binding

AAEP to EPG mapping

VMM Integration



EPG Object Scaling Decision

- Primarily decision should focus on segmentation.
- Consider breaking up a BD/Subnet into multiple EPGs as needed.
- Primary limiting factor for EPGs is VLAN limit per leaf.
  (Each EPG uses one VLAN.)

The Endpoint Security Group Object

An ESG is a managed object that contains a collection of endpoints. Unlike EPGs, ESGs have no direct involvement in subnet mapping or endpoint connectivity.

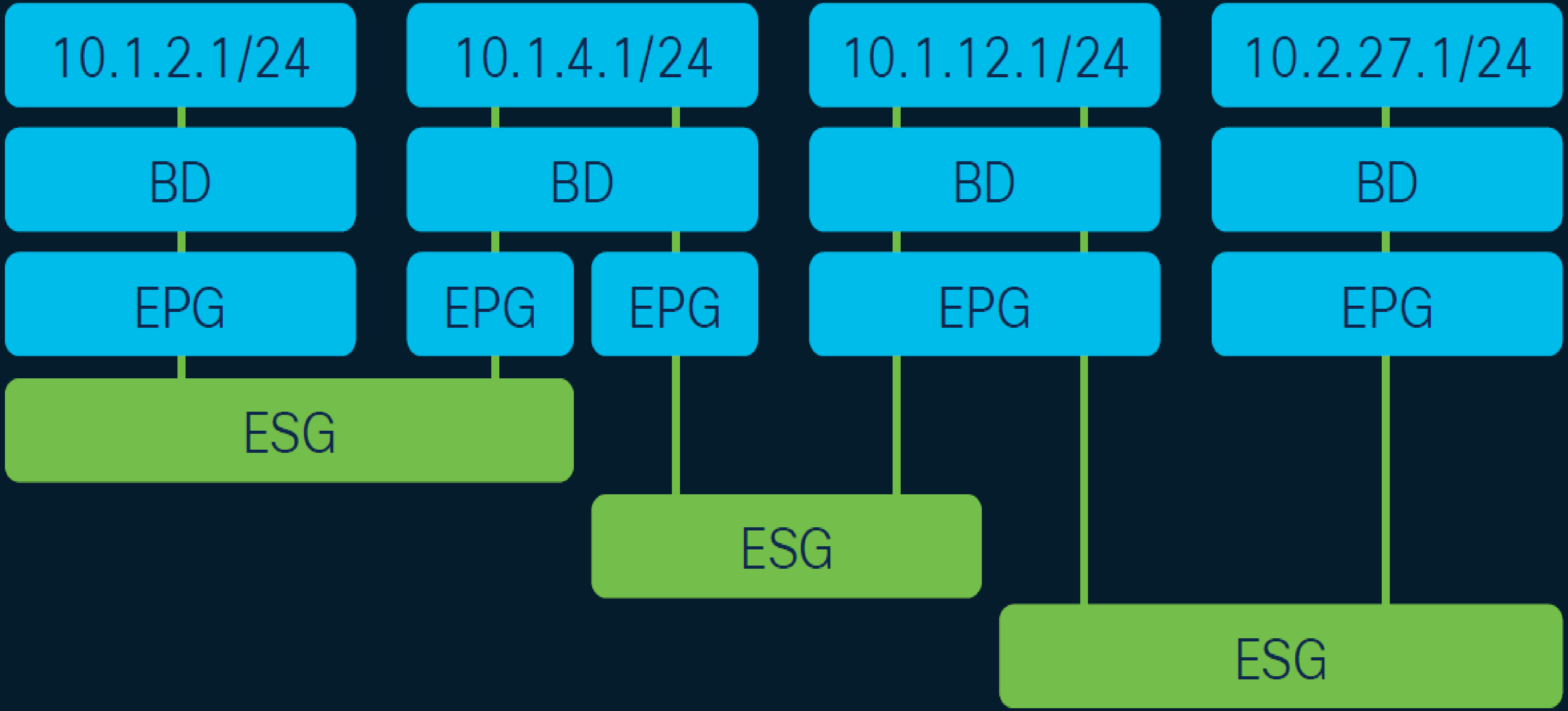
Endpoint Selectors:

- EPG membership
- Tag: MAC, IP, VM name, VM Tag
- IP/subnet

Endpoint Group

Class:	fvESg
DN Prefix:	esg-
Parent Object:	App Profile

ESG – Endpoint Security Group



The L3Out Object

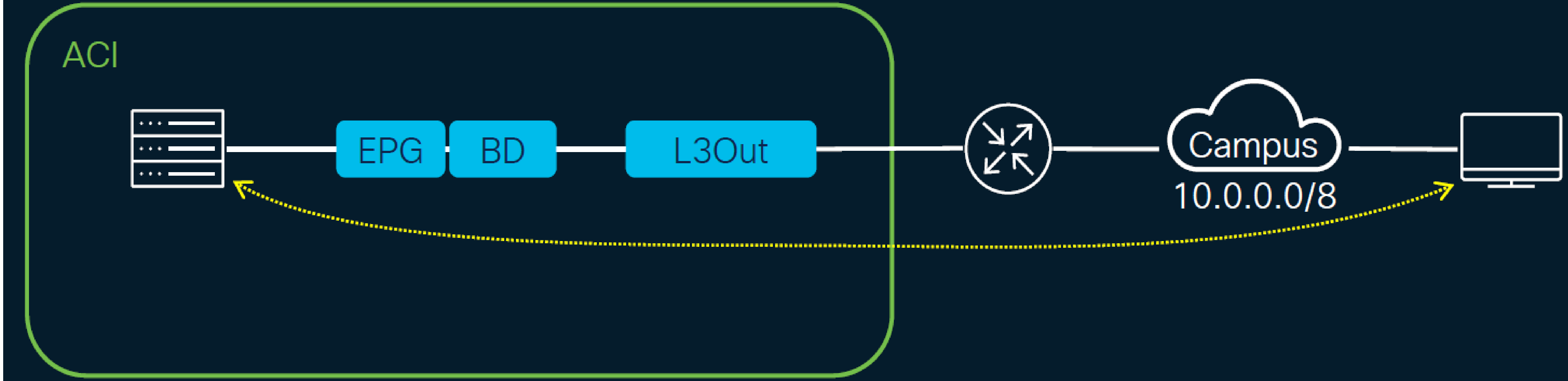
A Layer 3 connection between border leaves in the fabric and a set out external routing devices (router, firewall, L3 Switch, etc.). This connection provides a routed path out of the fabric; designed to reach a specific set of networks (defined by External EPG).

Uses:

- Define a routed path out of a Tenant
- A configuration point for routing protocols or static routes.

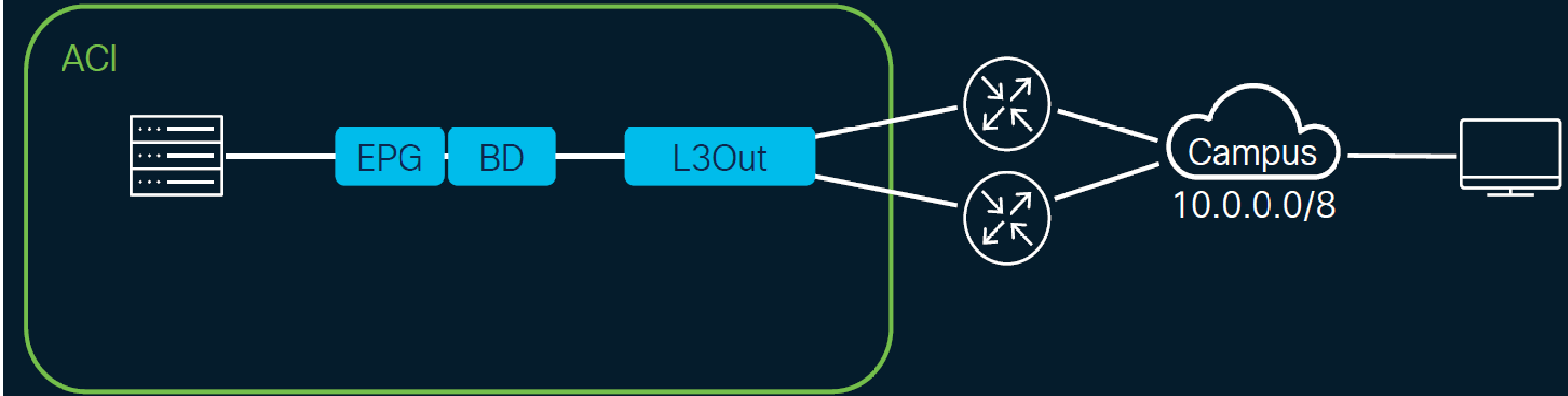
L3Out	
Class:	l3extOut
DN Prefix:	out-
Parent Object:	Tenant

L3Out Object



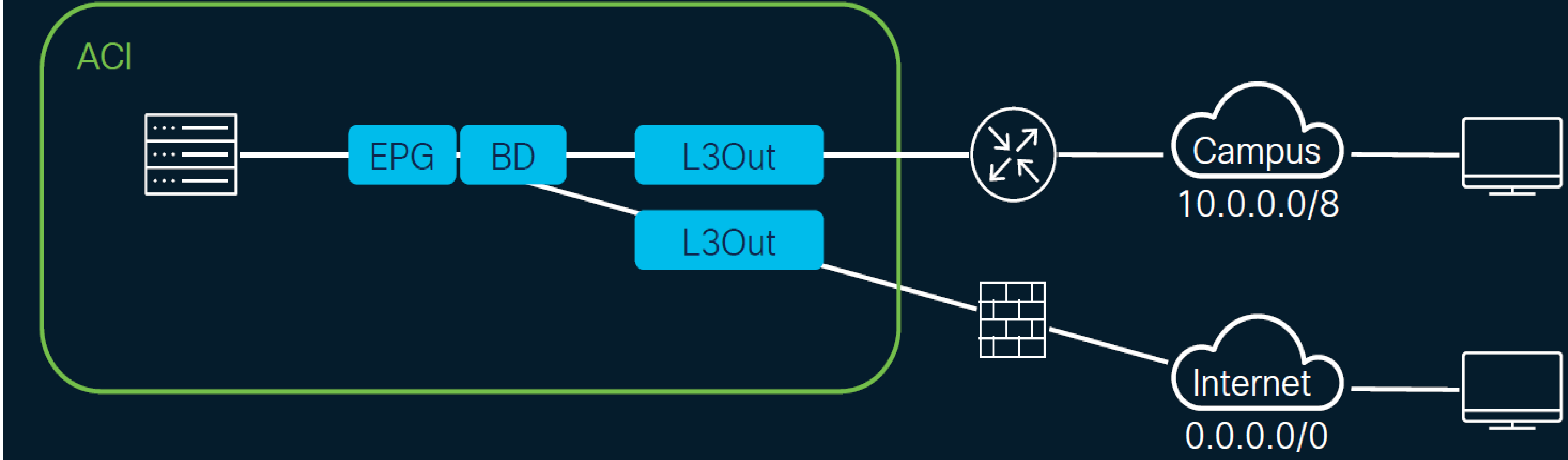
The purpose of the L3Out is to provide a routed path between EPGs and a defined set of external networks.

L3Out Object



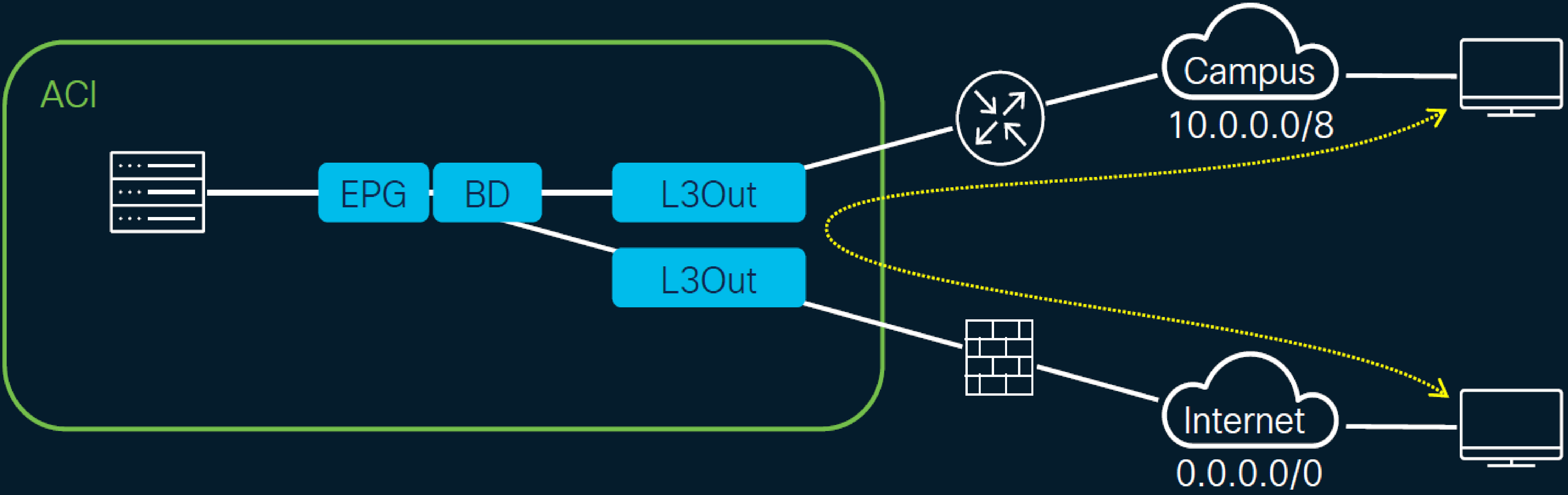
A single L3Out can and should be used even if there are multiple paths to the external networks.

L3Out Object



A different L3Out should be used when there is a different path to a different set of external networks.

Transit Routing



Disabled by default, Transit Routing allows traffic to pass through ACI between external networks connected through different L3Outs.

The Contract Object

A contract is a policy construct used to define communication between EPGs. Without a contract between EPGs, no unicast communication is possible unless the VRF is configured in unenforced mode, or those EPGs are in the preferred group.

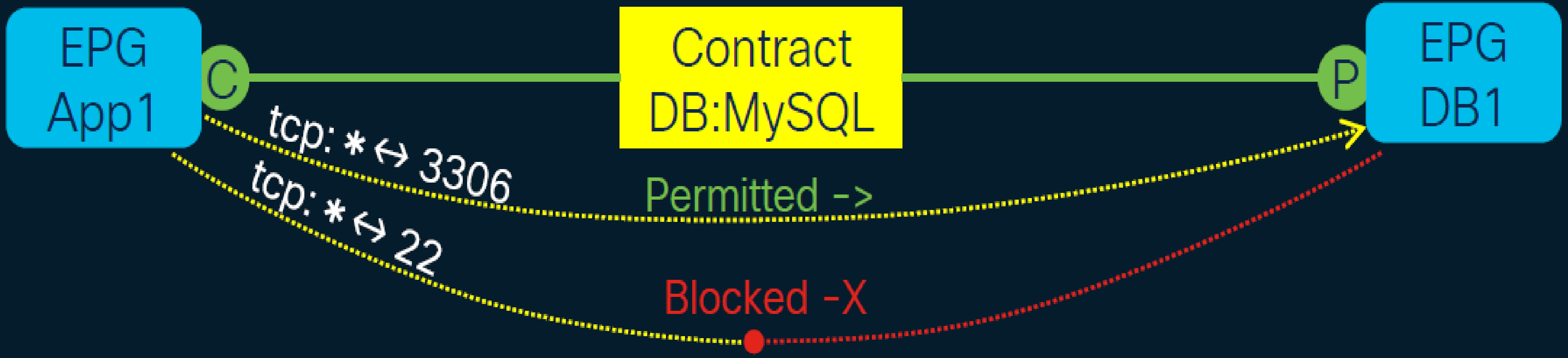
Uses:

- Define allowed communication between EPGs
- Route-leaking

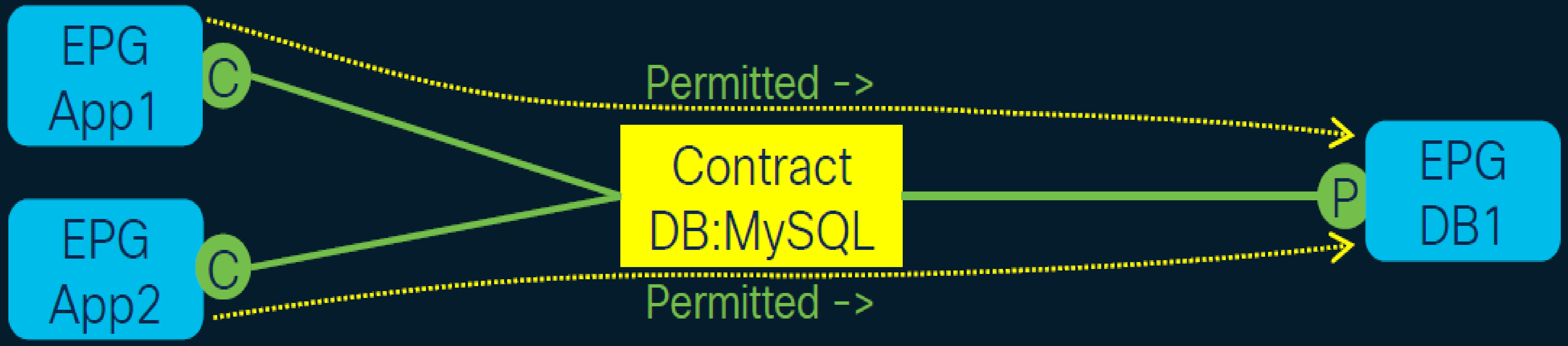
Contract

Class:	fvBrCP
DN Prefix:	brc-
Parent Object:	Tenant

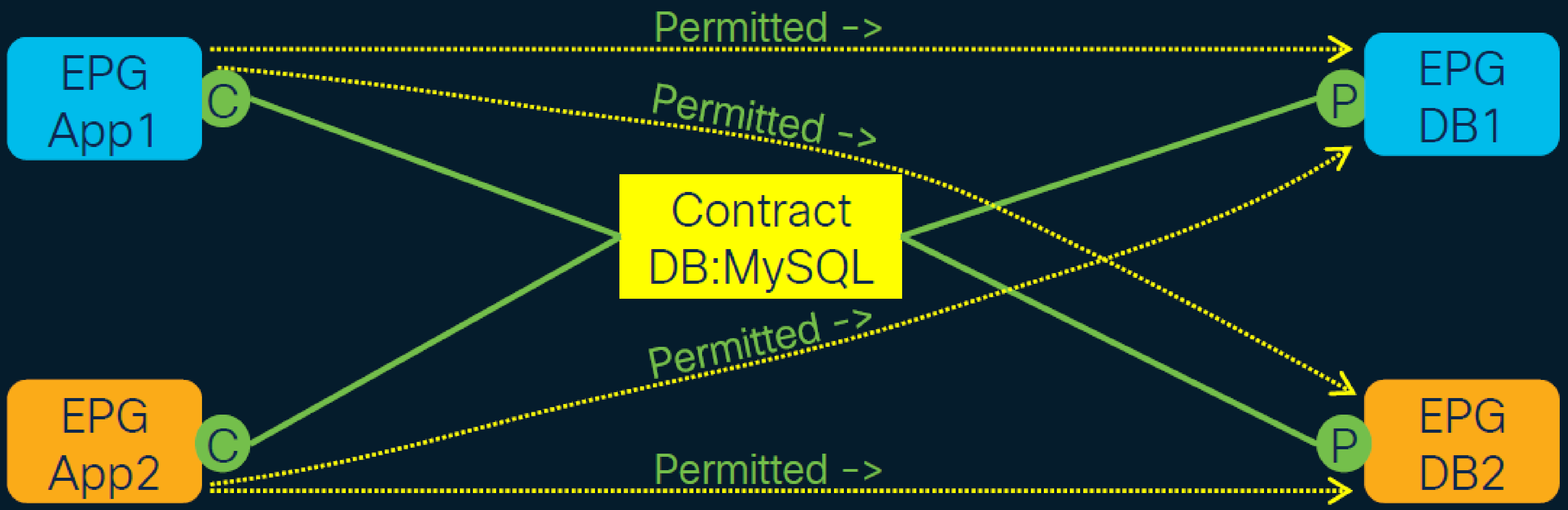
How Contracts Work



Reuse of Contracts

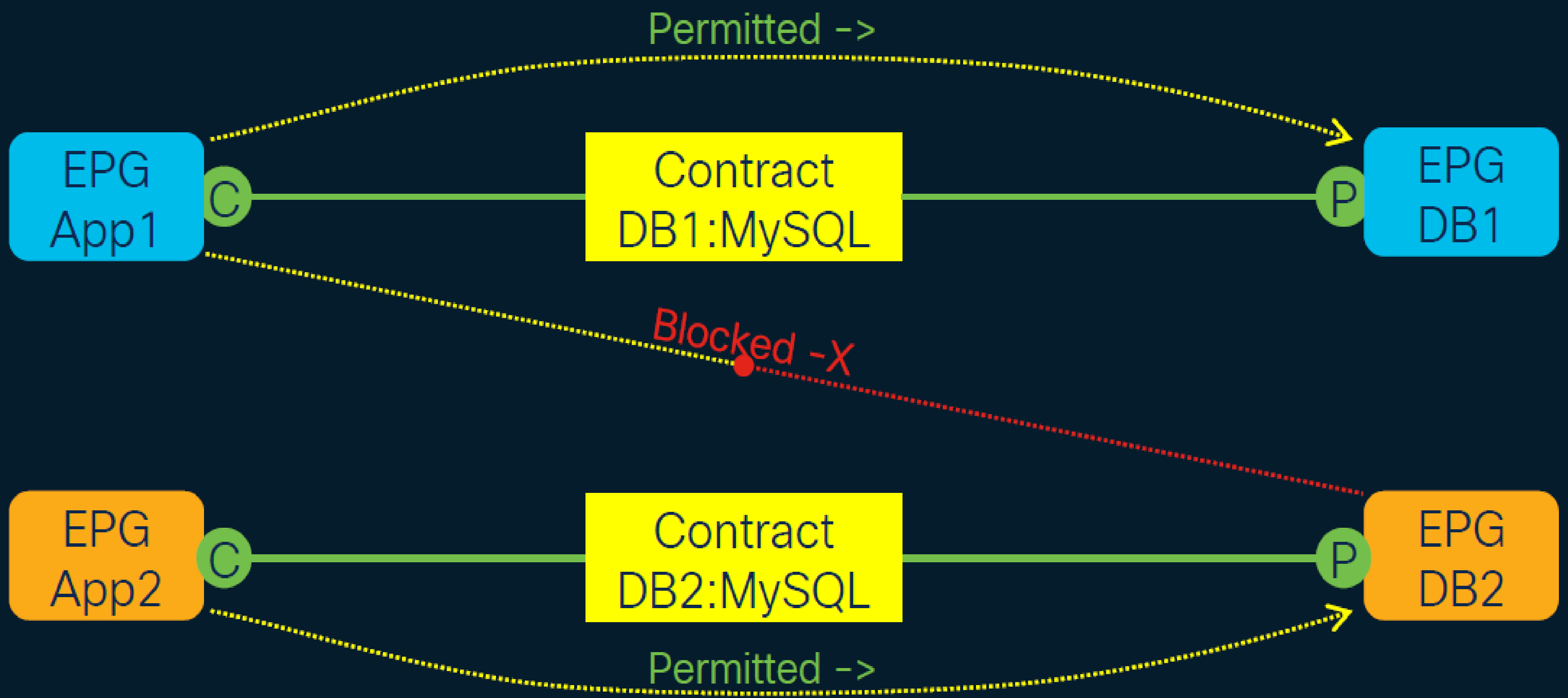


Reuse of Contracts

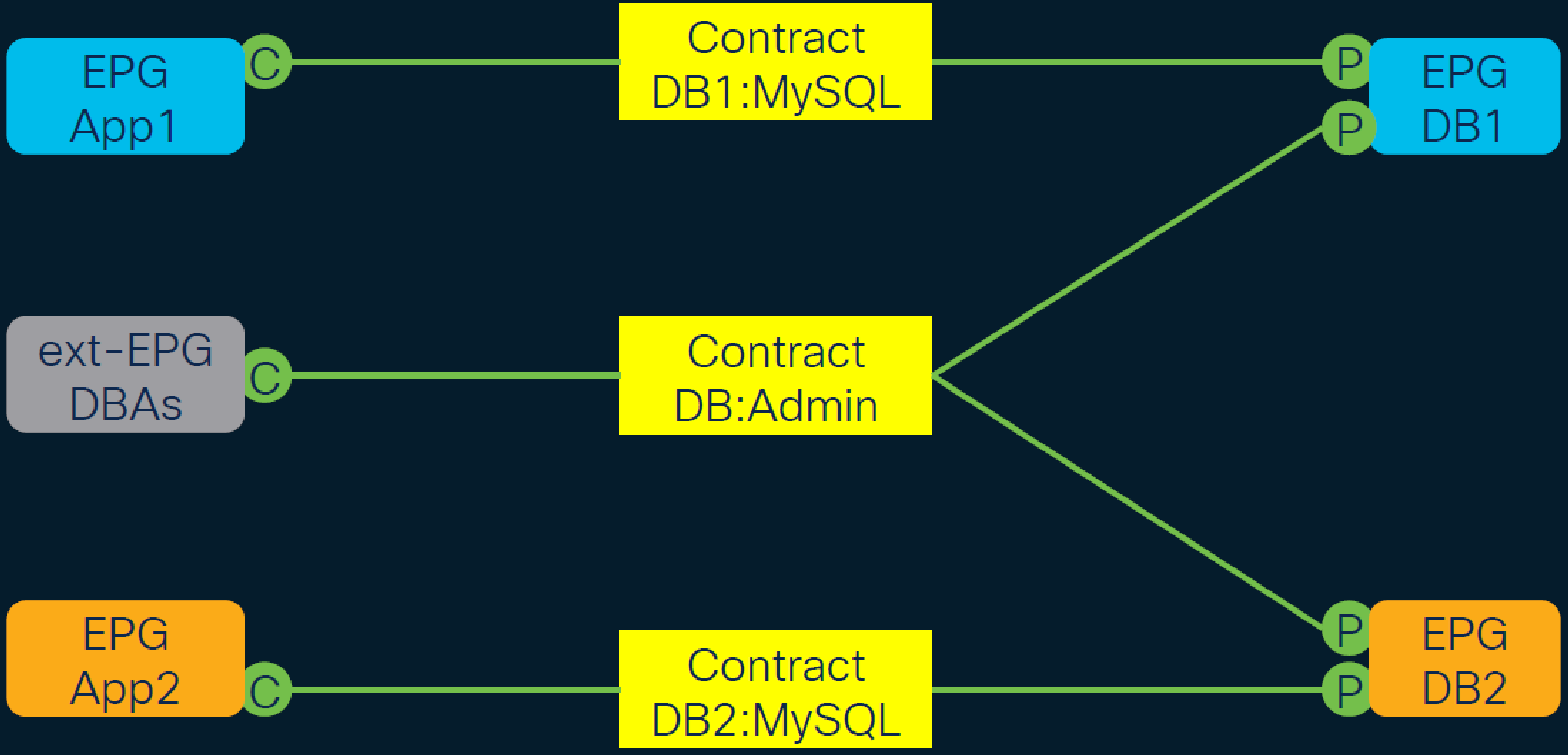


Reuse of contracts may permit traffic that was not intended!

Reuse of Contracts



Contract with multiple providers



The Subject Object

A Subject is a contract attachment point for filters and other contract related policies. This attachment point allows for different policies to be associated with specific filter sets within a single contract.

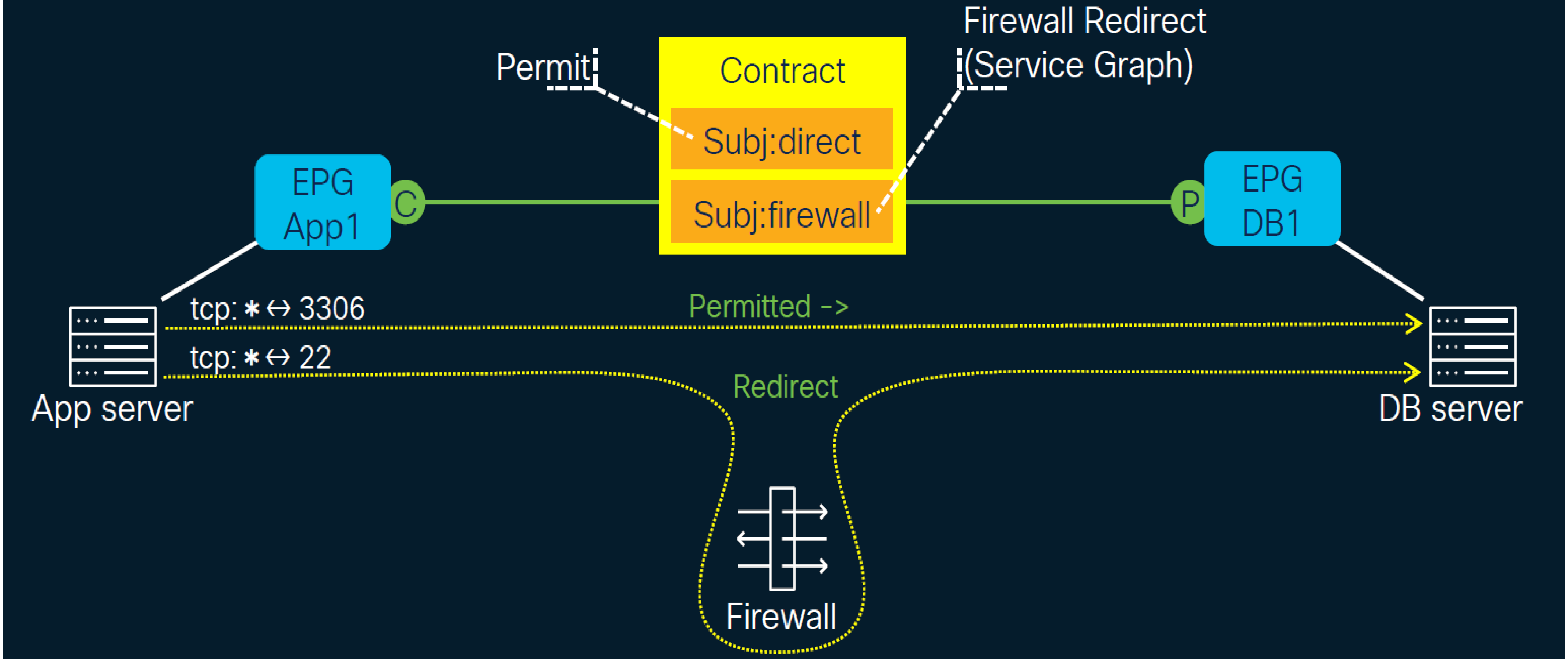
Uses:

- Control how a filter is applied to a contract
- Associate contract policy with filter sets
  - e.g. L4-L7 Service Graph or QoS priority

Subject

Class:	vzSubj
DN Prefix:	subj-
Parent Object:	Contract

Subject


Firewall Redirect
(Service Graph)
Permit
Contract
Subj:direct
Subj:firewall
EPG
App1
C
P
EPG
DB1
tcp: ∗ ↔ 3306
Permitted ->
tcp: ∗ ↔ 22
Redirect
App server
DB server
Firewall

The Filter Object

A Filter is a list of matching rules that define communications to associate with a contract. These rules include things like ethertype, protocol, and source and destination port(s).

Uses:

- Classify traffic for policy enforcement.

Filter

Class:	vzFilter
DN Prefix:	flt-
Parent Object:	Tenant

Filters

Filter: MySQL
tcp: ∗ ↔ 3306

Filter: FTP
tcp: ∗ ↔ 20
tcp: ∗ ↔ 21

Filter: DNS
udp: ∗ ↔ 53
tcp: ∗ ↔ 53

Filter: HTTP
tcp: ∗ ↔ 80

Filter: HTTPalt
tcp: ∗ ↔ 8080

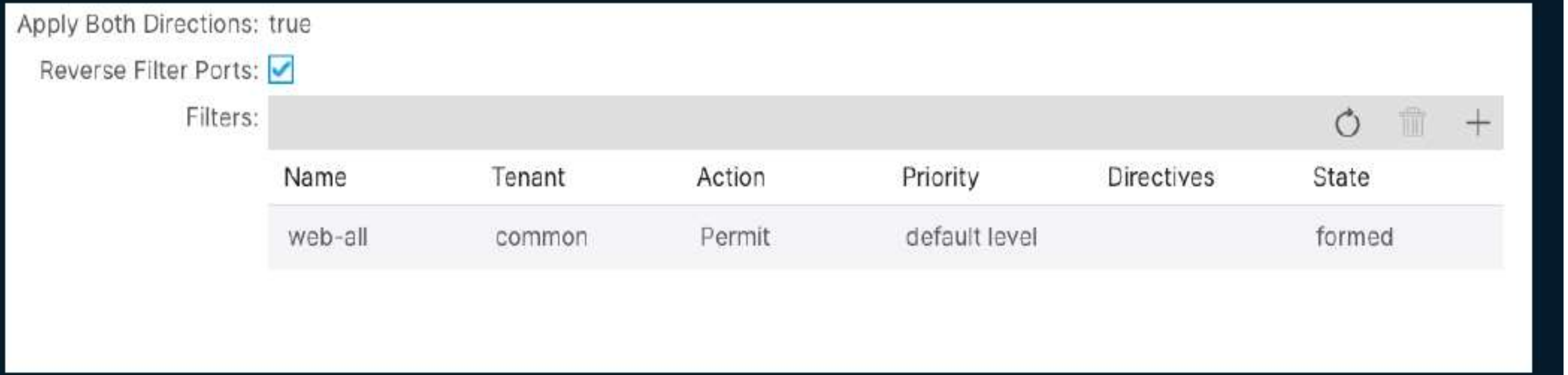
Filter: PKI
tcp: ∗ ↔ 80
tcp: ∗ ↔ 389
tcp: ∗ ↔ 636
tcp: ∗ ↔ 9389

Filter: HTTPS
tcp: ∗ ↔ 443

Filter: HTTPSalt
tcp: ∗ ↔ 8443

Filters

Filter: web-all
tcp: ∗ ↔ 80
tcp: ∗ ↔ 443
tcp: ∗ ↔ 8080
tcp: ∗ ↔ 8443



Apply Both Directions: true

Reverse Filter Ports: ☑

Filters:

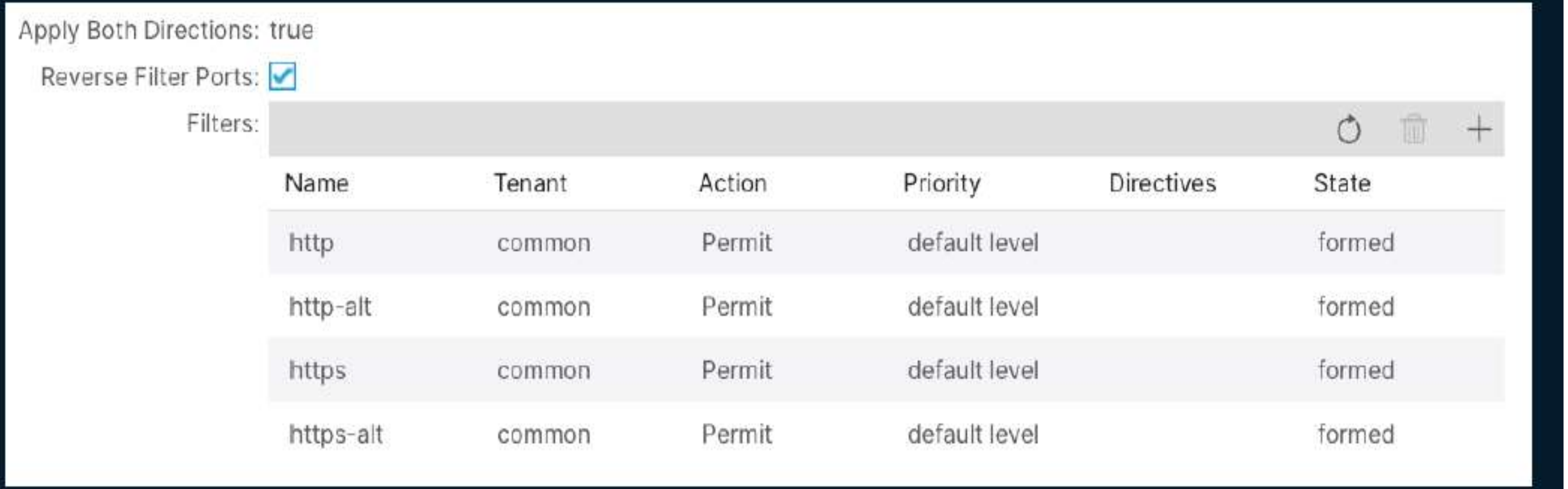
Name	Tenant	Action	Priority	Directives	State
web-all	common	Permit	default level		formed

Filter: http
tcp: ∗ ↔ 80

Filter: https
tcp: ∗ ↔ 443

Filter: https-alt
tcp: ∗ ↔ 8443

Filter: http-alt
tcp: ∗ ↔ 8080



Apply Both Directions: true

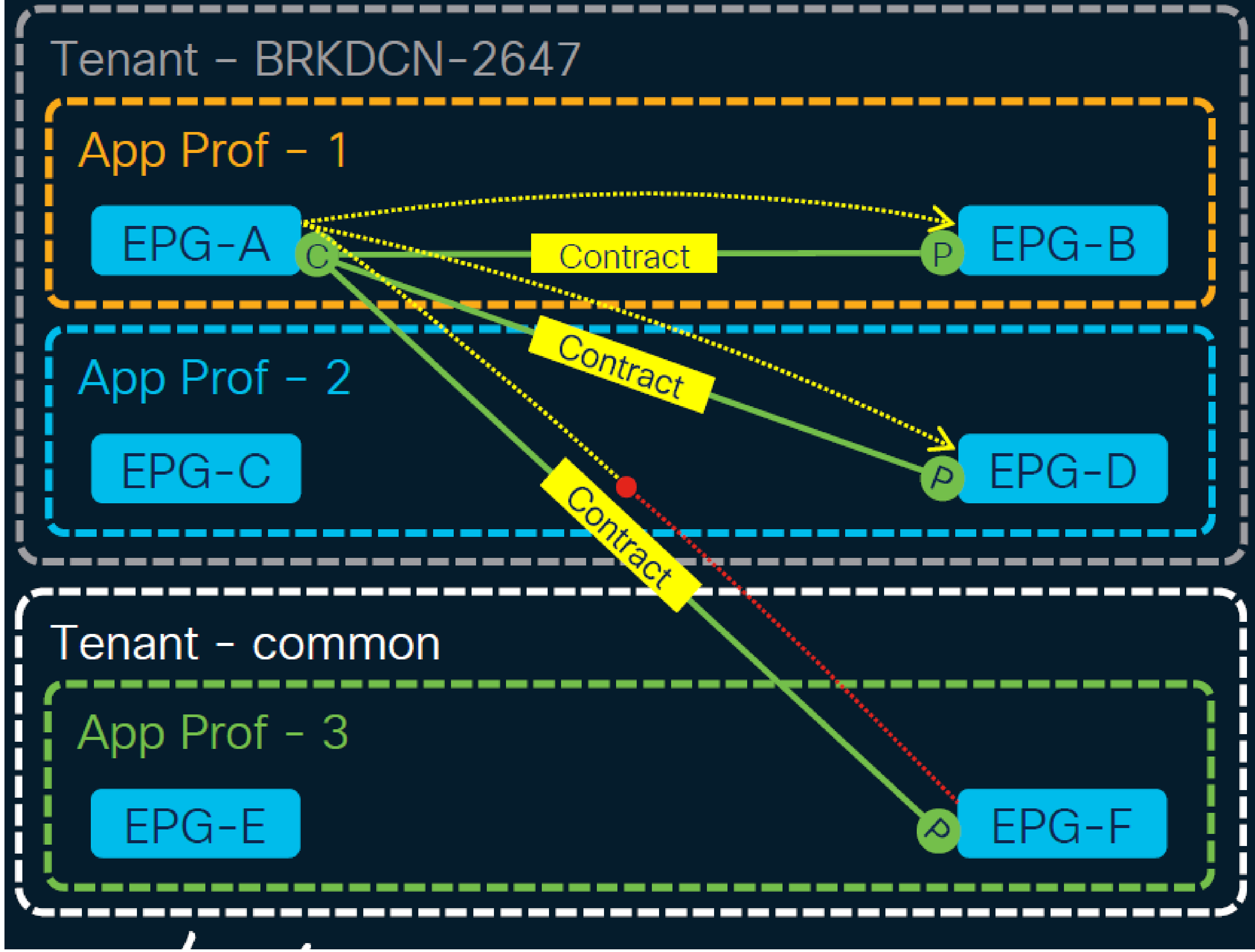
Reverse Filter Ports: ☑

Filters:

Name	Tenant	Action	Priority	Directives	State
http	common	Permit	default level		formed
http-alt	common	Permit	default level		formed
https	common	Permit	default level		formed
https-alt	common	Permit	default level		formed

Contract Scope

VRF/Tenant Scope*

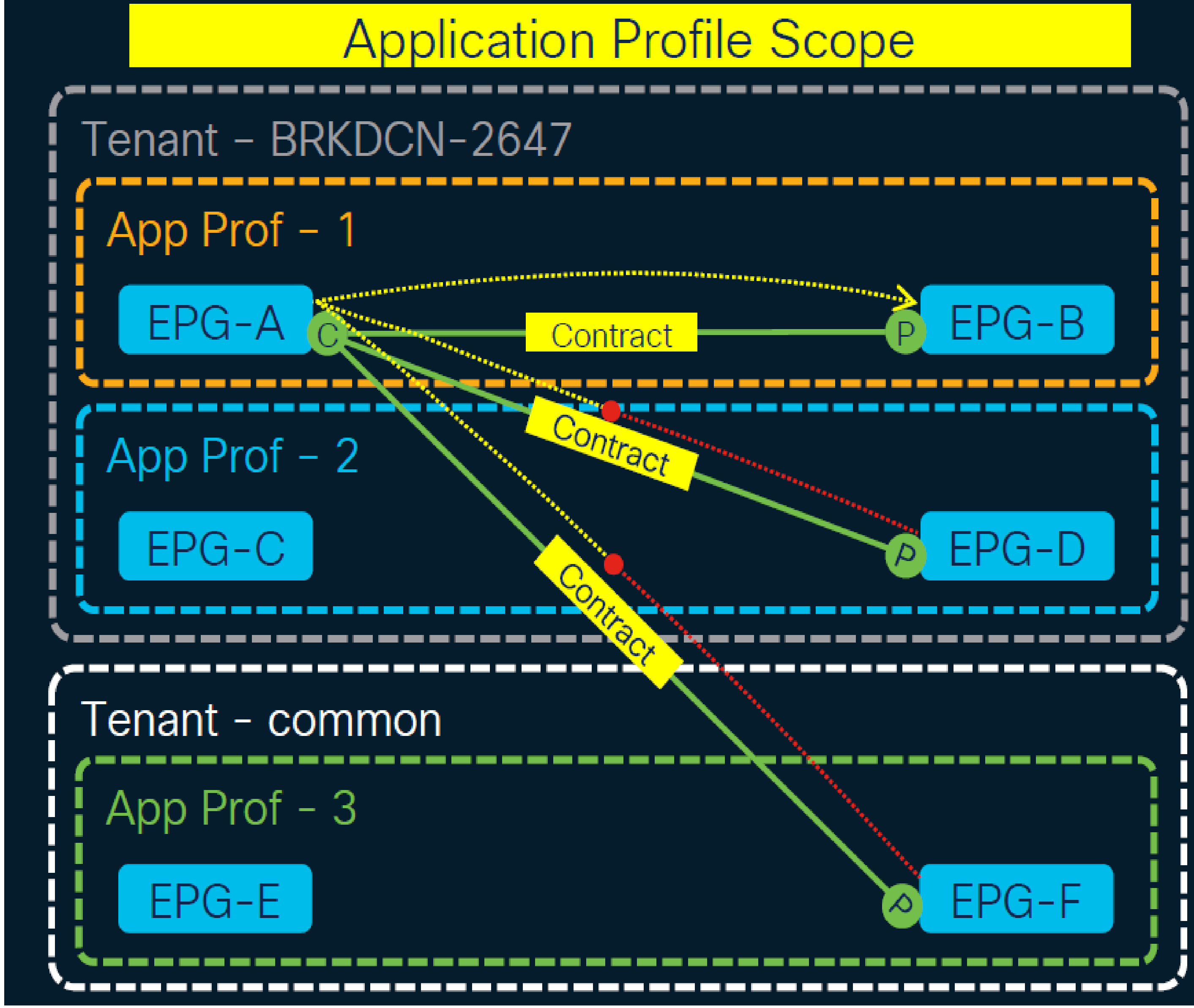


Default Contract Scope – VRF*

Contracts only permit EPGs to communicate within the same the VRF.

* 1:1 Tenant:VRF makes VRF scope functionally the same as Tenant scope.

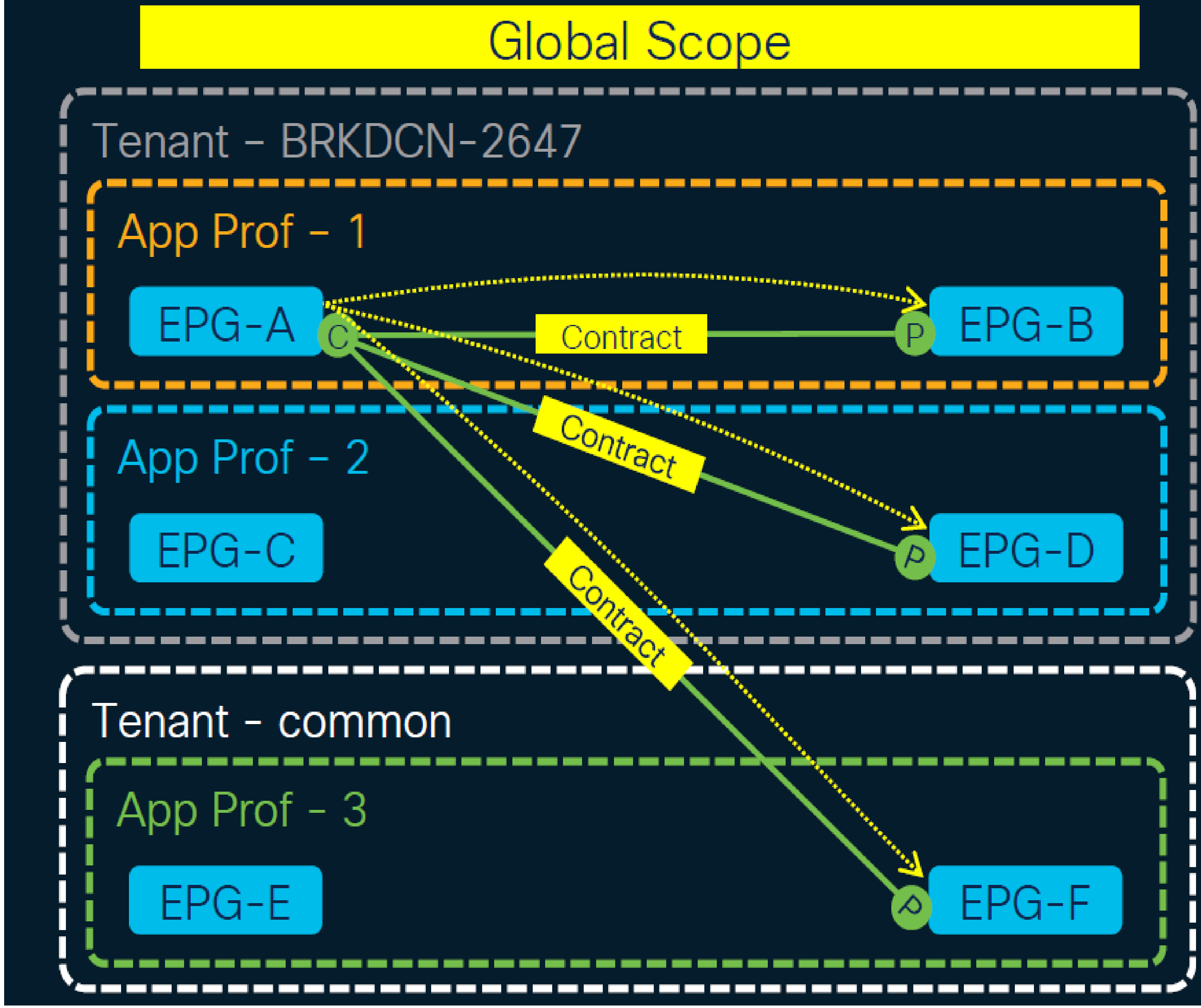
Contract Scope



Contract Scope – Application Profile

Contracts only permit EPGs to communicate within the same the same Application Profile.

Contract Scope



Default Contract Scope – Global

Contracts permit EPGs to communicate across Tenants and VRFs. Used for route-leaking.