Stuart Fordham

Cisco ACI Cookbook

Over 90 recipes to maximize automated solutions and policy-drive application profiles using Cisco ACI

Packt

Table of Contents

1
Understanding Components and the ACI Fabric

In this chapter we will cover:

- Understanding ACI and the APIC
- An overview of the ACI Fabric
- Converting Cisco Nexus NX-OS mode to ACI mode
- ACI Fabric Overlay
- An introduction to the GUI

Introduction

Cisco's **Application Centric Infrastructure** (**ACI**) is a big evolutionary step in data center networking. Not because it adds programmability to the network, this has been a rising trend over the last few years, but because of the increased compatibility between vendors. This is where the real benefits are.

We can see the start of this evolutionary step with Cisco's FlexPod (an amalgam of Cisco UCS, VMWare hypervisors, and NetApp storage). Here we see properly validated designs that span more than one vendor. This in itself was a big step, after all, it makes sense for one vendor to try and encourage the end-user to purchase their equipment instead of their competitors. This is done for two reasons; compatibility between devices and the vendors' financial success.

So, what of networks where one vendor can supply all of the equipment, from the networking to the storage, to the compute elements? It is actually quite rare to find an environment comprised of one single vendor in the real world, most networks (and I am

including virtualization platforms and storage within this term) have equipment from more than one vendor, because when you are looking for the best performance, you go with the big names (VMWare for virtualization, NetApp for storage and so on), because they have the longevity in the industry, the knowledge and support options that are required. The network becomes heterogeneous because it needs to be in order to fulfill user, application, and business demands.

The downside to this is that we lose some degree of compatibility. There are industry-standard protocols that give some level of compatibility back, such as **SNMP** (**Simple Network Management Protocol**), Syslog, and LLDP (Link Layer Discovery Protocol), that can facilitate alerting, logging and communication between devices, but ACI takes this all one step further, taking the heterogeneous data center network and making it, well, homogenous. Through ACI, the data center can be configured rapidly as the application demands and this includes physical and virtual network elements from multiple vendors. All of this can be performed through one GUI.

Before we dive in, let's take a few moments to understand what ACI is all about, dispelling some of the myths along the way.

- Myth: ACI is too expensive

ACI is not cheap to purchase, it is engineered for the data center, so commands data center process. Even the most basic of starter kits has a list price of $250,000. While a quarter of a million dollars is enough to get you started in the world of ACI, it is probably out of reach of most people. Even trying to sell ACI, as a "this could revolutionize our business" proposal, within most companies would be difficult. Despite the fact that most companies do not pay list price, ACI represents a huge risk, for a number of reasons.

ACI is in its infancy, so adoption will be slow. The companies that have the easily available financial resources to dive into it are, most likely, the same kind of businesses that are not typically early adopters. Established companies that have the cash have more accountability to stakeholders, shareholders, and the public, so are less likely to rush into investing six-figure sums, than the eager startup company, to whom $250,000 represents a massive proportion of their available funds.

Nevertheless, as ACI becomes more prevalent, its adoption rate will increase, despite the costs (which can always be negotiated).

- Myth: SDN (and ACI) will replace the engineer.

The idea of **Software-Defined Networking** (**SDN**) has caused quite a stir in the networking industry as engineers question whether having a programmable network will mean that the developer slowly takes their place. So, we have some degree of fear when it comes to ACI,

yet SDN and ACI only represents a small portion of the market. As the infrastructure scales up and out, SDN makes more sense. In smaller deployments, the costs outweigh the benefits, yet SDN (and ACI) will never replace the network engineer. The developer does not speak the language of networks in the same way that a traditional network engineer does not talk in development code. The two will remain separate entities in their little silos, ACI offers a bridge between the two, but both roles remain safeguarded.

So as much as ACI is expensive, data center specific, and occasionally perceived as a threat to the traditional network engineer, why should you look at it favorably?

- This is SDN, the Cisco way.

ACI allows the network administrator and the application developers to work closer together. Applications change, networks change. Both have lifecycles of varying length, and ACI allows for these lifecycles to coexist with each other and complement each other. Both teams can work together to achieve a common goal.

ACI reduces the complexity of the network, regarding deployment, management, and monitoring and does this through a common policy framework. Applications can be deployed rapidly, and the administrative overhead on the network is significantly reduced. It is, therefore, application-centric, and can facilitate services at layer 4 to 7 to enhance the application lifecycle.

Through ACI we can automate and program the network. We have a singular platform with which to provision the network. We can bring in, with ease, services such as virtualization (VMWare and Hyper-V), firewalls, load-balancers and a whole range of infrastructure that would, previously, have meant many hours being spent configuring and reconfiguring as the demands of the application change.

This automation is performed through policies. Policies are centrally configured on the **APIC (Application Policy Infrastructure Controllers)**, which are (usually) clustered.

The APIC is where we will start.

Understanding ACI and the APIC

ACI is for the data center. It is a fabric (which is just a fancy name for the layout of the components) that can span data centers using OTV or similar, overlay technologies, but it is not for the WAN. We can implement a similar level of programmability on our WAN links through **APIC-EM** (**Application Policy Infrastructure Controllers Enterprise Module**), which uses ISR or ASR series routers, along with the APIC-EM virtual machine to control and program them. APIC and APIC-EM are very similar, just the object of their focus is

different. APIC-EM is outside of the scope of this book, as we will be looking at data center technologies.

The APIC is our frontend. Through this, we can create and manage our policies, manage the fabric, create tenants and troubleshoot. Most importantly, the APIC is not associated with the data path. If we lose the APIC for any reason, the fabric will continue to forward the traffic.

To give you the technical elevator pitch, ACI uses a number of APIs (Application Programming Interfaces) such as REST (Representational State Transfer) using languages like JSON (JavaScript Object Notation) and XML (eXtensible Markup Language), as well as the CLI and the GUI, to manage the fabric and other protocols such OpFlex to supply the policies to the network devices. The first set (those that manage the fabric) are referred to as "northbound" protocols. Northbound protocols allow lower level network components talk to higher level ones. OpFlex (which we will discuss later in this chapter) is a "southbound" protocol. Southbound protocols (such as OpFlex and OpenFlow, which is another protocol you will hear in relation to SDN) allow the controllers to push policies down to the nodes (the switches).

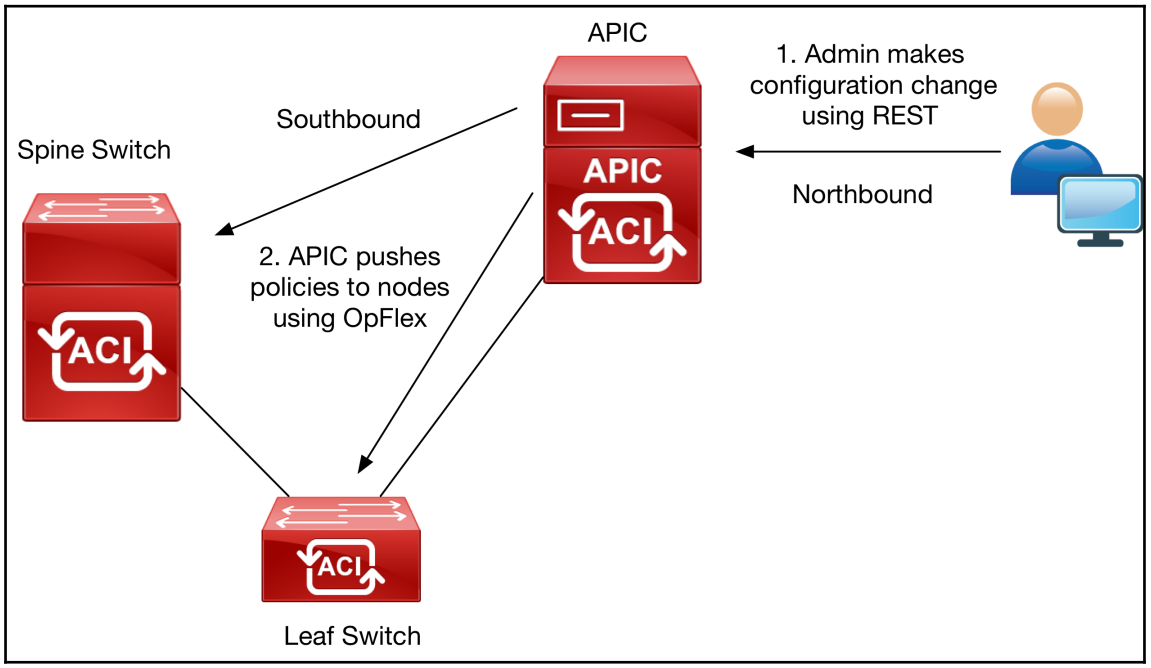


Figure 1.

That is a very brief introduction to the “how”. Now, let's look at the “why?”. What does ACI give us that traditional network does not?

In a multi-tenant environment, we have defined goals. The primary purpose is that one tenant should remain separate from another tenant. We can achieve this in a number of ways.

We could have each of the tenants in their own **DMZ** (**demilitarized zone**), with firewall policies to permit or restrict traffic as required. We could use VLANs to provide a logical separation between tenants. This approach has two drawbacks. It places a greater onus on the firewall to direct traffic, which is fine for northbound traffic (traffic leaving the data center), but is not suitable when the majority of the traffic is east-west bound (traffic between applications within the data center, see figure 2).

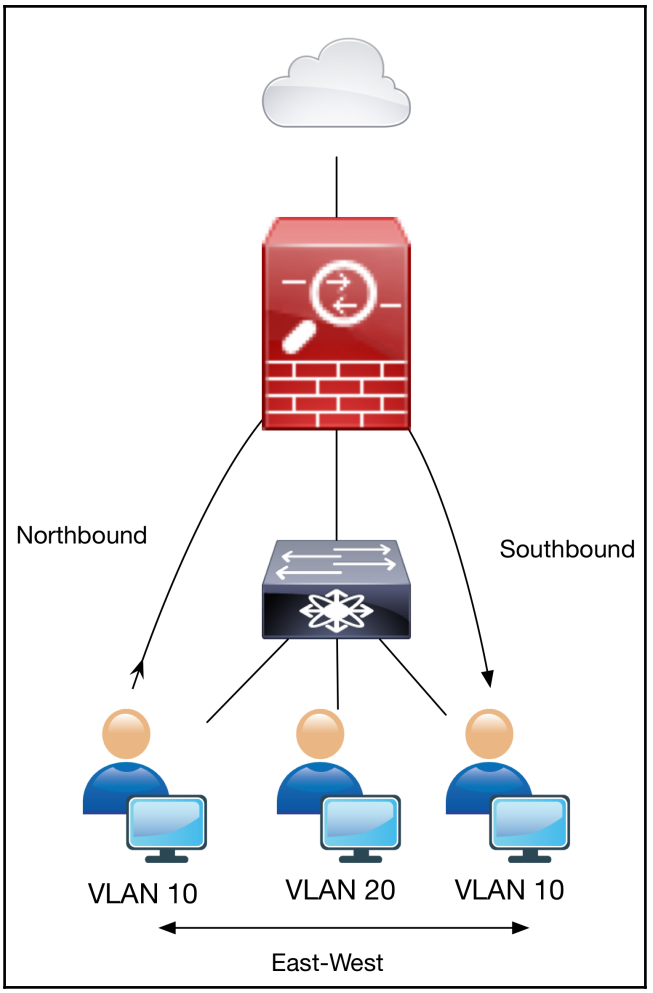


Figure 2.

We could use switches to provide layer-3 routing and use access-lists to control and restrict traffic, these are well designed for that purpose.

Also, in using VLANs, we are restricted to a maximum of 4096 potential tenants (due to the 12-bit VLAN ID).

An alternative would be to use **VRFs** (**Virtual Routing and Forwarding**). VRFs are self-contained routing tables, isolated from each other, unless we instruct the router or switch to share the routes, by exporting and importing **Route Targets** (**RT**). This approach is much better for traffic isolation, but when we need to use shared services, such as an Internet

pipe, VRFs can become much harder to keep secure.

One way around this would be to use route leaking. Instead of having a separate VRF for the Internet, this is kept in the global routing table and then leaked to both tenants. This maintains the security of the tenants, and as we are using VRFs instead of VLANs we have a service that we can offer to more than 4096 potential customers. But, we also have a much bigger administrative overhead. For each new tenant, we need more manual configuration, which increases our chances of human error.

ACI allows us to mitigate all of these issues.

By default, ACI tenants are completely separated. To get them talking to each other we need to create contracts, which specify what network resources they can and cannot see. There are no manual steps required to keep them separate from each other, and we can offer Internet access rapidly during the creation of the tenant. We also are not bound by the 4096 VLAN limit. Communication is through VXLAN, which raises the ceiling of potential segments (per-fabric) to 16 million (by using a 24-bit segment ID).

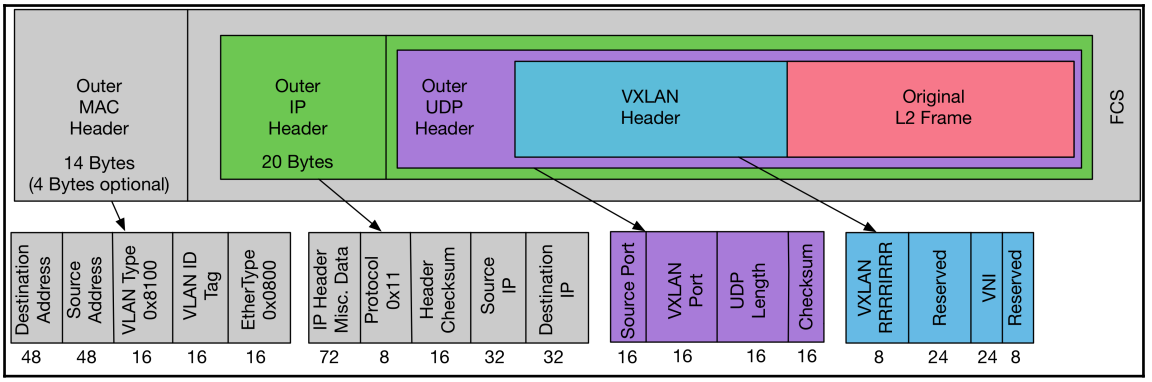


Figure 2

VXLAN is an overlay mechanism that encapsulates layer-2 frame within layer-4 UDP packets, also known as MAC-in-UDP (figure 2). Through this, we can achieve layer-2 communication across a layer-3 network. Apart from the fact that through VXLAN, tenants can be placed anywhere in the data center, and the number of endpoints that far outnumbers the traditional VLAN approach, the biggest benefit of VXLAN is that we are no longer bound by the Spanning Tree Protocol. With STP, the redundant paths in the network are blocked (until needed). VXLAN, by contrast, uses layer-3 routing, which enables it to use **equal-cost multipathing** (**ECMP**) and link aggregation technologies to make use of all the available links, with recovery (in the event of a link failure, in the region of 125 microseconds.

With VXLAN, we have endpoints, referred to as **VXLAN Tunnel Endpoints** (**VTEPs**), and these can be physical or virtual switchports. **Head-End Replication** (**HER**) is used to forward broadcast, unknown destination address, and multicast traffic, which is referred to (quite amusingly) as BUM traffic.

This 16M limit with VXLAN is more theoretical, however. Truthfully speaking, we have a limit of around 1M entries in terms of MAC addresses, IPv4 addresses, and IPv6 addresses, due to the size of the TCAM (**Ternary content-addressable memory**). The TCAM is a high-speed memory, used to speed up the reading of routing tables and performing matches against access control lists. The amount of available TCAM became a worry back in 2014 when the BGP routing table first exceeded 512 thousand routes, which was the maximum number supported by many of the Internet routers. The likelihood of having 1M entries within the fabric is also pretty rare, but even at 1M entries, ACI remains scalable in that the spine switches only let the leaf switches know about the routes and endpoints that they need to know about. If you are lucky enough to be scaling at this kind of magnitude, however, it would be time to invest in more hardware and split the load onto separate fabrics. Still, a data center with thousands of physical hosts is very achievable.

An overview of the ACI Fabric

A **fabric** is a fancy term for how the computing, network and software components of a data center are laid out. The name itself comes from the crisscross of the network, much like an item of weaved clothing. The ACI fabric is relatively simplistic. It employs a two-tier design made of **spine** and **leaf** switches, but very particular switches.

ACI hardware

In figure 3, we can see a typical deployment with two spines and three leaves. The Nexus 9500 modular switches are deployed at the top of the topology and act as spines, which in a traditional three-tiered network design would be the aggregation or core switches. Line cards are used to provide the **ASIC**s (**Application Specific Integrated Circuitry**) required for ACI. There are different line cards available, so make sure that you are not purchasing an NX-OS mode only card.

The next component is the leaf switches. These are the Nexus 9300 series switches.

The spines connect to the leaves through 40 GE ports, but the spines and leaves are never connected (spine to spine, or leaf to leaf).

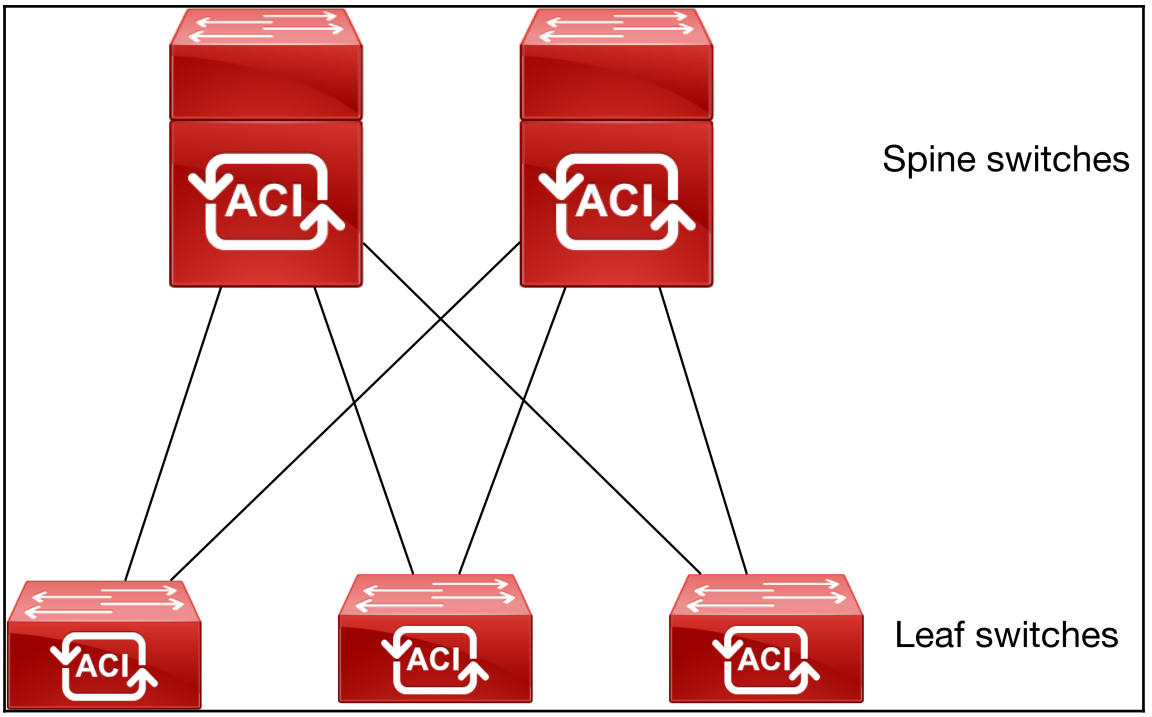


Figure 3

We can also extend the network, offering greater port density through Nexus 2000 series fabric extenders:

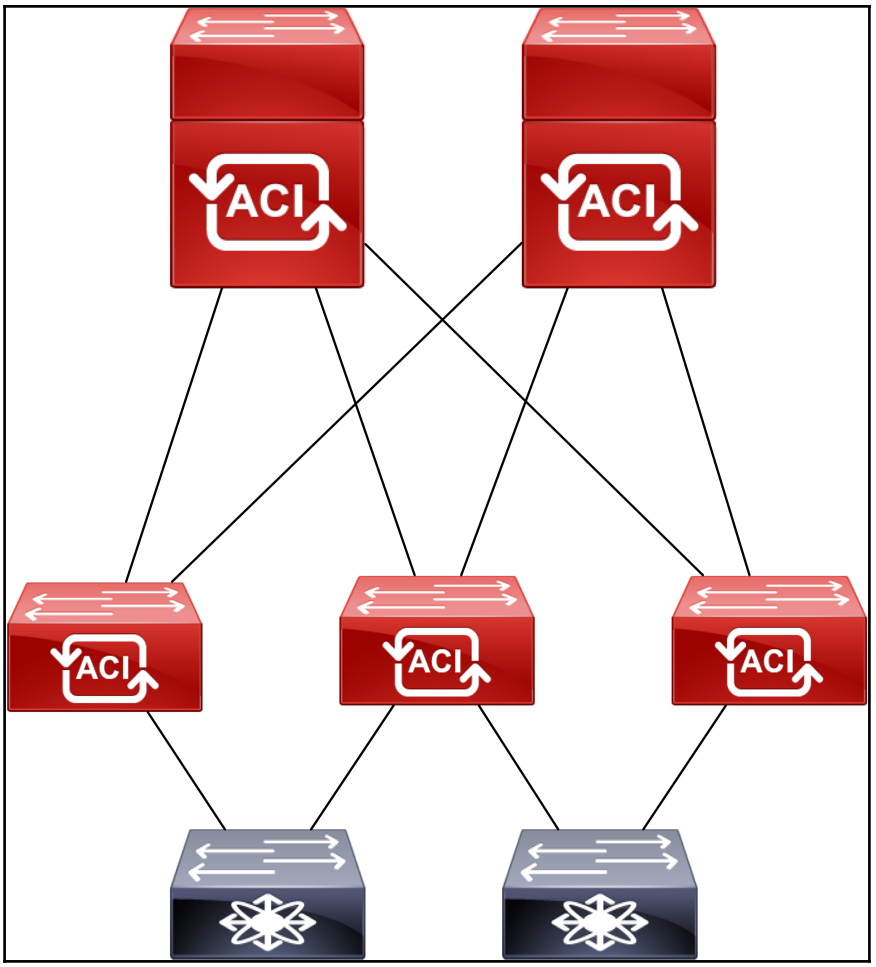


Figure 4

We can also add storage, directly into the leaves themselves

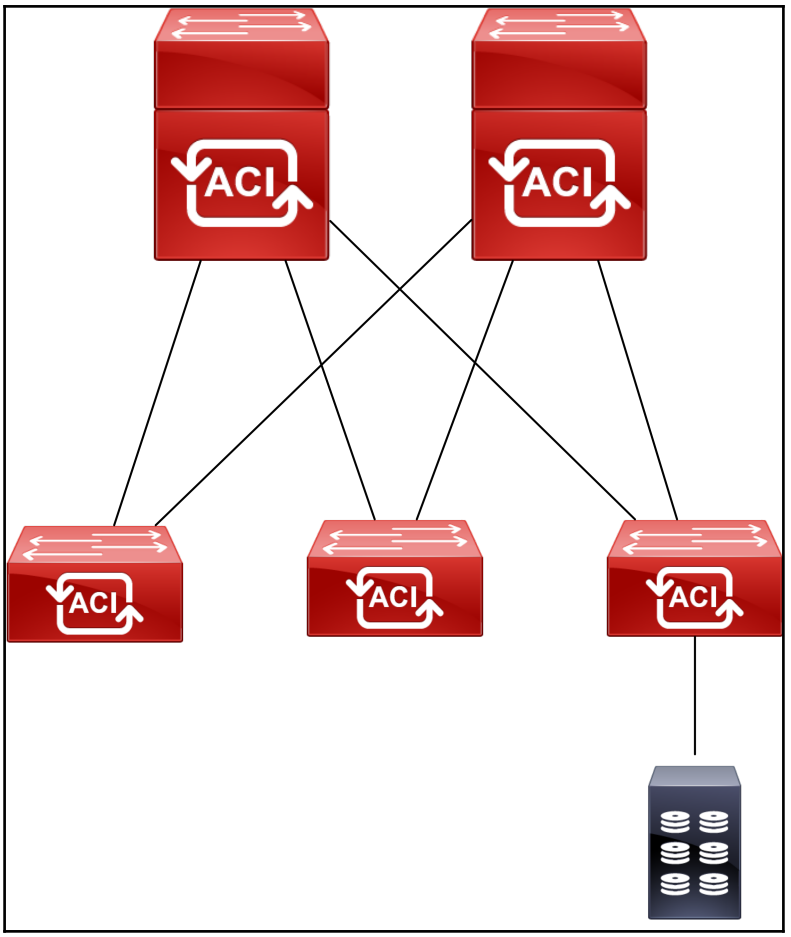


Figure 5

Or, using another pair of switches, such as Cisco Nexus 5000 series switches, which would connect to the leaf switches:

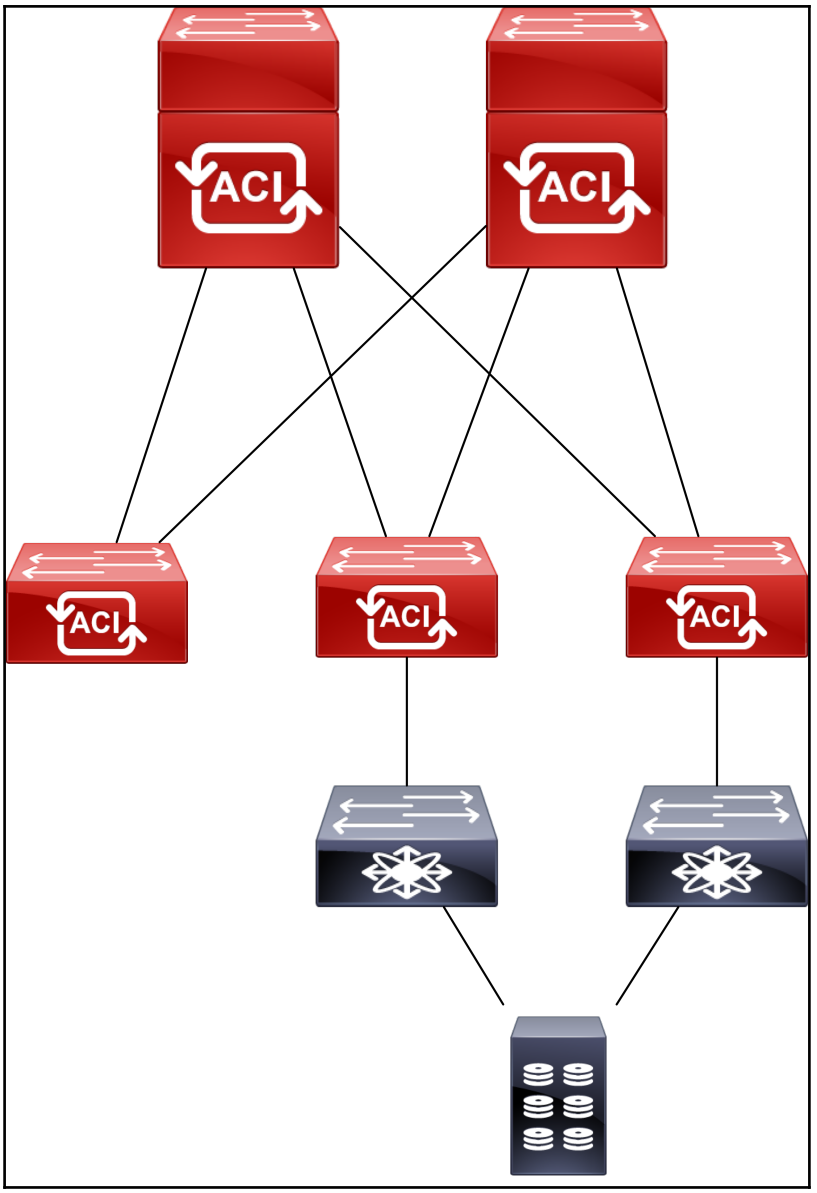


Figure 6

The APIC controllers connect to the leaf switches.

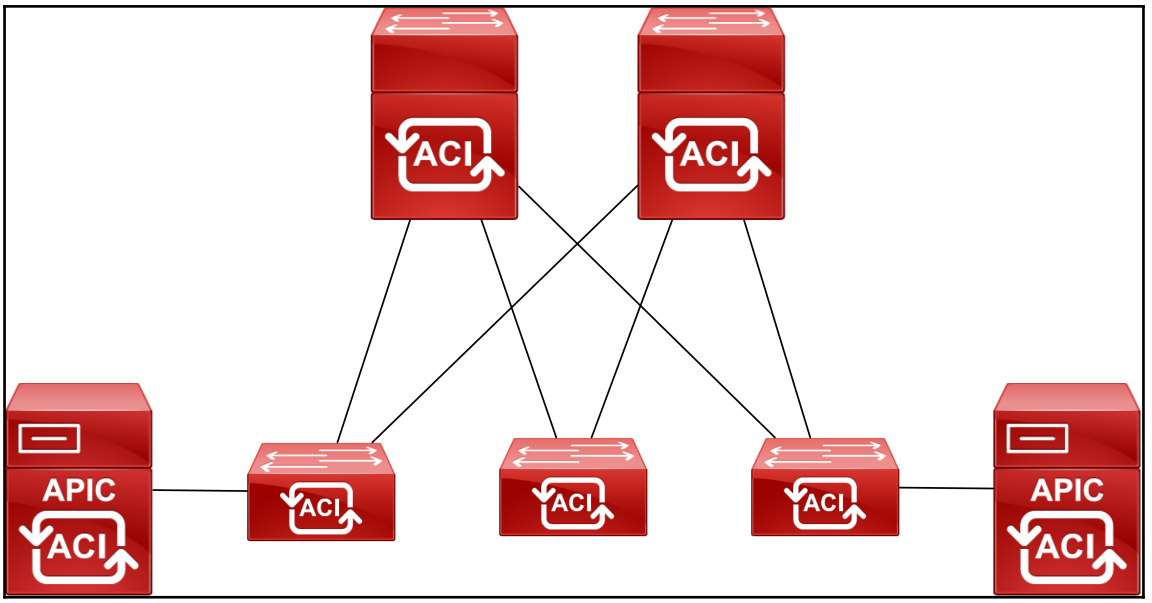


Figure 7

The APIC controllers are completely separate from the day-to-day running of ACI. We need them to push new policies. However, they do not play a part in the functioning of the ACI fabric's data plane, only in the control plane.

The control plane is what we learn, such as routing information. The data plane is the movement of packets based upon information in the control plane.

If we lose one controller, then our tenants' traffic still flows. If we lose all our controllers, then our tenants' traffic still flows. We are just unable to push new policies until the controllers are reinstated into the network.

ACI best-practice states that we should have a minimum of three controllers. Having three controllers offers high availability, three offers physical redundancy as well as database redundancy. So, why not two controllers? Three controllers (well, just an odd number greater than one) works better in a split-brain scenario (one controller disagreeing with another). in such an event, the majority would rule. The controllers use LLDP to find each other, which is part of the process discussed in the ACI fabric overlay section later on in this chapter. We will look at how to use multiple controllers in the troubleshooting section, as the majority of this book uses a much simpler design with just one controller, one spine and two leaf switches, as seen when we look at the fabric menu, later on in this chapter.

Understanding third-party integration

One of the most attractive reasons to deploy ACI is the ease of integration with other Cisco products (such as the ASAv firewall) and third-party systems.

This integration is performed through **OpFlex**. OpFlex is an open standards-based southbound protocol, designed to facilitate multi-vendor integration in both data center and cloud networks. OpFlex is important as it differentiates ACI from other SDN models, which have integration but do not support the full feature-set. The easiest way to try and explain this would be to look at it in the context of SNMP.

SNMP (**Simple Network Machines Protocol**) allows monitoring of network hardware, and all devices support the most basic **MIB** (**Management Information Base**) of `iso.org.dod.internet.mgmt`, so at the most basic level, you can pull out data such as interfaces, IP addresses and so on. We are getting data but at the lowest common denominator. We need extra information, by way of specific MIBs, to be able to monitor our firewall's VPN tunnels or the nodes on our load balancers. OpFlex gives us all the information, but the data is not bound to any particular format. It is a declarative model, which benefits any interested party. This declarative model is based on promise theory.

Promise theory, developed by Mark Burgess in the 1990s, sets ACI aside from other SDN implementations. They use imperative control, in which we have a controlling system, and the system being controlled is relieved of the burden of doing the thinking. While this does offer more autonomy to the controller, it can also create a bottleneck within the system. ACI, however, uses a declarative model. This model states what should happen, but not how it should be done (leaving that up to the node being controlled). The node then makes a promise to achieve the desired state and, importantly, communicates back to the controller the success or failure of the task, along with the reason why. The controller is no longer a bottleneck in the system, and the commands are simpler; instead of separate commands to implement the same function on different vendor equipment, we have one command set understandable by both vendors equipment. This is the benefit of open standards.

Even with open standards, though, there can be some ulterior motive. It is all well and good having the next-best thing for integrating different technologies, but when this is designed for those technologies to run under one particular companies product, there can be some hesitation. However, there is a large backing from several very well known companies, such as Microsoft, IBM, Citrix, RedHat, F5, SunGard Availability Services, and Canonical. So, why has OpFlex gathered such a wide backing?

With the *traditional* SDN model, there is a bottleneck; the SDN controller. As we scale out there is an impact in both performance and resiliency. We also lose simplicity and agility;

we still need to make sure that all the components are monitored and safeguarded, which invariably means *bolting on* more technology to achieve this.

OpFlex takes a different approach. A common language ensures that we do not need to add any extras that are not part of the original design. There is still complexity, but this is moved towards the edges of the network, and we maintain resiliency, scalability, and simplicity. If we lose all of the controllers, then the network continues to operate, we may not be able to make policy changes until we restore the controllers, but the tenant's data still flows, uninterrupted.

The protocol itself uses XML or JSON as the transmission medium. It allows us to see each node as a **managed objec**t (**MO**). Each MO consists of the following:

- Properties
- Child Relations
- Parent Relations
- MO Relations
- Statistics
- Faults
- Health

While the ins-and-outs of these are beyond the scope of this book, you can read about them more in the IETF drafts. The first one in 2014 (`https://tools.ietf.org/html/draft-smith-opflex-00`) listed all seven items above, but subsequent drafts, the most recent being October 27th, 2016 (`https://tools.ietf.org/html/draft-smith-opflex-03`) compress the last four items into one, labeled *observables*.

What this all means is that for third-parties, OpFlex means greater integration across SDN platforms. If and when OpFlex does become a truly open standard then, by using a simple JSON file, different vendors equipment can speak the same language.

Converting Cisco Nexus NX-OS mode to ACI mode

To use ACI, we need to make sure that we are running our switches in ACI mode. We can check which version we are running by using the `show version` command:

```
BIOS: version 08.06
NXOS: version 6.1(2)I3(3)
BIOS compile time: 12/03/2014
```

```
NXOS image file name is: bootflash:///n9000-dk9.6.1.2.I3.3.bin
NXOS compile time: 12/05/2014 10:50:20 [12/05/2014 2:25]
```

We can tell that we are running an NX-OS mode switch as the image filename begins with `n9000`. ACI image filenames begin with `aci-n9000`.

The instructions below are for NX-OS release 6.1(2)l3(3) and above, and ACI image version 11.0(2x) or later. There are slight differences with earlier releases, so it is best to make sure you are on the above releases before attempting the switch from NX-OS mode to ACI-Mode.

Check hardware is supported – look in release notes for Cisco Nexus 9000 Series ACI-Mode Switches.

Remove or turn off any unsupported module (`poweroff module <module>` command). If you do not do this step, the software will use a recovery/retry mechanism before powering down the unsupported module, which can cause delays.

If you have a dual-supervisor system, then make sure that the standby supervisor module is in the ha-standby state using the command `show module`.

Use: *`show install all impact epld <epld-image-name>`* to check that the switch does not require any EPLD image upgrade. The EPLD is the Electronic Programmable Logic Device and these enhance hardware functionality or to resolve known issues. EPLD upgrades are quite infrequent, but they should not be overlooked.

Uploading the ACI image

We have a number of ways of performing the upgrade. We can use SCP to copy the image from the APIC to the switch, or from another SCP server, or copy it directly from a USB port. We will look at all three methods and are assuming that the Nexus switch has already been introduced into the network and has connectivity.

A word of warning when using USB drives, though. Smaller is better. Taking a 1TB drive loaded with all your favorite Nexus images and expecting it to work will only leave you hunting around for a 2GB drive that has sat in a drawer gathering dust for a few years. This is due to the level of filesystem support. Older IOS versions only supported FAT16, which has a filesize limit of 2GB, newer ones support FAT32 (such as IOS 15.1). Sometimes it is easier to play it safe and go with the FAT16.

How to do it…

Method 1: Using SCP to copy the ACI image from the APIC

1. Enable `SCP` on the Nexus switch:

```
switch(config)# features scp-server
```

2. Copy the image from the APIC server to the Nexus switch using the CLI:

```
scp -r /firmware/fwrepos/fwrepo/<switch-image-name> admin@switch-ip-address:switch-image
```

Method 2: Using SCP to copy the ACI image from another SCP server

1. Copy the file from the SCP server using the switch's command line:

```
Switch# copy scp: bootflash:
```

You will be prompted for the details if the SCP server and filenames.

Method 3: Using a USB drive to copy the ACI image

We can copy an image from a USB drive to the bootflash, using the "dir" command first, so that we can cut and paste the filename in the copy command.

```
Switch# dir usb1:
(or dir usb2: depending on which USB slot you have plugged the drive into)
Switch# copy usb1:<ACI-image-name> bootflash:
```

If we have a dual-supervisor system, we have an additional step, which is to copy the ACI image to the standby supervisor module:

```
Switch(config)# copy bootflash:aci-image bootflast://sup-standby/
```

Upgrading the image

The next step is to upgrade the image.

How to do it…

In the following code, we first turn off NXOS mode. We then make sure that the first change survives a reboot. In the third line, we boot the supervisor modules using the ACI image specified. Lastly, we perform a reload of the switch.

```
Switch(config)# no boot nxos
Switch(config)# copy running-config startup-config
Switch(config)# boot aci bootflash:aci-image-name
Switch(config)# reload
```

Logging in

Once the switch has rebooted with the new image, we can log in.

How to do it…

We log in using the username admin and the password specified during setup. Notice that the fabric discovery process is started at this point. It may be some minutes before the services start and we are able to access the switch via the console.

```
User Access Verification
(none) login: admin
**************************************************************************
**
    Fabric discovery in progress, show commands are not fully functional
    Logout and Login after discovery to continue to use show commands.
**************************************************************************
**
(none)#
```

Reverting to NX-OS mode

If, for any reason, you need to revert to NX-OS mode from ACI Mode, then follow these steps:

1. Reload the switch

   ```
   admin@apic1:aci> reload
   ```

2. Access the bootloader

```
Ctrl+]
Loader>
```

3. Boot using the NX-OS image

```
loader> boot nxos-image-name
```

This can take a little while (usually under half an hour) while the system reformats the system to make subsequent reloads faster.

As you can see, from the screenshot above, the switch performs a fabric discovery. We will look at this in the next section.

ACI Fabric Overlay

ACI uses **Inter-Fabric Messaging** (**IFM**) to communicate between the different nodes. IFM uses TCP packets, which are secured by 1024-bit SSL encryption, and the keys are stored on secure storage. The **Cisco Manufacturing Certificate Authority** (**CMCA**) signs the keys.

Issues with IFM can prevent fabric nodes communicating and from joining the fabric. We will cover this in greater depth in *Chapter 9*, but we can look at the output of the checks on a healthy system:

```
apic1# netstat -ant | grep :12
tcp        0    0 10.0.0.1:12151          0.0.0.0:*          LISTEN
tcp        0    0 10.0.0.1:12215          0.0.0.0:*          LISTEN
tcp        0    0 10.0.0.1:12471          0.0.0.0:*          LISTEN
tcp        0    0 10.0.0.1:12279          0.0.0.0:*          LISTEN
<truncated>
tcp        0    0 10.0.0.1:12567          10.0.248.29:49187 ESTABLISHED
tcp        0    0 10.0.0.1:12343          10.0.248.30:45965 ESTABLISHED
tcp        0    0 10.0.0.1:12343          10.0.248.31:47784 ESTABLISHED
tcp        0    0 10.0.0.1:12343          10.0.248.29:49942 ESTABLISHED
tcp        0    0 10.0.0.1:12343          10.0.248.30:42946 ESTABLISHED
tcp        0    0 10.0.0.1:50820          10.0.248.31:12439 ESTABLISHED
apic1# openssl s_client -state -connect 10.0.0.1:12151
CONNECTED(00000003)
SSL_connect:before/connect initialization
SSL_connect:SSLv2/v3 write client hello A
SSL_connect:SSLv3 read server hello A
depth=1 O = Cisco Systems, CN = Cisco Manufacturing CA
verify error:num=19:self signed certificate in certificate chain
verify return:0
SSL_connect:SSLv3 read server certificate A
SSL_connect:SSLv3 read server key exchange A
```

```
SSL_connect:SSLv3 read server certificate request A
SSL_connect:SSLv3 read server done A
SSL_connect:SSLv3 write client certificate A
SSL_connect:SSLv3 write client key exchange A
SSL_connect:SSLv3 write change cipher spec A
SSL_connect:SSLv3 write finished A
SSL_connect:SSLv3 flush data
SSL3 alert read:fatal:handshake failure
SSL_connect:failed in SSLv3 read server session ticket A
139682023904936:error:14094410:SSL routines:SSL3_READ_BYTES:sslv3 alert
handshake failure:s3_pkt.c:1300:SSL alert number 40
139682023904936:error:140790E5:SSL routines:SSL23_WRITE:ssl handshake
failure:s23_lib.c:177:
---
Certificate chain
0 s:/CN=serialNumber=PID:APIC-SERVER-L1 SN:TEP-1-1, CN=TEP-1-1
   i:/O=Cisco Systems/CN=Cisco Manufacturing CA
1 s:/O=Cisco Systems/CN=Cisco Manufacturing CA
   i:/O=Cisco Systems/CN=Cisco Manufacturing CA
---
Server certificate
-----BEGIN CERTIFICATE-----
<runcated>
-----END CERTIFICATE-----
subject=/CN=serialNumber=PID:APIC-SERVER-L1 SN:TEP-1-1, CN=TEP-1-1
issuer=/O=Cisco Systems/CN=Cisco Manufacturing CA
---
No client certificate CA names sent
---
SSL handshake has read 2171 bytes and written 210 bytes
---
New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: zlib compression
Expansion: NONE
SSL-Session:
   Protocol : TLSv1.2
   Cipher   : DHE-RSA-AES256-GCM-SHA384
   Session-ID:
   Session-ID-ctx:
   Master-Key: 419BF5E19D0A02AA0D40BDF380E8E959A4F27371A87EFAD1B
   Key-Arg   : None
   PSK identity: None
   PSK identity hint: None
   SRP username: None
   Compression: 1 (zlib compression)
   Start Time: 1481059783
```

```
    Timeout   : 300 (sec)
    Verify return code: 19 (self signed certificate in certificate chain)
---
apic1#
```

IFM is essential in the success of the discovery process. A fabric node is only considered *active* when the APIC and the node can exchange heartbeats, through IFM. Going forward, though, we still need IFM once we have active nodes, as it is also used by the APIC to push policies to the fabric leaf nodes

The fabric discovery process has three stages and uses IFM, LLDP (Link Layer Discovery Protocol), DHCP (Dynamic Host Configuration Protocol) and TEPs (Tunnel Endpoints).

- Stage 1:

  The leaf node that is directly connected to APIC is discovered.

- Stage 2:

  A second discovery brings in any spines connected to initial “seed” leaf.

- Stage 3:

  In this stage, we have the discovery of other leaf nodes and other APICs in the cluster.

The process can be visualized as in figure 8:

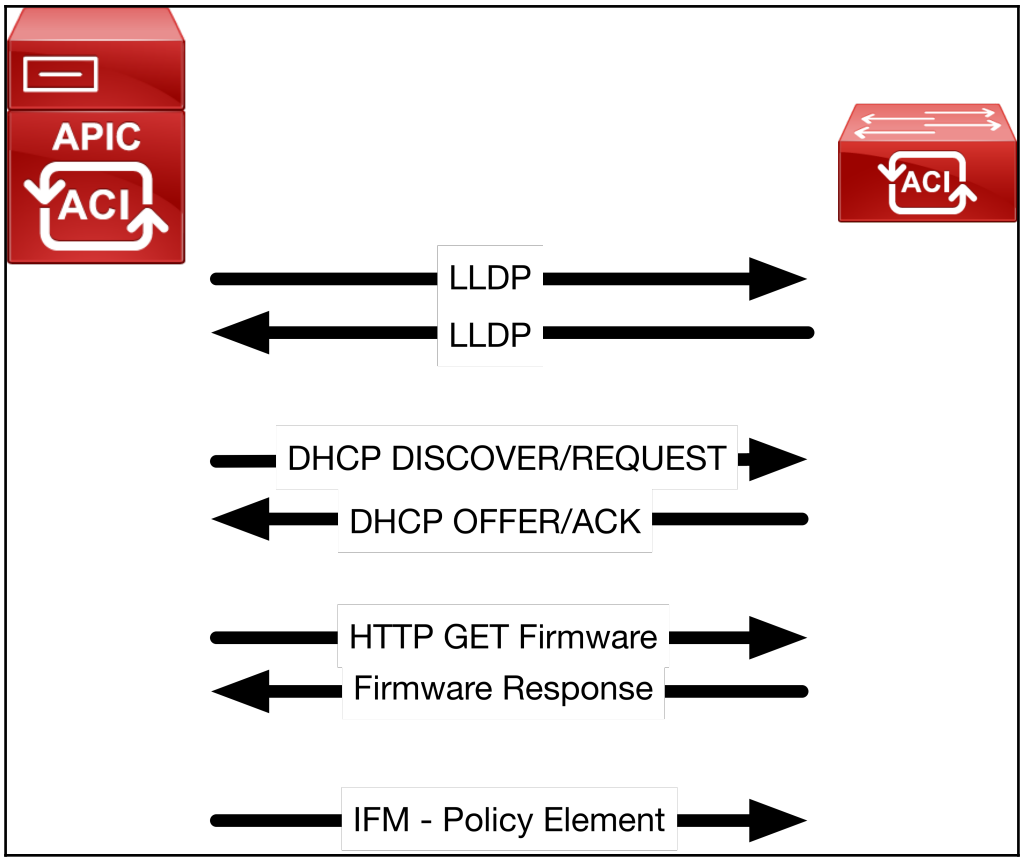


Figure 8

The node can transition through a number of different states during the discovery process:

- Unknown – node discovered but no node ID policy configured
- Undiscovered – Node ID configured but not yet discovered
- Discovering – node discovered but no IP address assigned
- Unsupported – node is not a supported model
- Disabled – node has been decommissioned
- Inactive – no IP connectivity
- Active – node is active

Using the command `acidiag fnvread` can see the current state. Below, the leaf node is in a state of "**unknown**" (note that I have removed the final column in the output that was

"LastUpdMsg", the value of which was 0:

```
apic1# acidiag fnvread
ID Pod ID Name      Serial Number   IP Address      Role          State
--------------------------------------------------------------------------
 0      0              TEP-1-101      0.0.0.0  unknown         unknown
Total 1 nodes
apic1#
```

During fabric registration and initialization a port may transition to an *out-of-service* state. In this state, the only traffic permitted is DHCP and CDP or LLDP. There can be a number of reasons why we would transition to this state, but these are generally due to human error, such as cabling or LLDP not being enabled, again, these are covered in chapter 9.

There are a couple of ways in which we can check the health of our controllers and nodes. We can use the CLI to check LLDP (*`show lldp neighbors`*), or we can use the GUI (**System | Controllers | Node | Cluster as Seen By Node**):

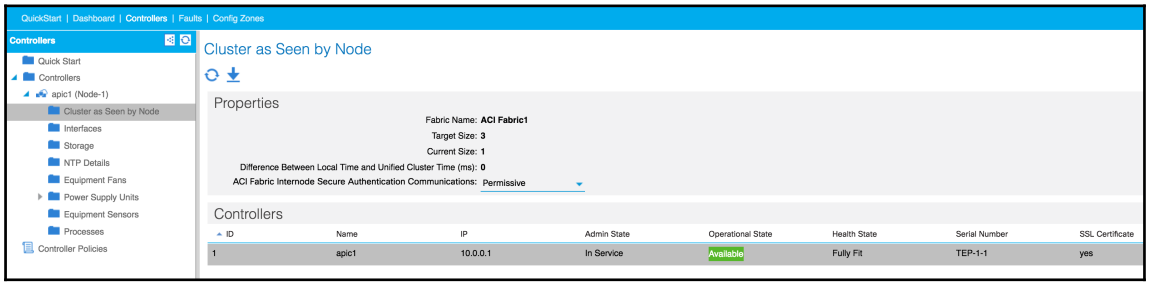


Figure 9

This shows us the APIC, and we can look at our leaf nodes from the Fabric menu. In the code output from *`acidiag fnvread`*, we saw a node named `TEP-1-101`. This is a leaf node, as we can see from the GUI (Fabric > Inventory > Fabric Membership):

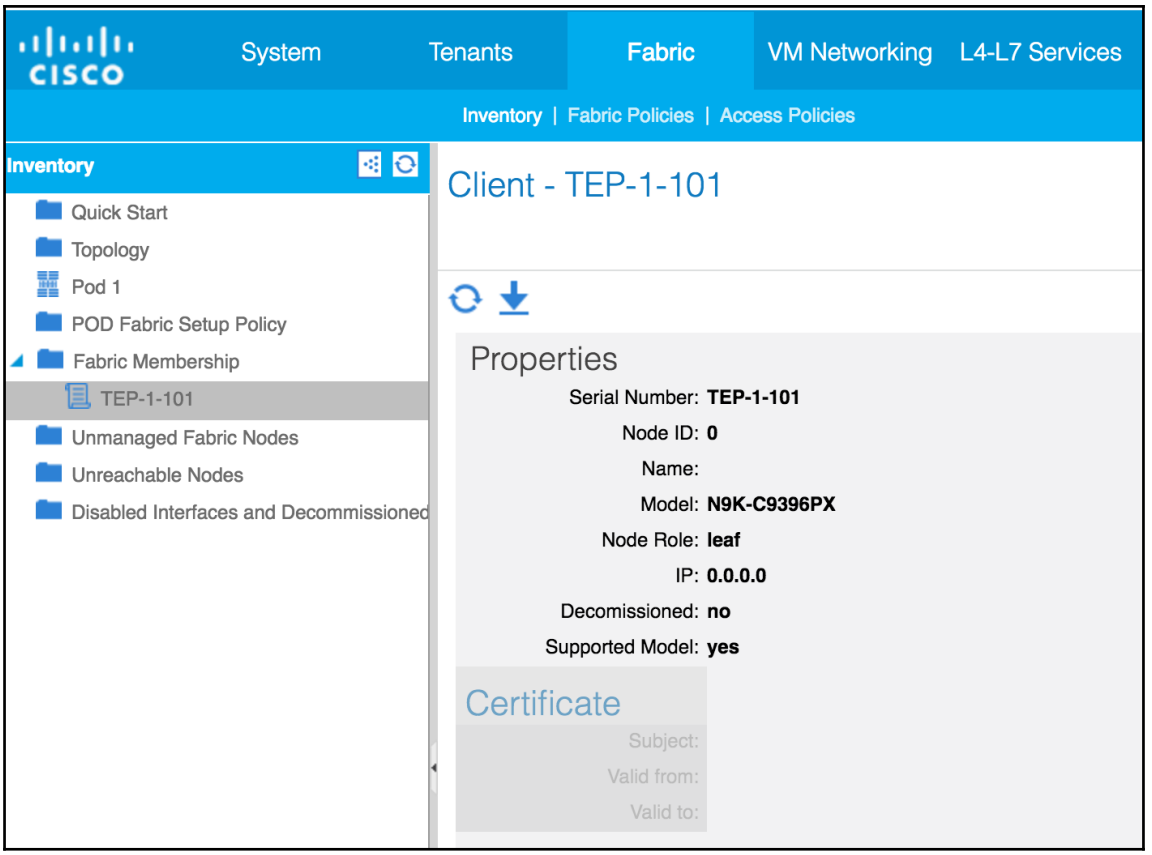


Figure 10

We will look at the GUI in the next section.

An introduction to the GUI

On accessing the APIC, we are presented with the login page. It can take a few minutes for the system to fully initialize before we can log in.

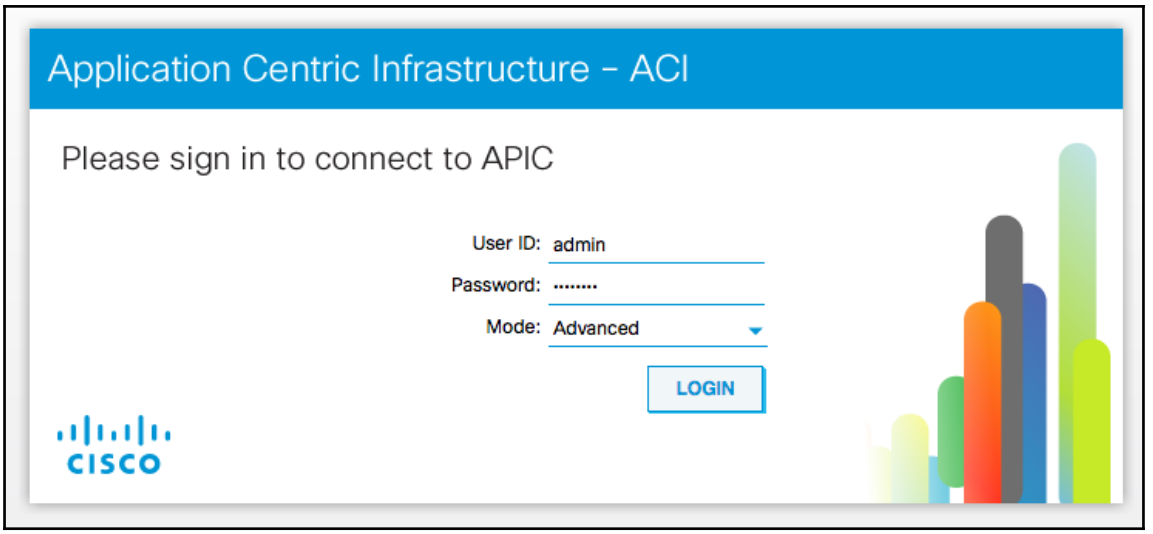


Figure 11

Once we have successfully logged in we are shown the System page. We have the main menu at the top, and each menu item has a submenu.

Here we can see the System menu, with its submenu showing Quickstart, Dashboard, Controllers, Faults and Config Zones:

System menu

The system page shows us the health of the system, separated into the overall system health and nodes and tenants with under 99% health. Faults counts are listed on the right-hand side, by domain, and by type. We can see the controller status in the bottom right-hand corner.

Figure 12

Moving on to the Controllers submenu we can see the screen from the previous section.

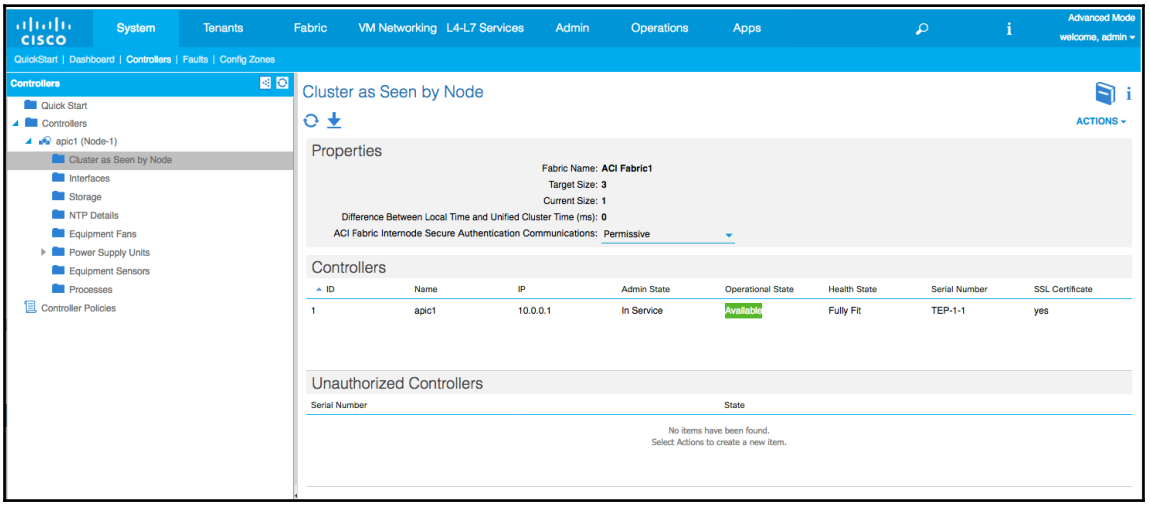


Figure 13

We have one controller (apic1), its IP address is 10.0.0.1, it is in service, available and the health state is “Fully Fit.” If we click on any of the column headings, we can sort them by ascending or descending order, which is useful if you have a large number of controllers.

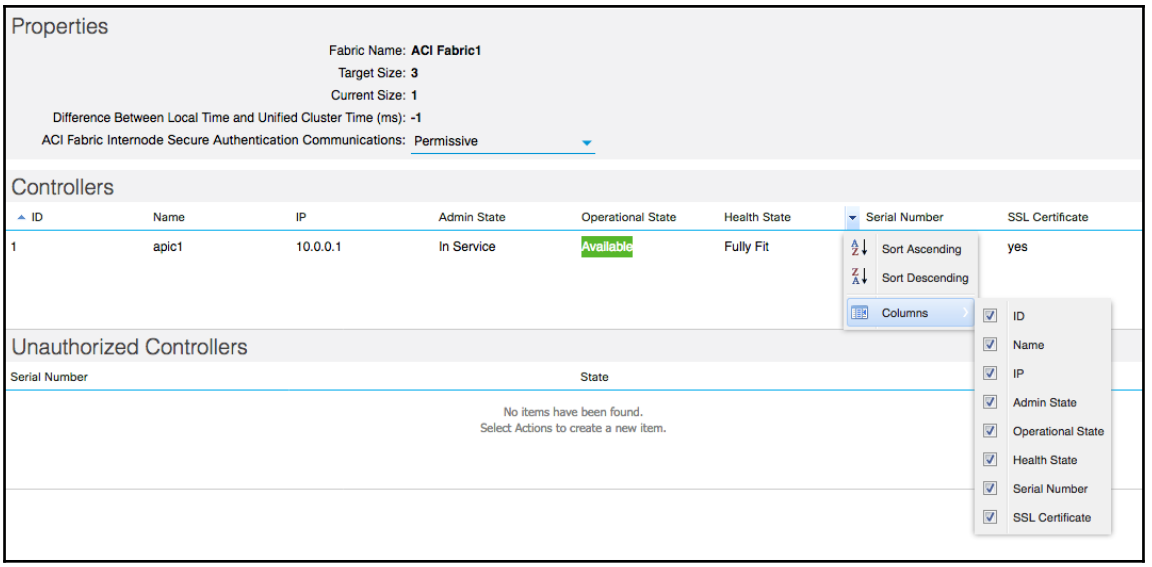


Figure 14

We can see the interfaces present on our controller (apic1) from the Interfaces menu on the left-hand side:

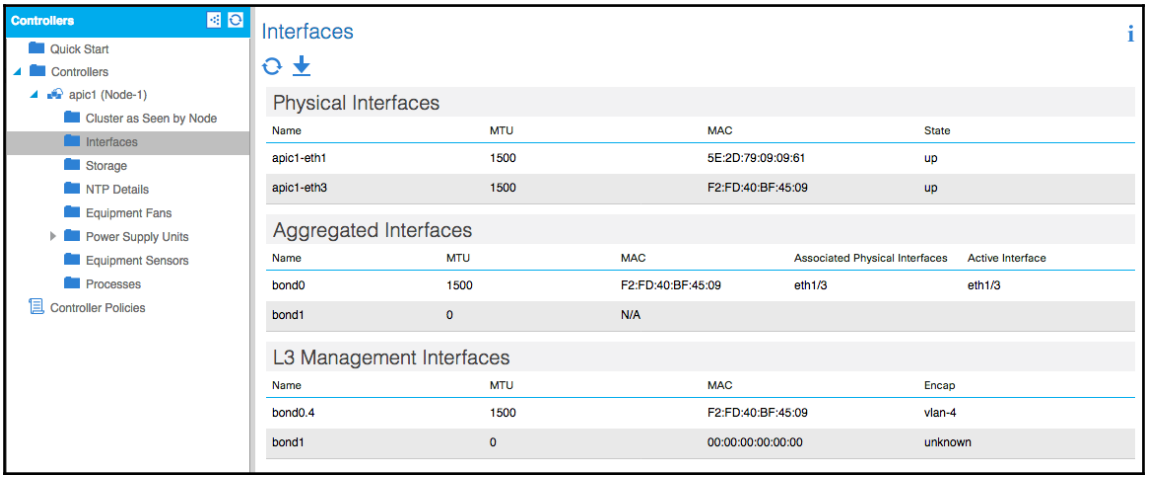


Figure 15

We can keep track of how much storage we have used from the Storage menu option, and again, this is sortable by clicking on the column heading:

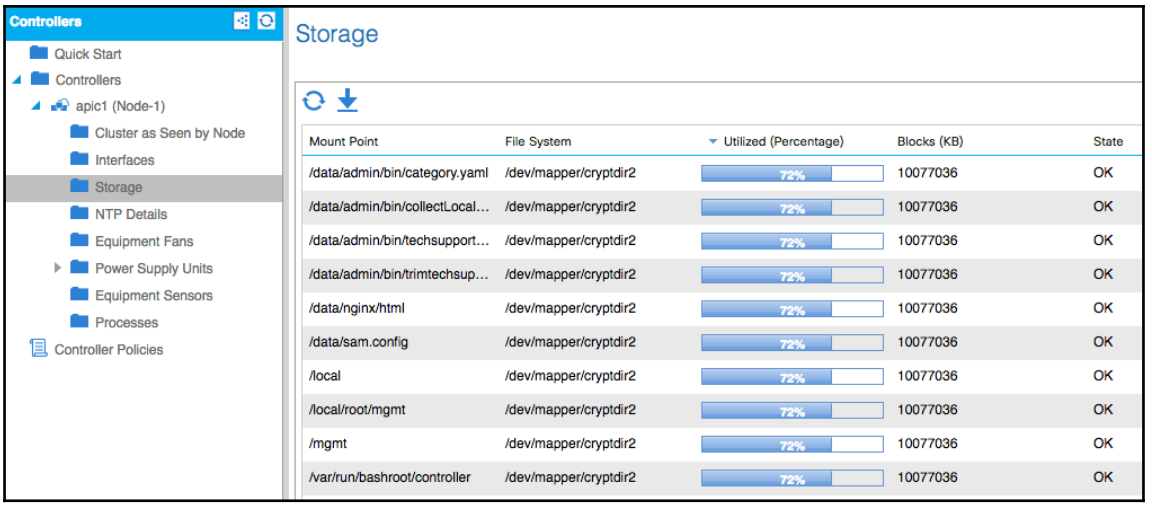


Figure 16

You will notice that the screen flickers every few seconds as it refreshes.

Also in this section, we can see stats on our NTP servers, our fans and power supply units and the equipment sensors (which in the simulator are all empty). Under **Processes**, we can see how much memory we are currently using from the **Stats** option:

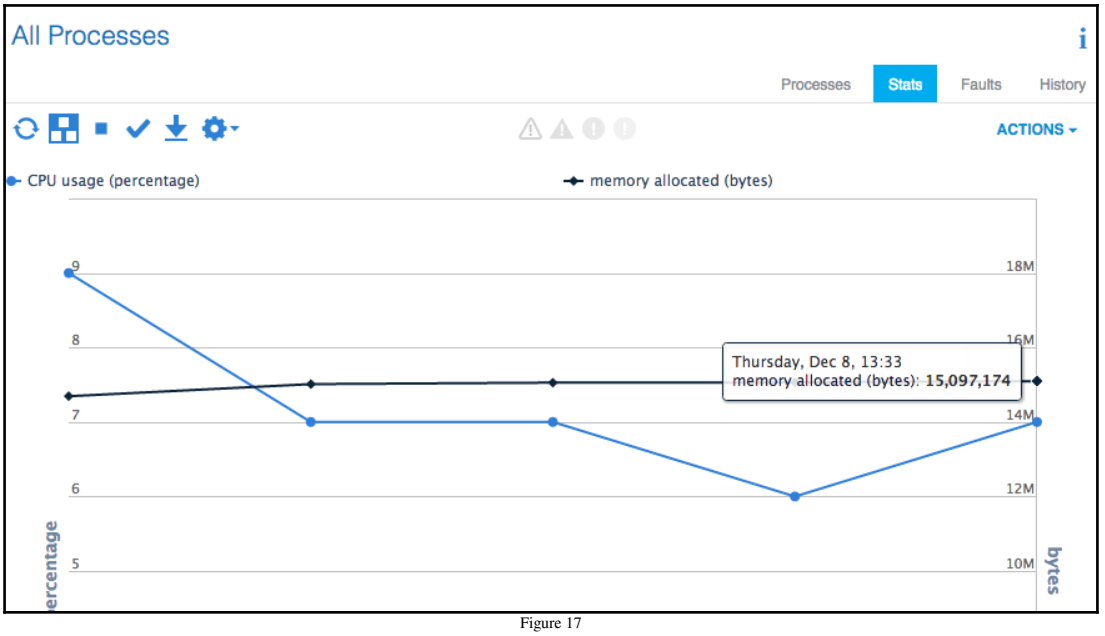


Figure 17

We can also check on the CPU usage from the same window.

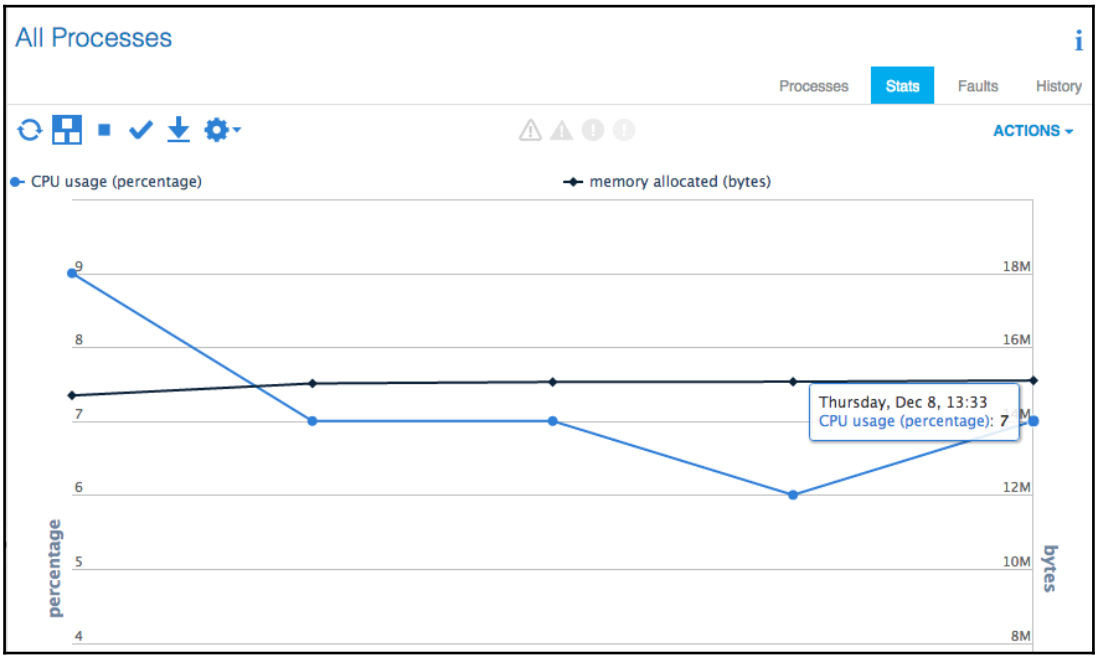


Figure 18

We can also see current and historical faults end events from the Processes menu.

The last menu option under **Controllers** is **Controller Policies**, here we can set policies for exporting diagnostic information. We will look at this in the troubleshooting section.

The final two options are Faults, in which we can see a couple of examples and Config Zones.

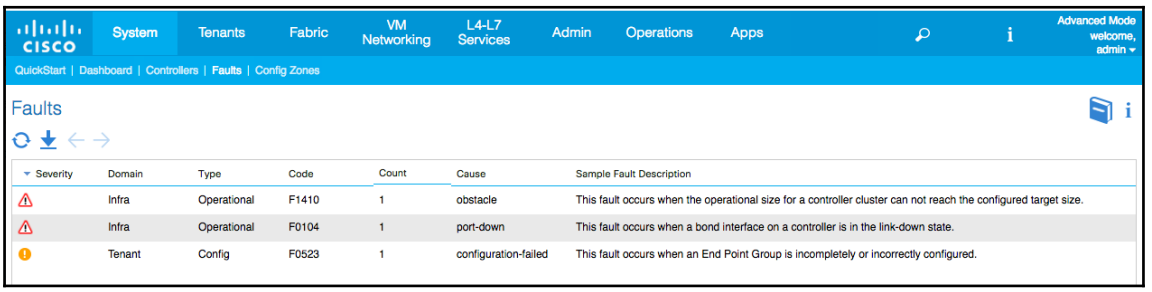


Figure 19.

We do not have any Zones, but we can create one from the drop-down menu. Configuration Zones allow us to sub-divide our ACI fabric, meaning that we can make configuration changes to one zone at a time, reducing the chances of an error affecting the entire fabric. Note that if you get access to the Cisco Devnet sandbox environment, that the Config Zones option is unavailable. Devnet is a great place to learn the developer-side of Cisco's products, as well as getting access to hands-on labs and virtual and hardware-based sandbox environments. The virtual ones are fairly limited, but available all the time. The physical rack equipment offers the full range of functionality, but does get booked up months in advance. You can find more about Devnet by going to the Devnet site: `https://developer.cisco.com/site/devnet/home/index.gsp`.

Tenants menu

The Tenants tab shows us all our tenants. We have three pre-configured (common, infra and mgmt.):

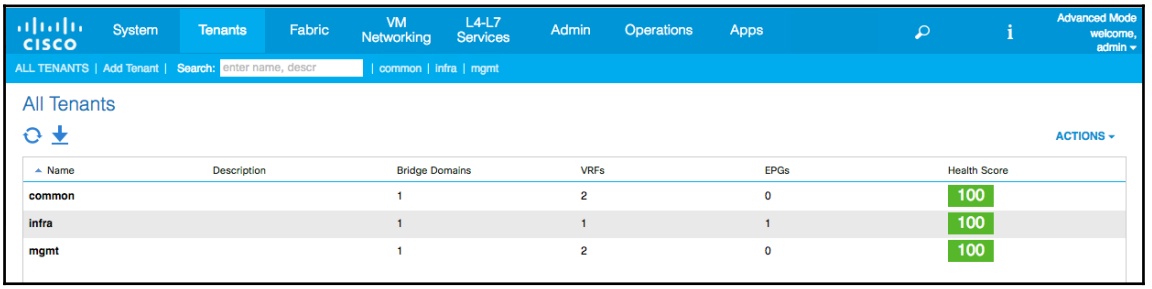


Figure 20.

If we select a tenant and go through the options, we can see the Application Profiles assigned to it, the Networking configuration, Layer4-Layer7 Service Parameters, and all of the policies. We will go through these in greater detail in the next chapter when we set up some tenants.

This is where we would create new tenants.

Fabric menu

From the fabric menu and the Inventory submenu, we can see our topology:

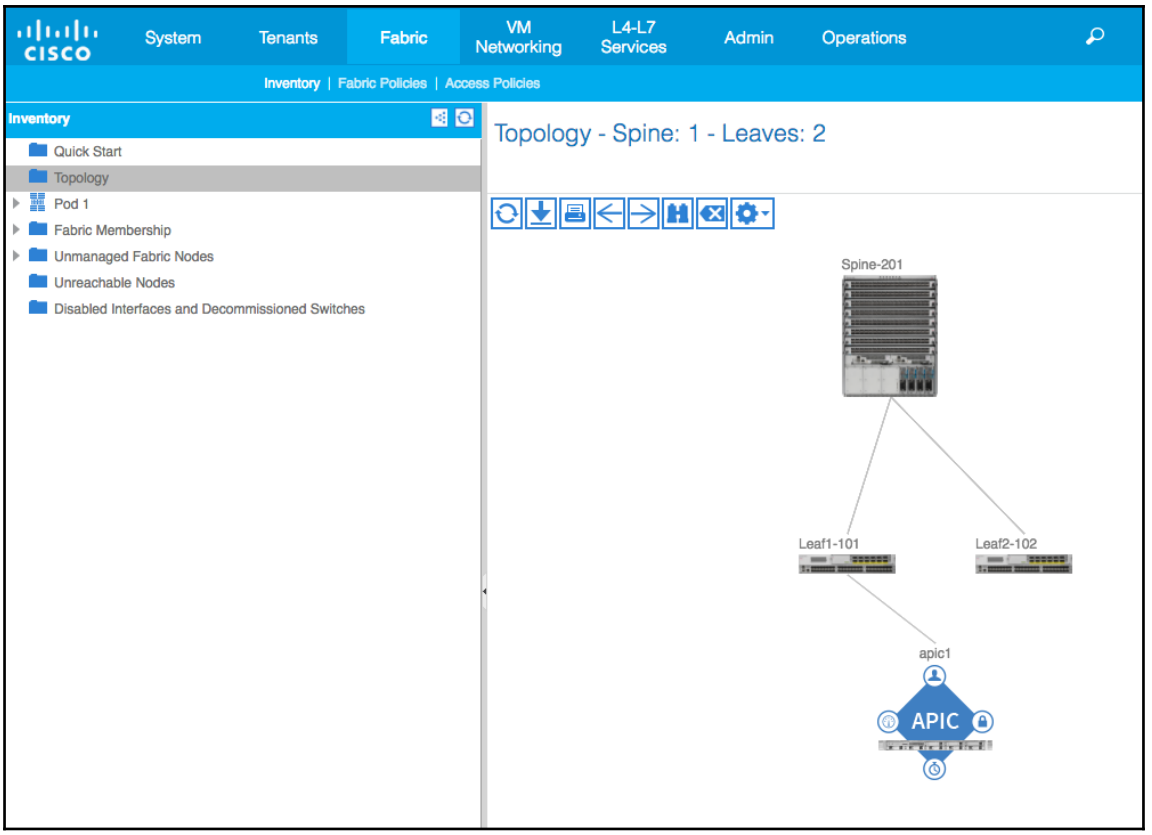


Figure 21

If we expand the Pod out, we can see all of our leaf and spine nodes:

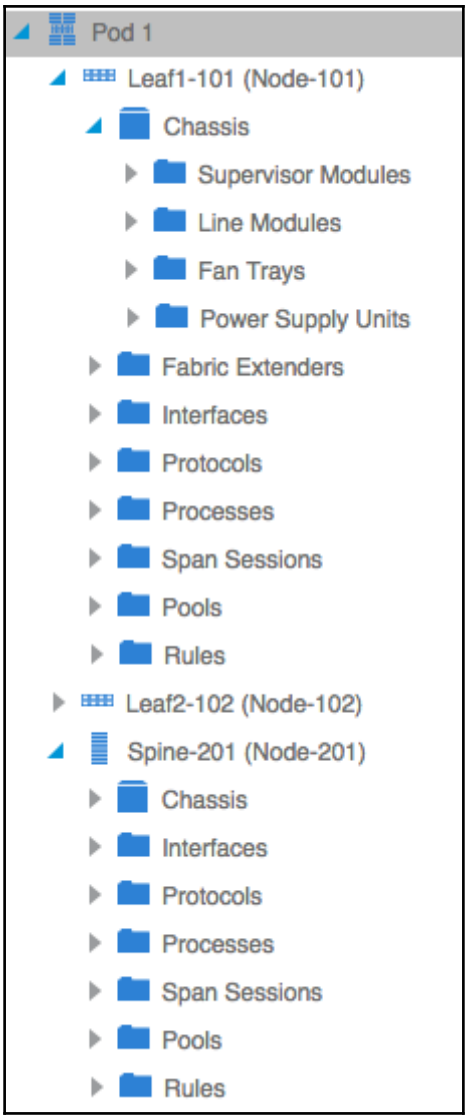


Figure 22

Going through these, we can see our interfaces, routing tables, processes, Pools, and Rules. One thing to note here is that we have many more routing options with a leaf node than we do a spine node:

Figure 23

Under the Fabric Membership option, we have a list of our leaf and spine nodes, which shows us the serial numbers, ID, name, model, role and assigned IP address. It also gives us the certificate information, so we know that SSL is healthy, so IFM can function.

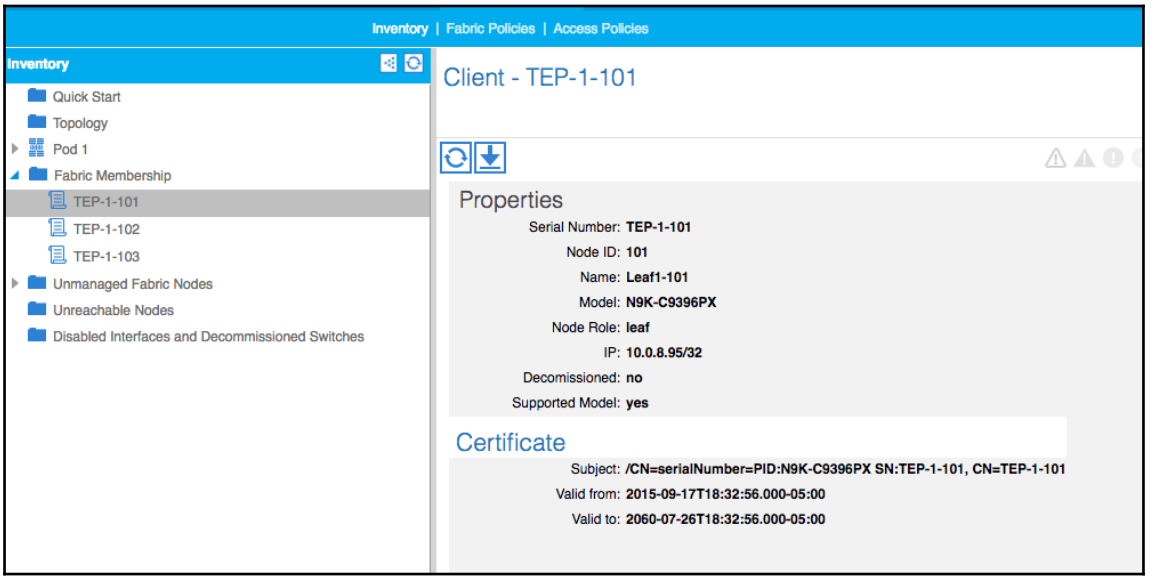


Figure 24

The last three options in this menu are for the Nodes we may be having issues with, whether they currently are unmanaged, unreachable or disabled and decommissioned.

The other options in the Fabric submenu are our policies. Here we can set up Callhome policies, monitoring, and troubleshooting, and spanning tree, VPC policies, whether we want to have CDP turned off or on, and various layer-2 policies. Many of these options have three options, take LLDP for example. We have an option for LLDP-OFF (disabled), LLDP-ON (enabled) and default (Receive State is enabled, and Transmit State is enabled). Similarly, for CDP we have CDP-OFF (disabled), CDP-ON (enabled) and default (where the Admin State is “Disabled”).

VM Networking

Under the VM Networking menu is where we would start to connect ACI into our third-party vendors. The default options are Microsoft, OpenStack, and VMWare.

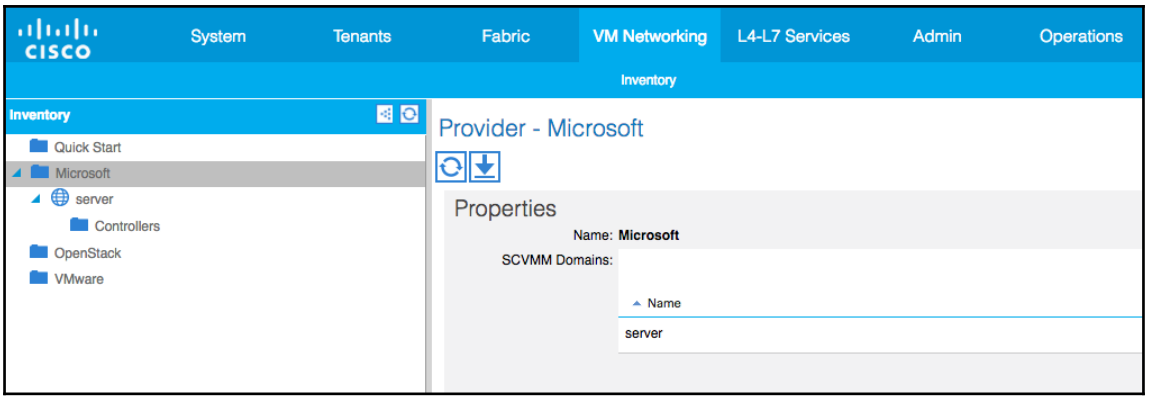


Figure 25

L4-L7 Services

L4-L7 Services allows us to further extend our ACI fabric with additional third-party solutions. This is performed through the addition of packages, which we can import from the Packages submenu, Quick Start option. This is something we will look at in a later chapter.

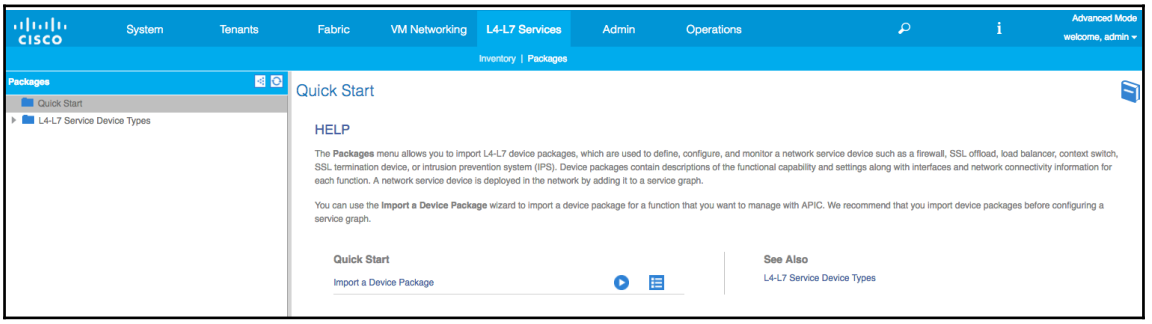


Figure 26

Admin

Under the Admin menu is where we configure **AAA** (**Authentication**, **Authorization**, and **Accounting**). Here we can setup RBAC and also connect to authentication providers, such as LDAP, like Microsoft Active Directory, RADIUS or TACACS+. We can also setup PKI to use certificate chains.

We can create maintenance schedules, either one-off tasks or regular ones. We can configure our log retention policies, upgrade our firmware, and configure Callhome (after setting up the policies in the Fabric menu), SNMP and Syslog. The Admin menu is also where we would perform configuration rollbacks, and the importing of configuration files and exporting of technical support files.

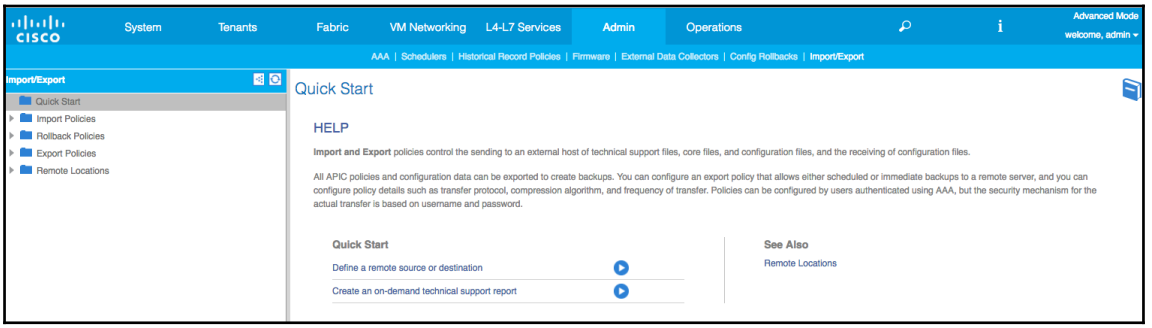


Figure 27

Operations

The final menu option is Operations. This is where will can perform most of our troubleshooting, should the need arise. From here we find endpoints and look at the traffic path, along with any faults along the path. We can perform traceroute as well to check the data plane. This is something we will look at in Chapter 9.

We can also check out usage with the Capacity Dashboard, create Optimizer configuration templates, track our endpoints, and even look at a traffic map.

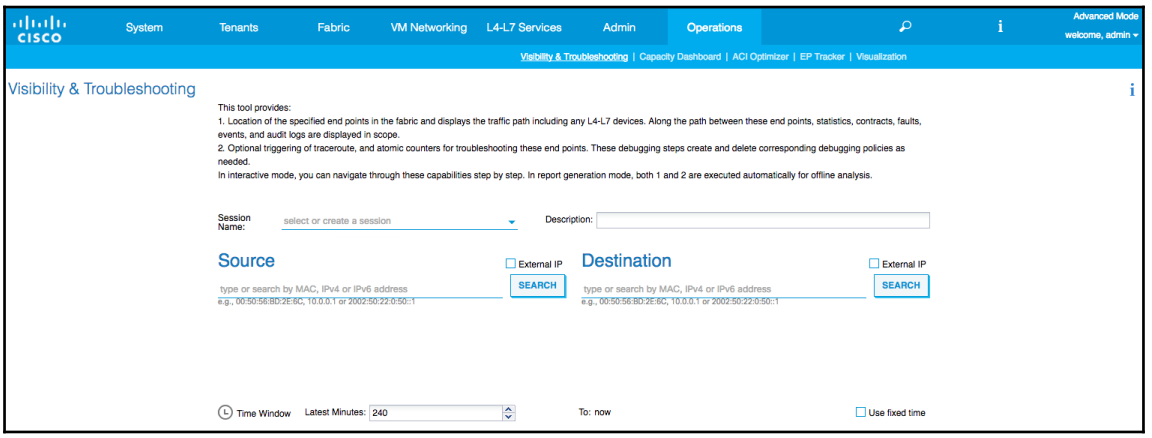


Figure 28

For now, though, it's time to start configuring!

2
Configuring Policies and Tenants

In this chapter, we will cover the following recipes:

- Creating Fabric Policies
- Creating Access Policies
- Creating Tenants
- Configuring Bridge Domains
- Configuring Contexts
- Creating Application Network Profiles
- Creating Endpoint Groups
- Using Contracts between Tenants
- Creating Filters
- Creating Contracts within Tenants
- Creating Management Contracts

Introduction

We will start to configure the ACI fabric, by creating some policies and a couple of tenants.

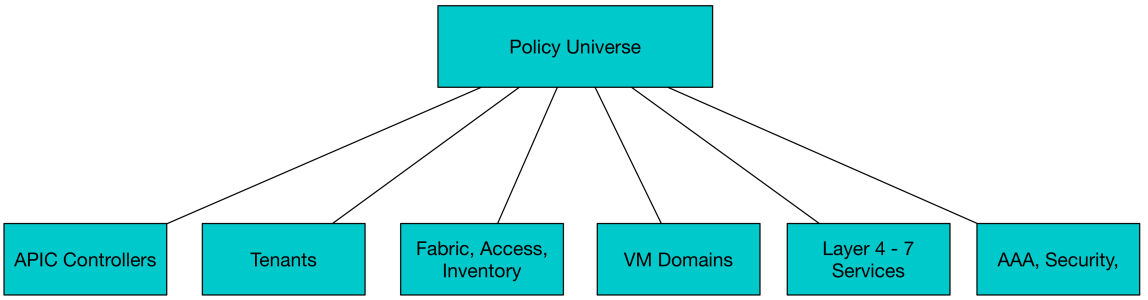
The ACI policy model is all about mapping application requirements to policies. We need Tenant A to talk to an SQL server; we create a policy for that. We also need Tenant A to talk the storage system; we create a policy for that.

The APIC looks after the policies. When we make a change to an object within the fabric, it is the job of the APIC to apply this change to the policy model, which then makes the

change to the affected endpoint. Such an example would be adding a new device to the fabric. Communication with the new device is prohibited until the policy model is updated to include the new device.

There are different policies, but these can be split into fairly distinct groups; ones that govern the ACI fabric as a whole and those that are concerned with tenants.

All the policies are recorded in the MIT, or Management Information Tree.

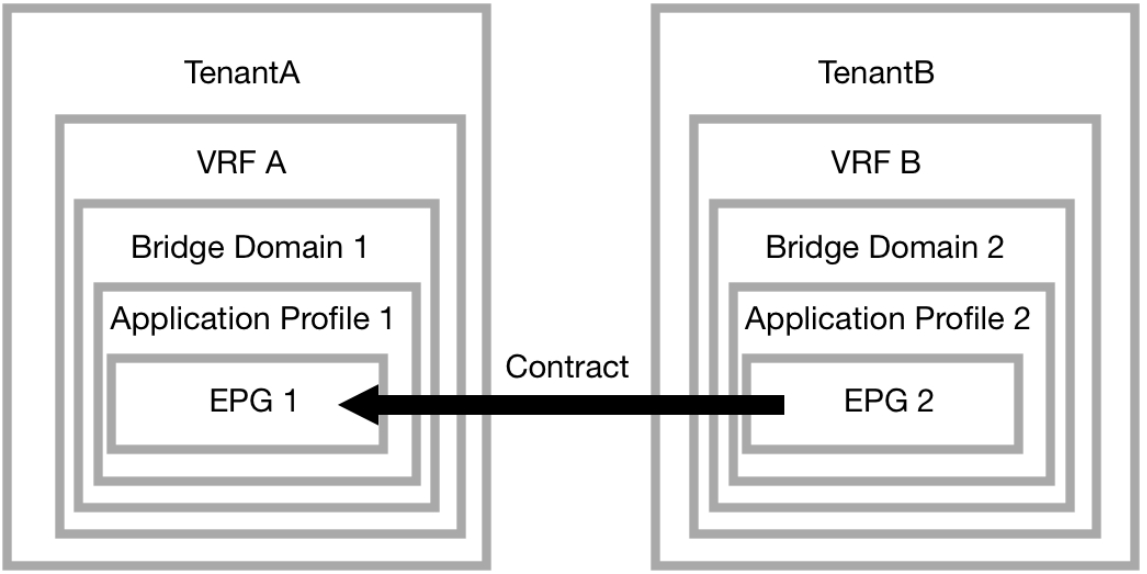


The MIT.

In this chapter, we will start by creating a fabric policy to enable NTP (Network Time Protocol), as this is an essential service for the smooth functioning of the fabric (along with DNS, which is covered in chapter 4). We will look at access policies and enable CDP (Cisco Discovery Protocol) across the fabric.

We will then create our first tenant, and set it up for networking by creating the networking and application components and then we will give it something to do, by creating a contract which we will provide to a second tenant to consume.

This is a basic idea of what we will be configuring:

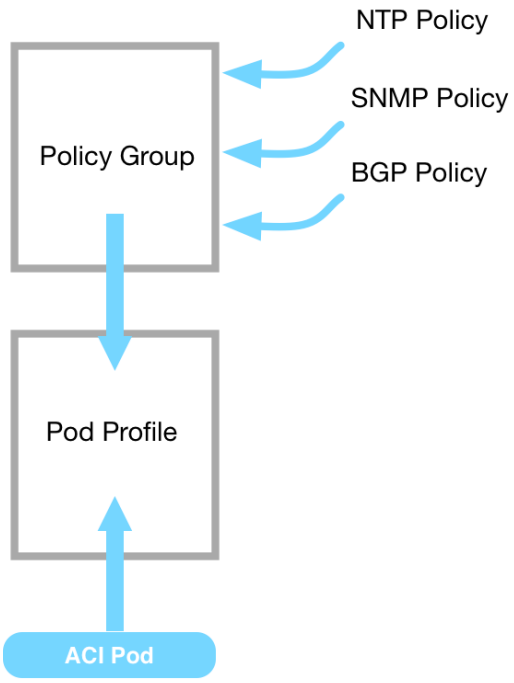


We will also look at creating a management contract for permitting SNMP traffic, which we will be needing for Chapter 8.

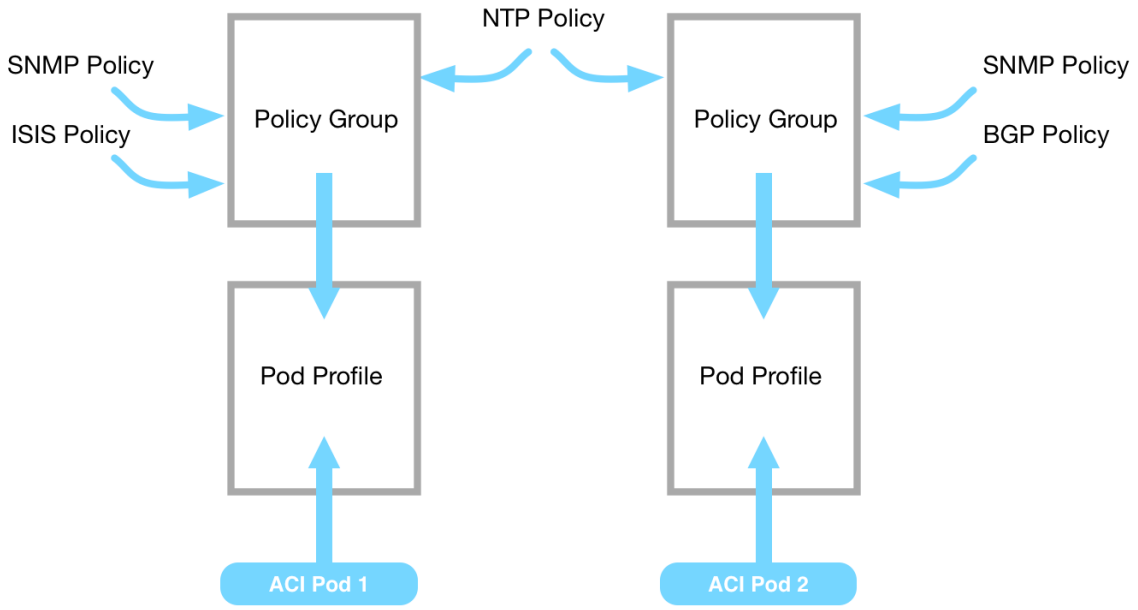
Creating fabric policies

In this recipe, we will create an NTP policy, and assign it to our POD. NTP is a good place to start, as having a common and synched time source is critical for third-party authentication, such as LDAP and logging.

In this recipe, we will use the Quick Start menu to create an NTP policy, in which we will define our NTP servers. We will then create a POD policy and attach our NTP policy to it. Lastly, we create a POD profile, which calls the policy and applies it to our pod (our fabric).



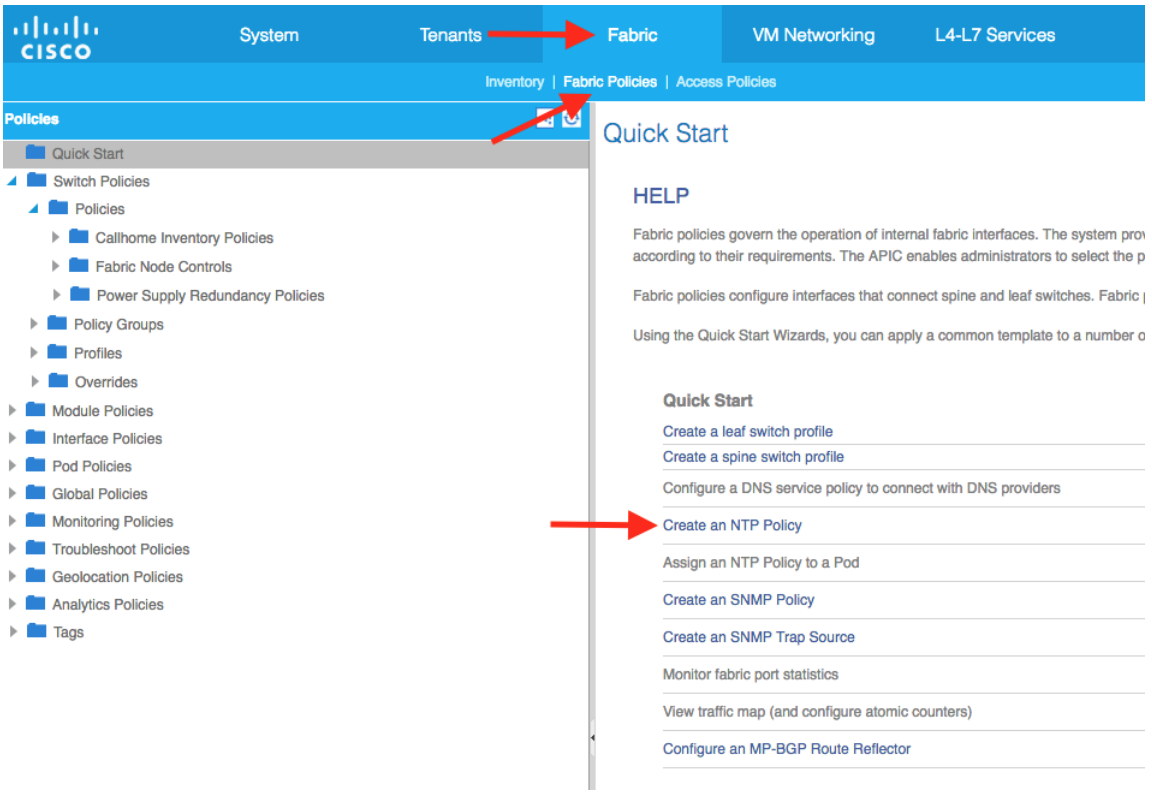
We can assign pods to different profiles, and we can share policies between policy groups. So, we may have one NTP policy, but different SNMP policies for different pods.



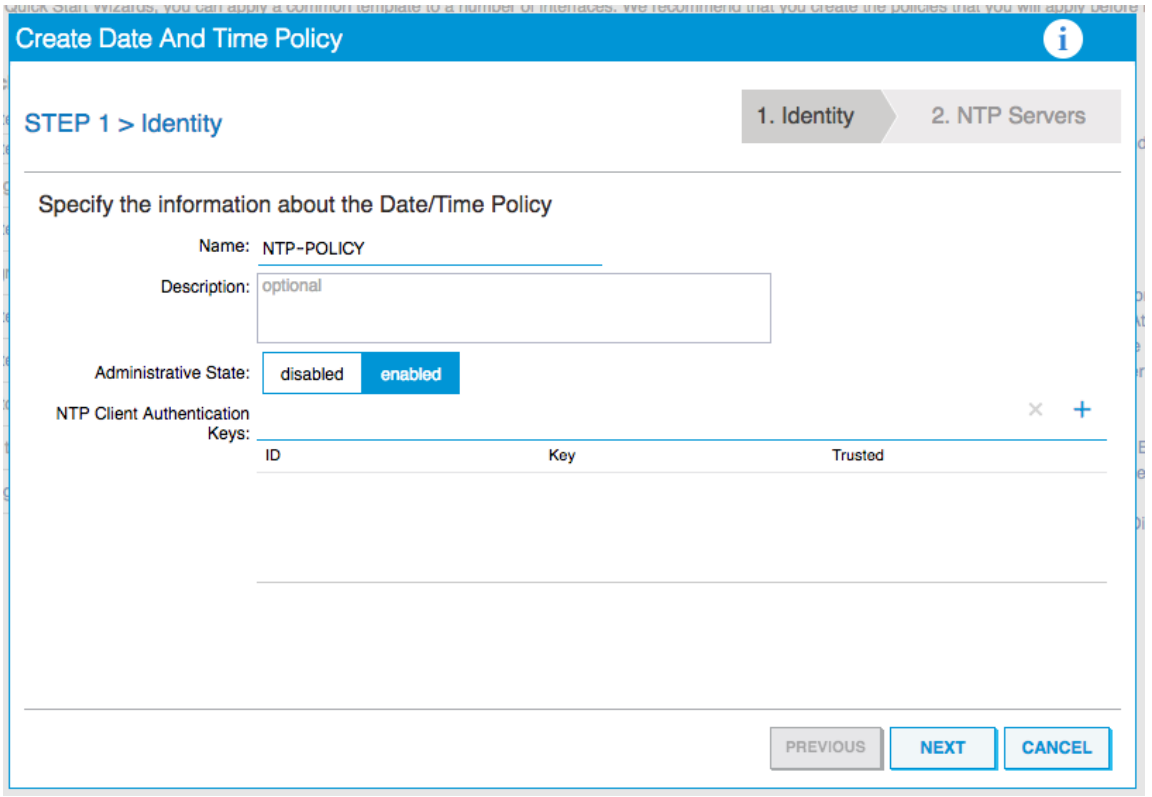
The ACI fabric is very flexible in this respect.

How to do it…

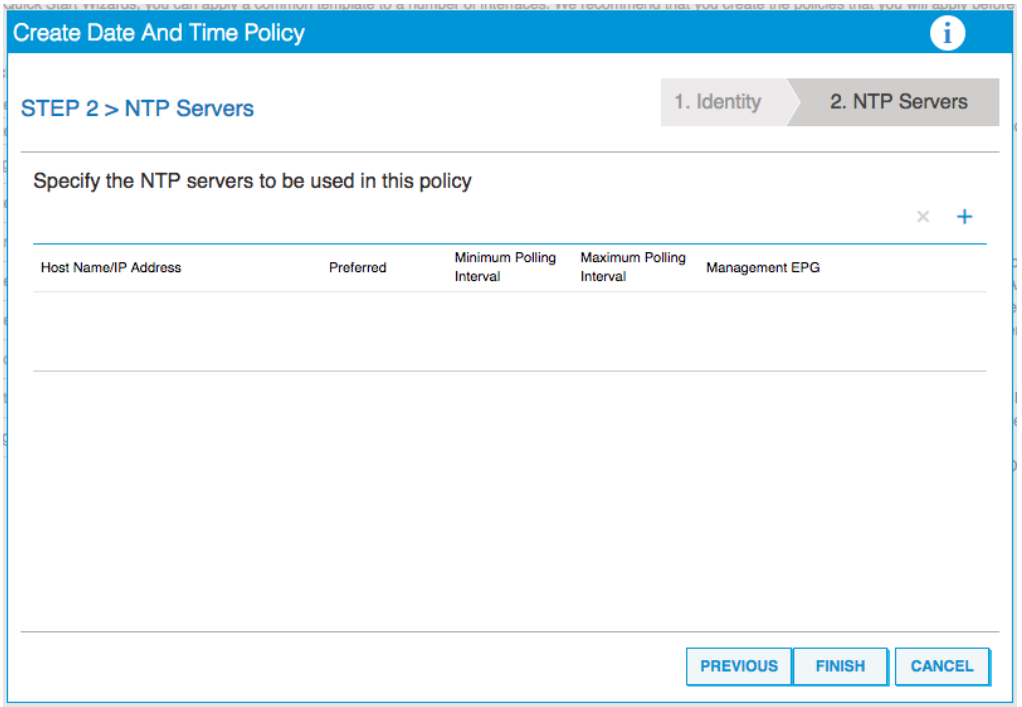
1. From the Fabric menu, select **Fabric Policie**s. From the **Quick Start** menu, select **Create an NTP Policy**:



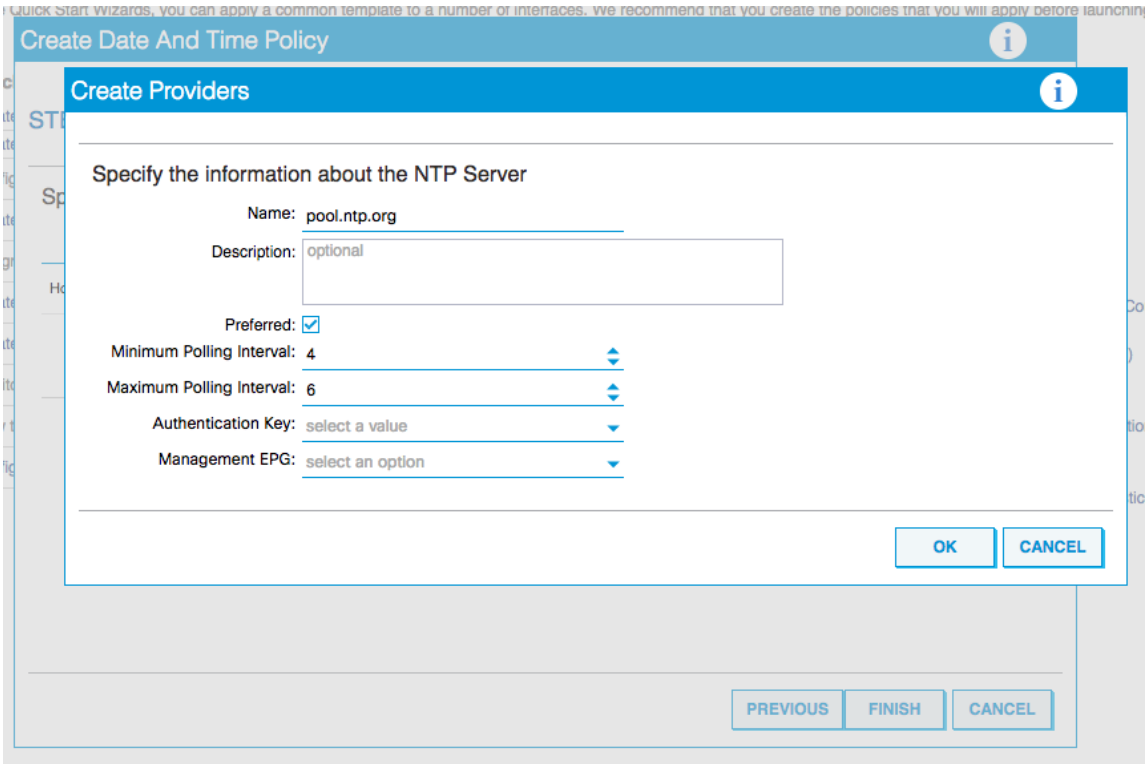
2. A new window will pop up, and here we give our new policy a name, a description (which is optional) and enable it. We can also define any authentication keys, should the servers use them. Clicking on **Next** takes us to the next page, where we specify our NTP servers.



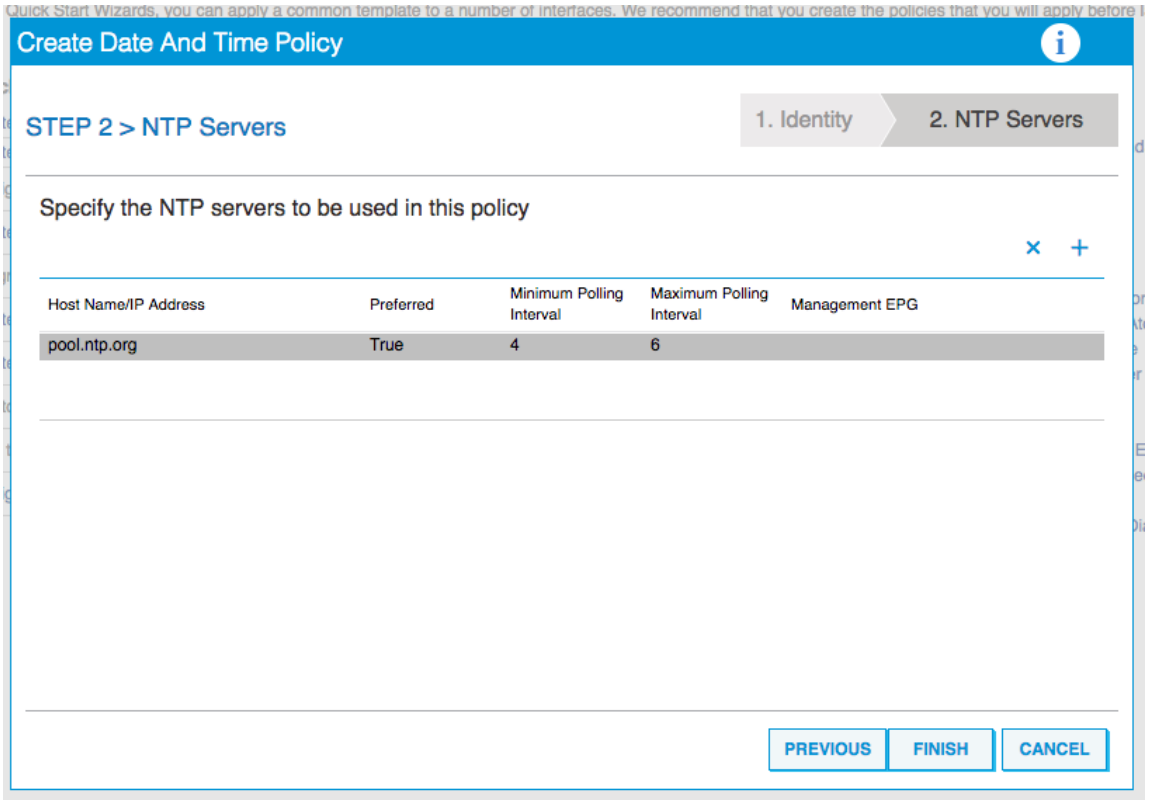
3. We click on the plus sign on the right-hand side, and enter the IP address or **Fully Qualified Domain Name** (**FQDN**) of the NTP server(s):



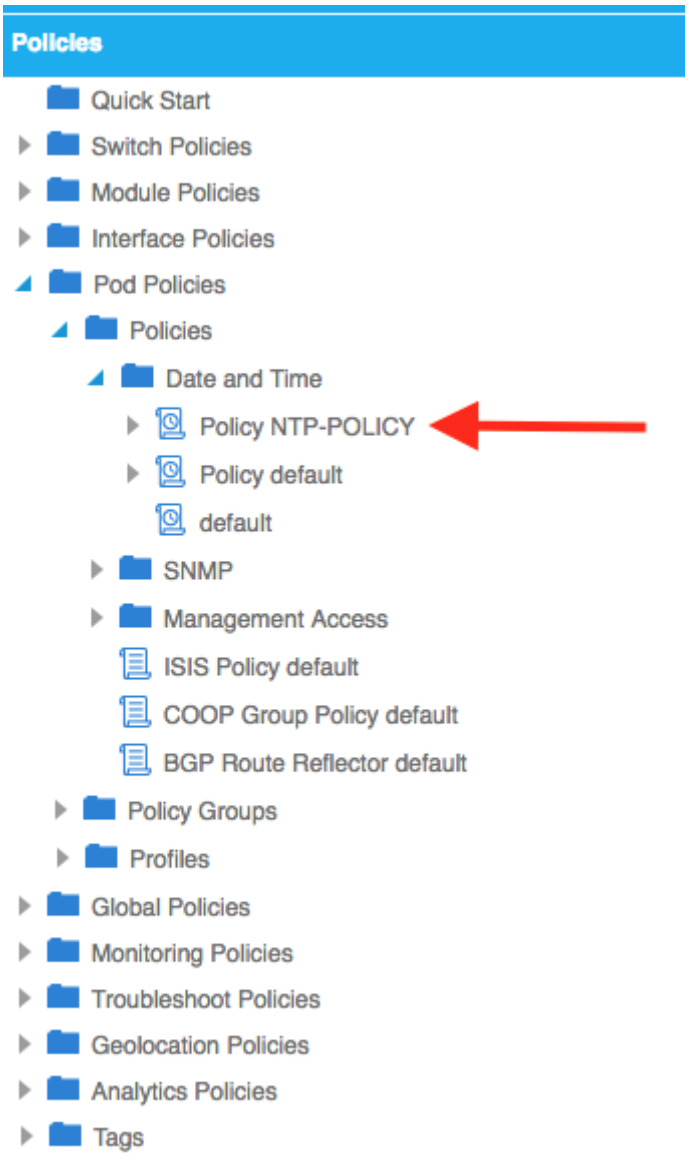
4. We can also select a management EPG, which is useful if the NTP servers are outside of our network. Then we click on **OK**.



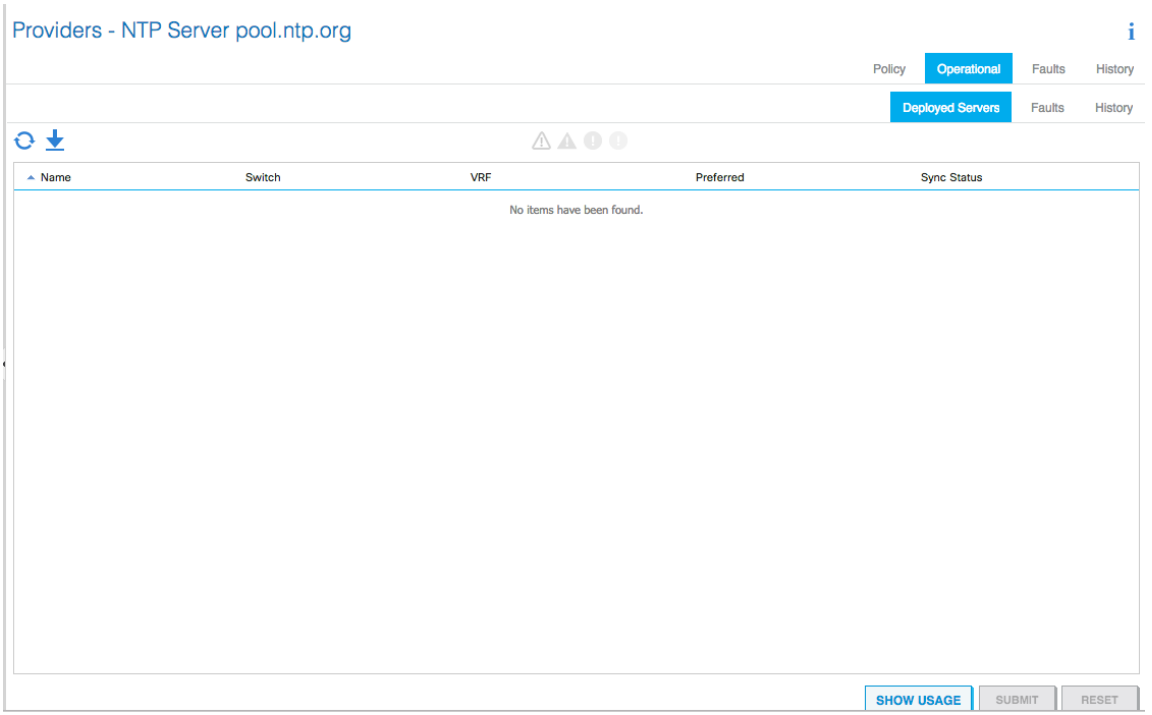
5. Then click on **Finish**.



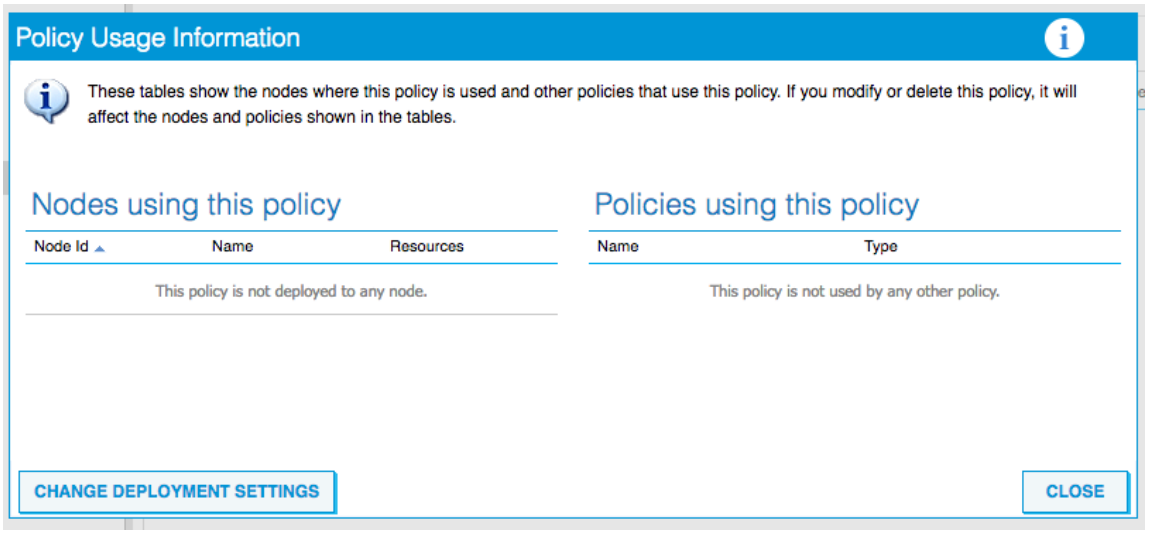
We can now see our custom policy under POD policies:



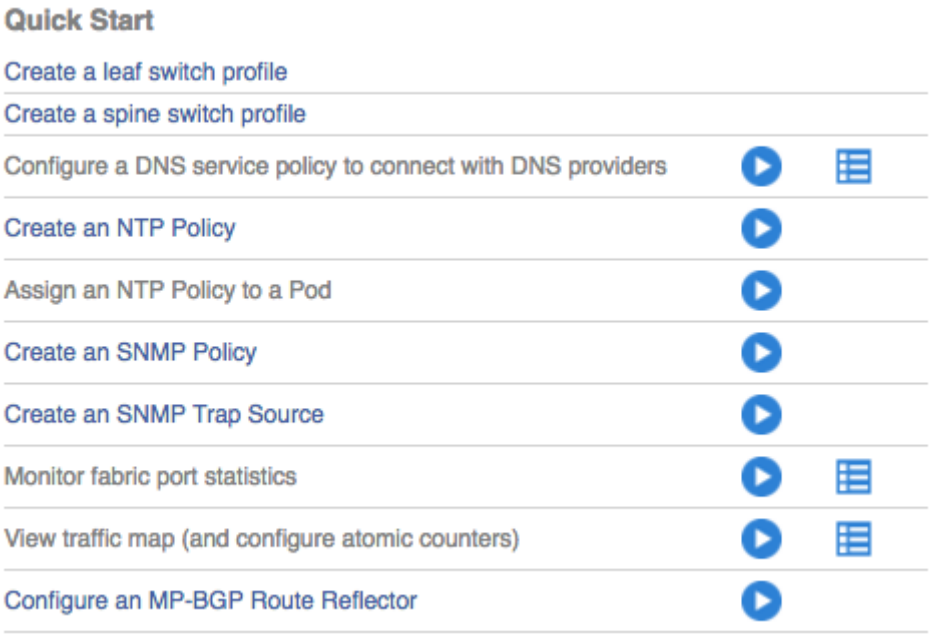
At the moment, though, we are not using it:



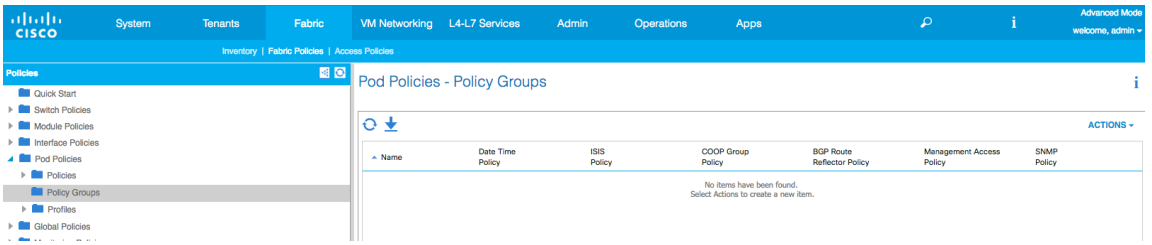
6. Clicking on **Show Usage** at the bottom of the screen shows that no nodes or policies are using the policy.



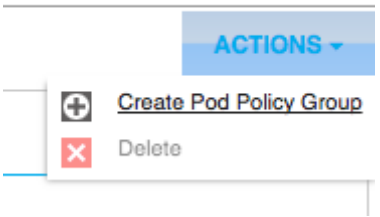
7. To use the policy, we must assign it to a POD, as we can see from the Quick Start menu:



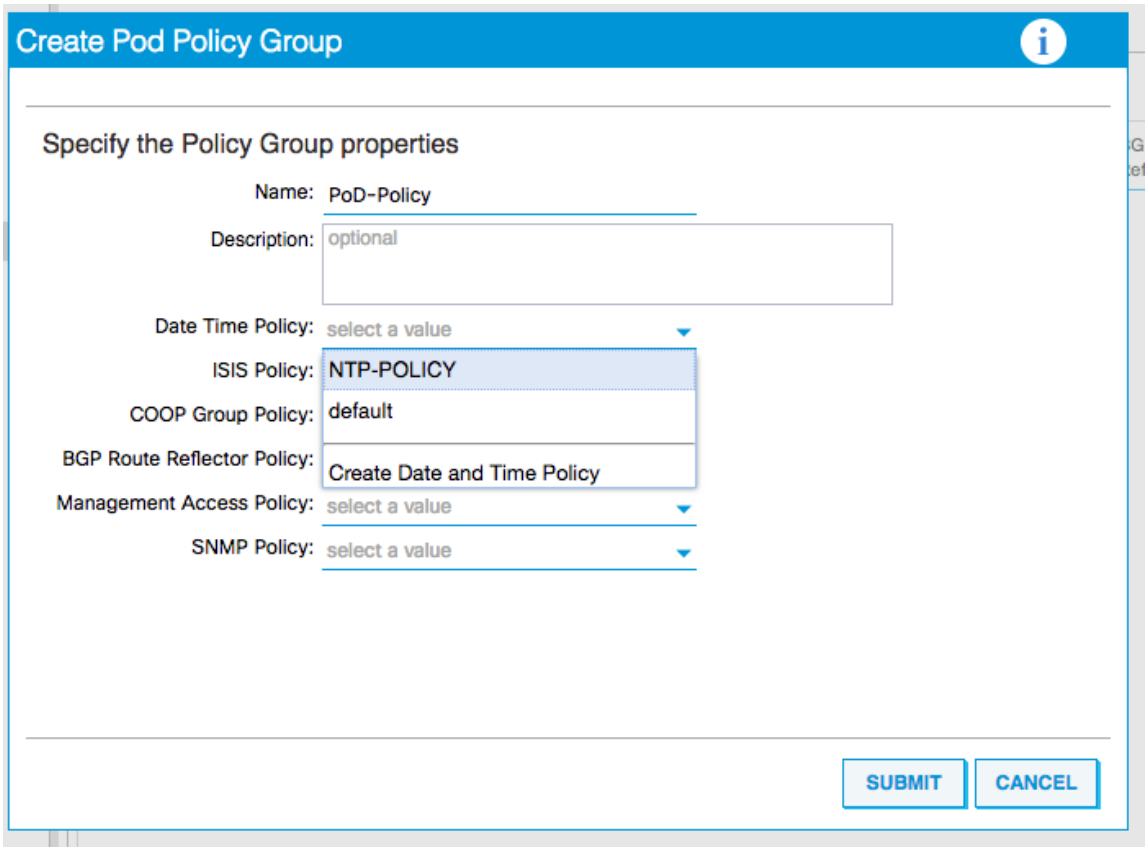
8. Clicking on the arrow in the circle will show us a handy video on how to do this. We need to go into the policy groups, under POD policies and create a new policy:



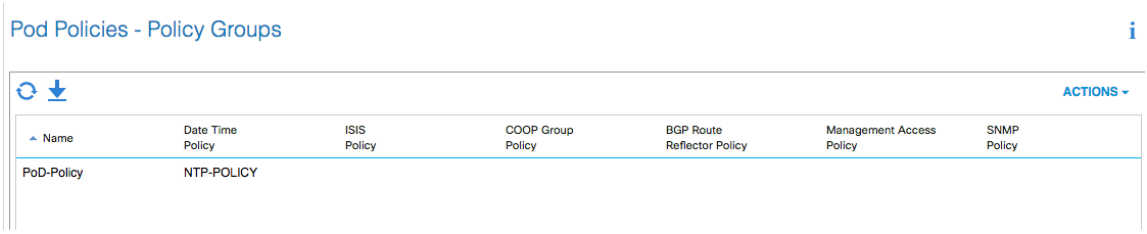
9. To create the policy, click on the **Actions** menu, and select **Create POD Policy Group**



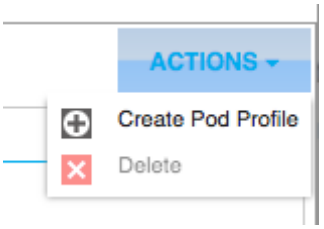
10. From here we can attach our NTP-POLICY to the PoD-Policy. To attach the policy, click on the drop-down next to "Date Time Policy" and select NTP-POLICY from the list of options:



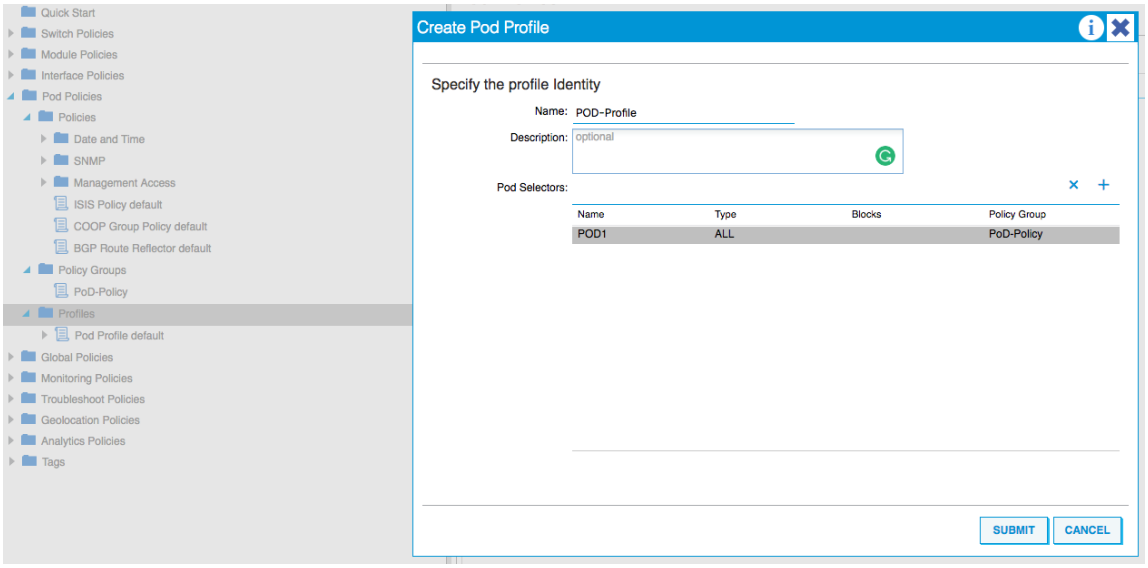
We can see our new policy.



11. We have not finished yet, as we still need to create a POD profile and assign the policy to the profile. The process is similar to before, we go to **Profiles** (under the Pod Policies menu), and select **Actions**, then **Create Pod Profile**



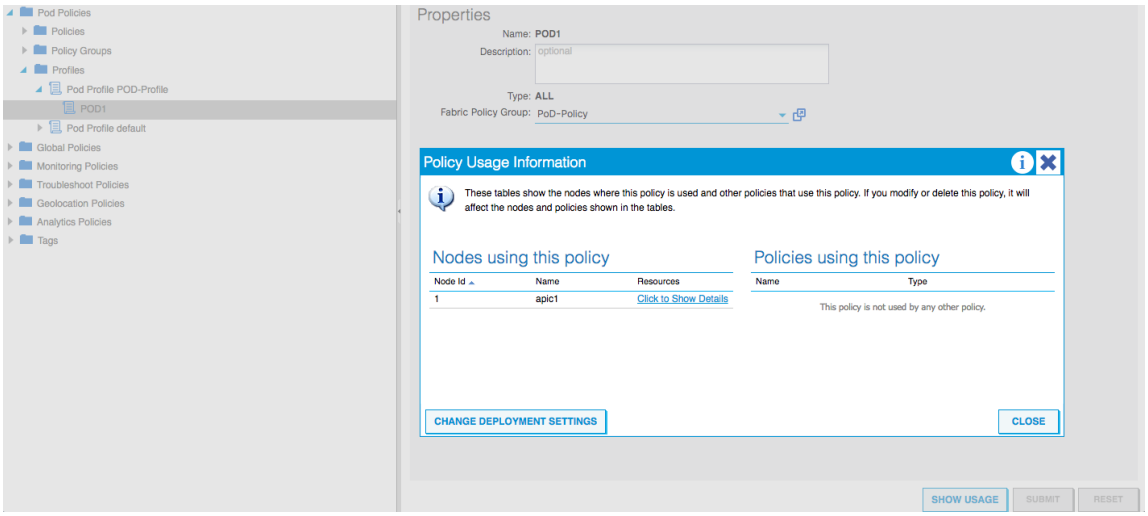
12. We give it a name, and associate our policy to it.



How it works…

Once we create a policy, we must associate it with a Pod policy. The Pod policy must then be associated with a Pod Profile.

We can see the results below:



Our APIC is set to use the new profile, which will be pushed down to the spine and leaf nodes.

We can also check the NTP status from the APIC CLI, using the command "**show ntp**" (you may want to add NTP servers using the IP address until the DNS recipe from Chapter 4 is completed).

```
apic1# show ntp
 nodeid      remote           refid      st     t   when   poll    reach    delay
offset    jitter
 --------  -  ---------------  --------  -----  --  ------  ------  -------  -------
--------  --------
 1           216.239.35.4     .INIT.    16     u   -       16      0        0.000
0.000     0.000
apic1#
```

Creating Access policies

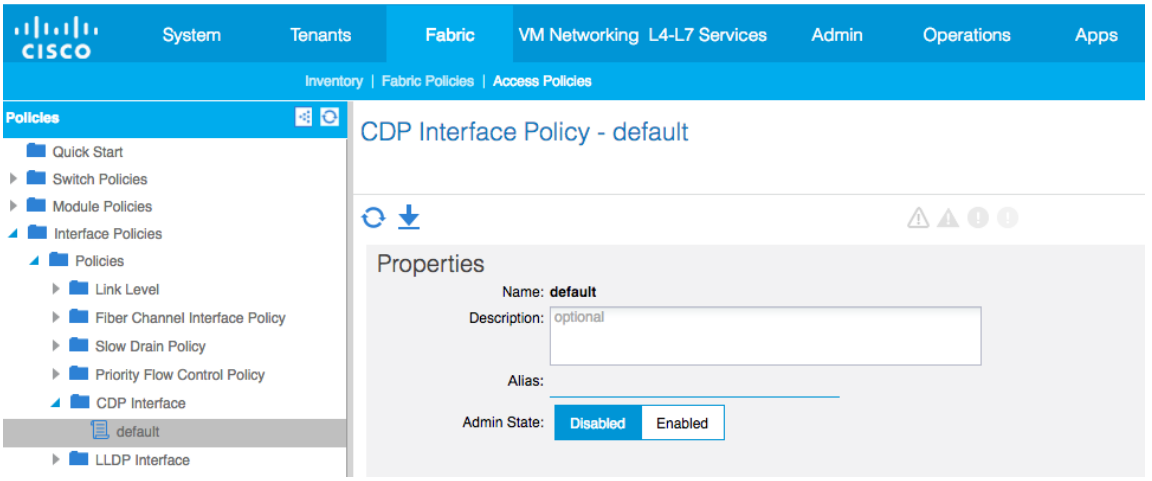
Access policies control the operation of switch ports, allowing for connectivity to resources

such as storage and compute, hypervisors, layer 4 to layer 7 devices, and protocols such as CDP, LLDP, and STP.

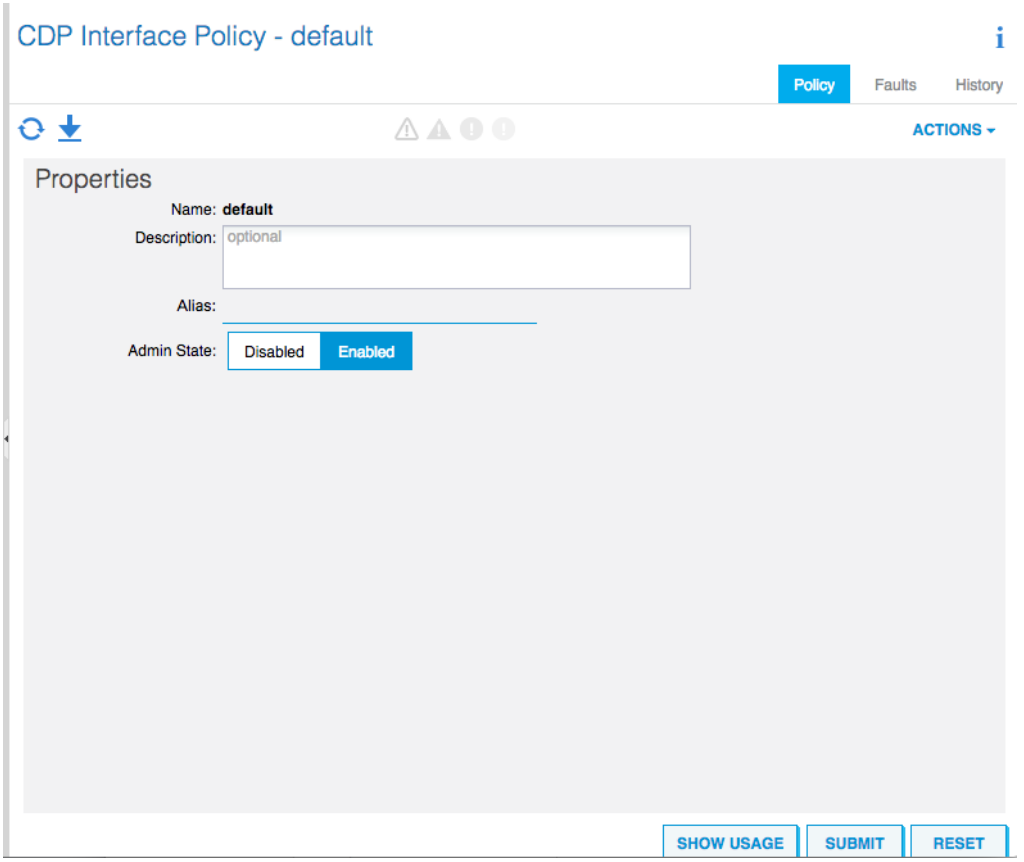
In this recipe, we are going to look at access policies and enable a pre-configured policy. We wil then look at how to override this policy on a per-port basic, and also to override blocks of ports on a leaf.

How to do it…

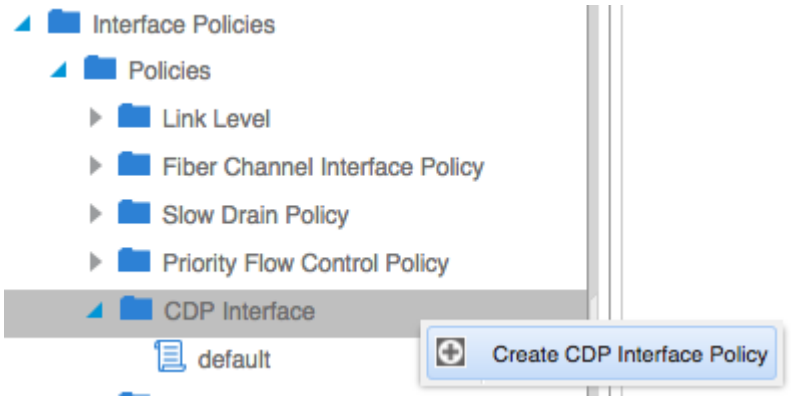
1. From the Fabric menu, select **Access Policies**. Expand out Interface Policies, Policies and then CDP Interface. We can see that there is already a **default** policy:



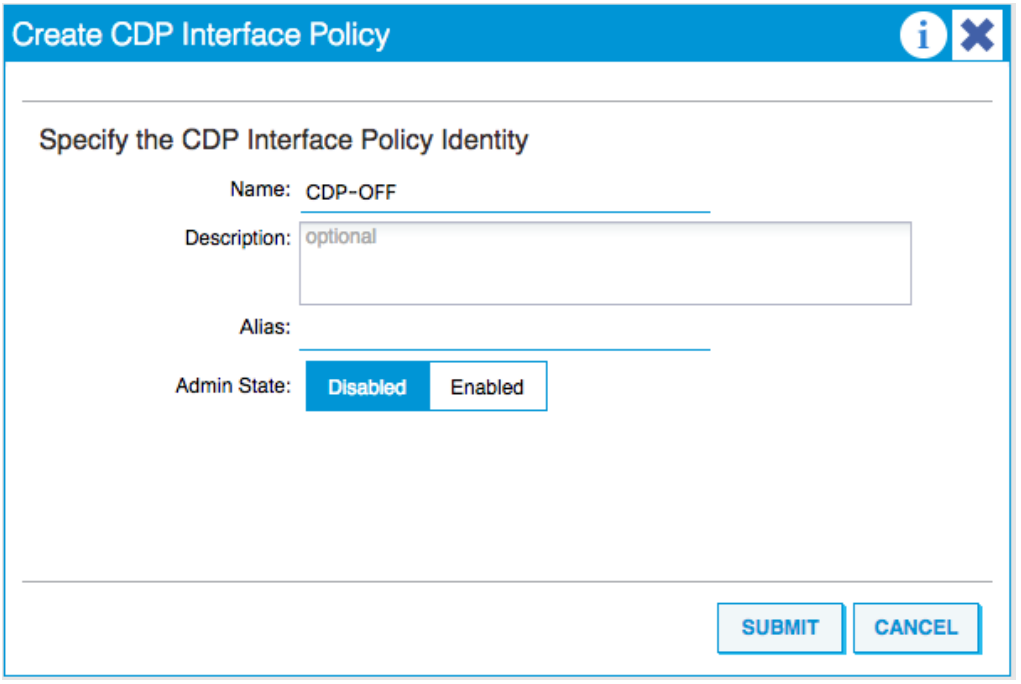
2. The default is for CDP to be disabled. So switch the Admin State to enabled, and click on **SUBMIT** in the bottom corner of the window:



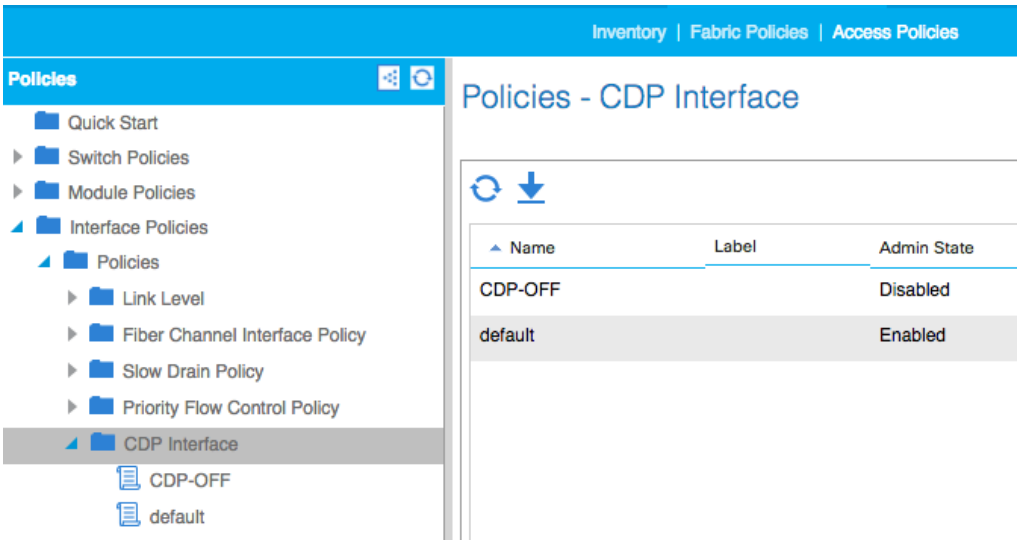
3. This has enabled CDP globally, but what if we need to be a little more selective and disable on a single port?
4. Right-click on CDP Interface and select “Create CDP Interface Policy”.



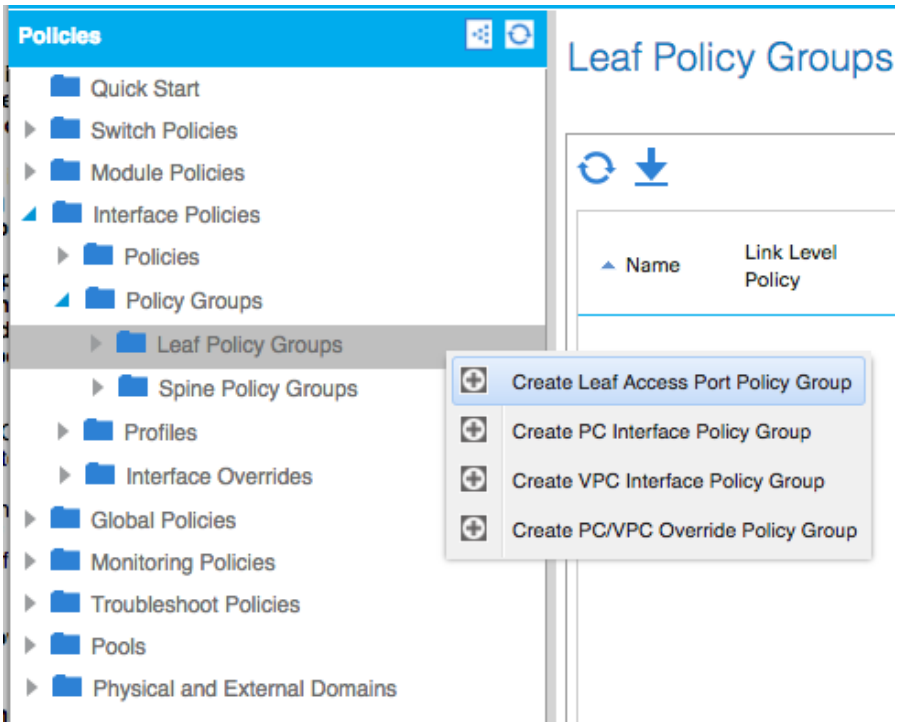
5. Name the new policy "CDP-OFF" and set the state to disabled.



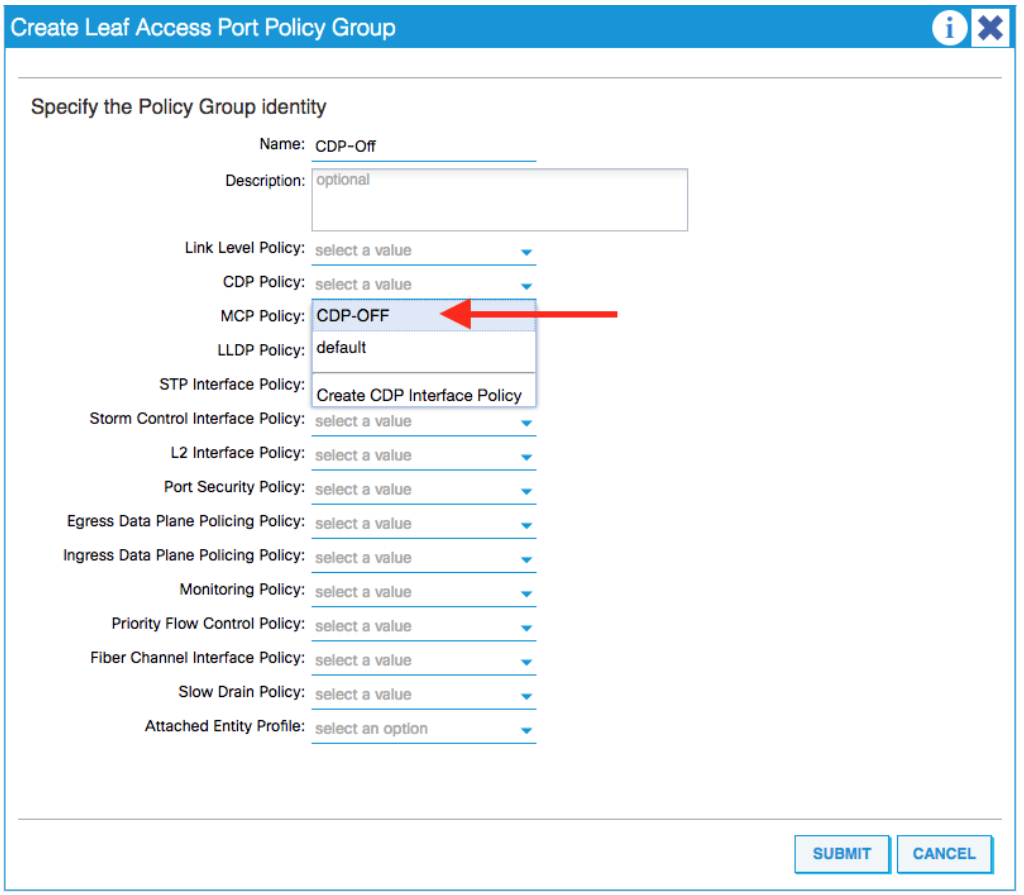
6. Click Submit.
7. We now have two policies, one with CDP enabled, the other has CDP disabled.



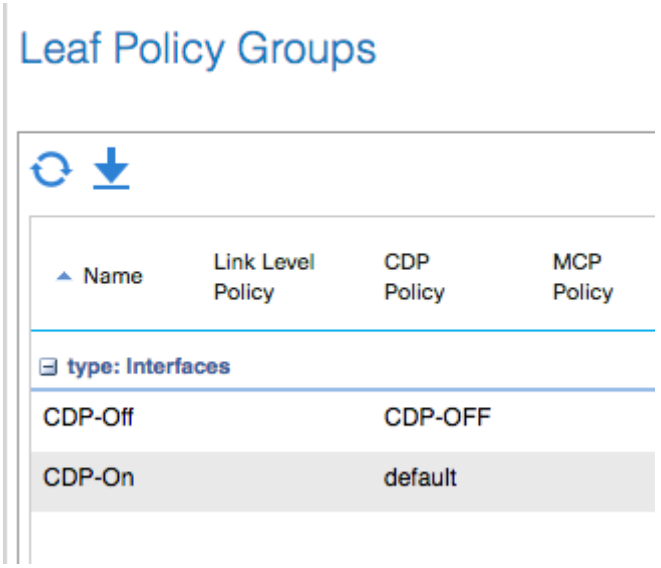
8. We can now create a Leaf Policy Group (well, we should create two, actually). Right-click on Leaf-Policy Groups and select “Create Leaf Access Port Policy Group”.



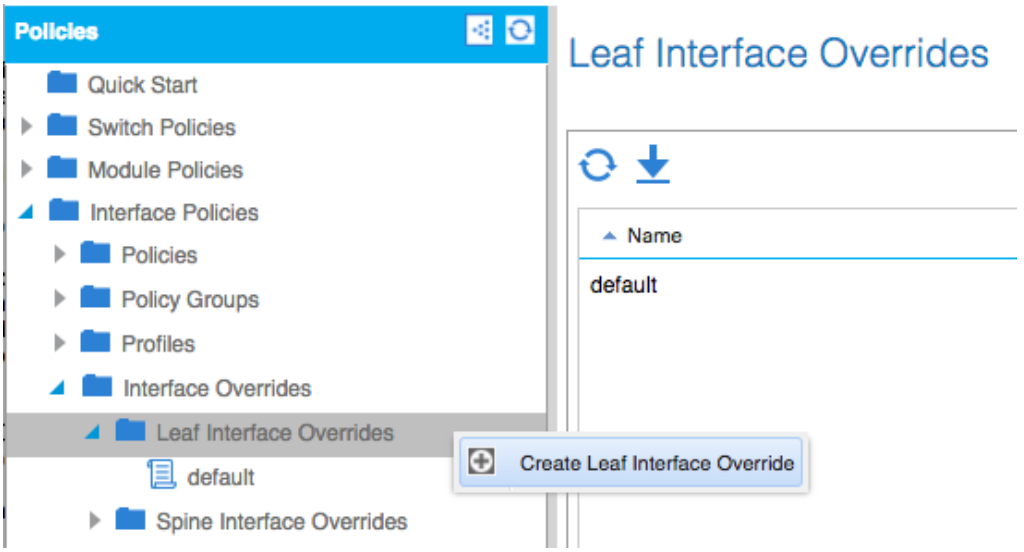
9. Name the new policy “CDP-Off” and choose the CDP-OFF policy from the drop-down menu next to CDP Policy.



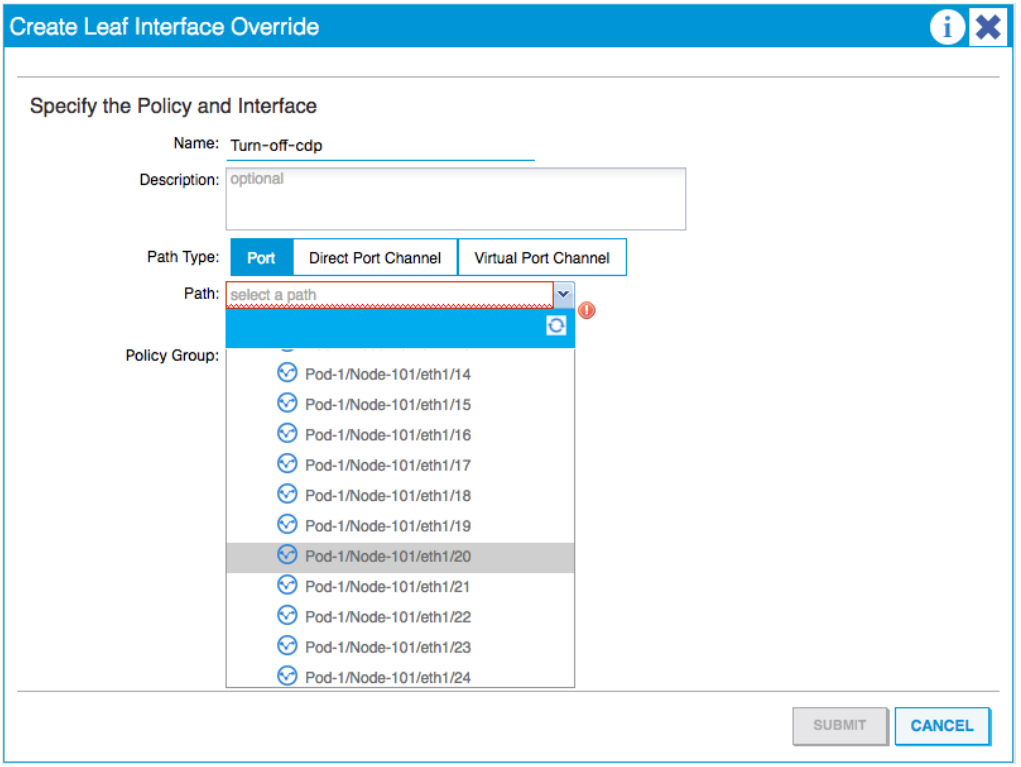
10. Click Submit.
11. Repeat the process to create a second Leaf Access Port Policy Group, this time selecting default from the CDP Policy drop-down. The results should look like this:



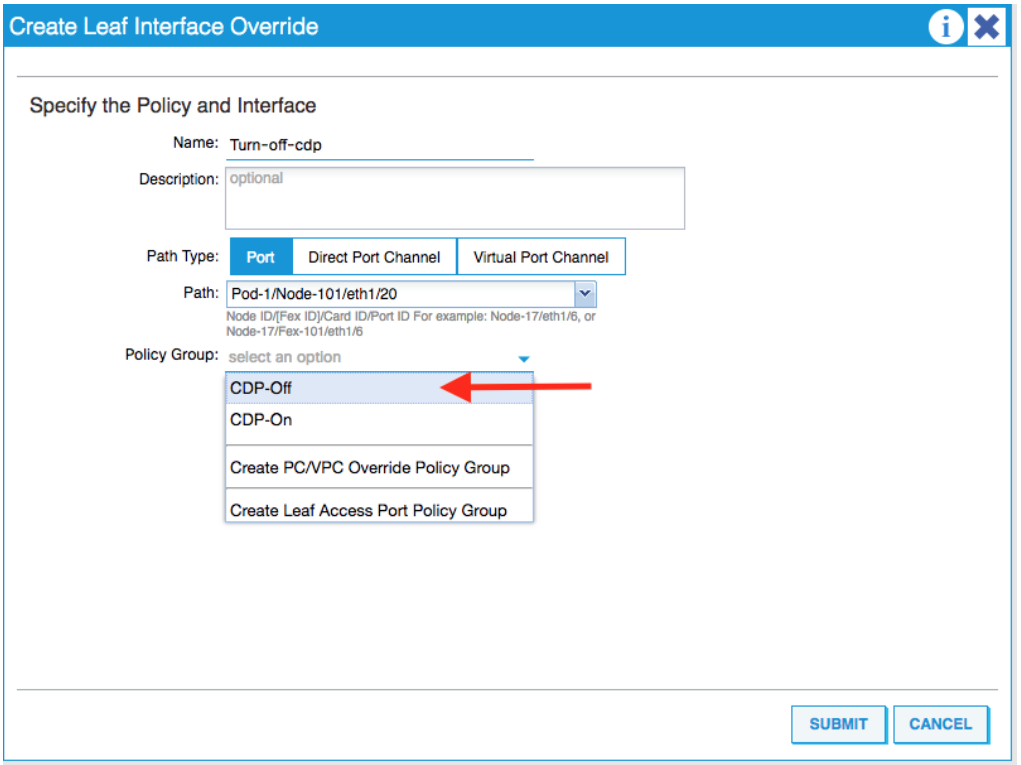
12. Navigate to Interface Policies > Interface Overrides > Leaf Interface Overrides and select “Create Leaf Interface Override”.



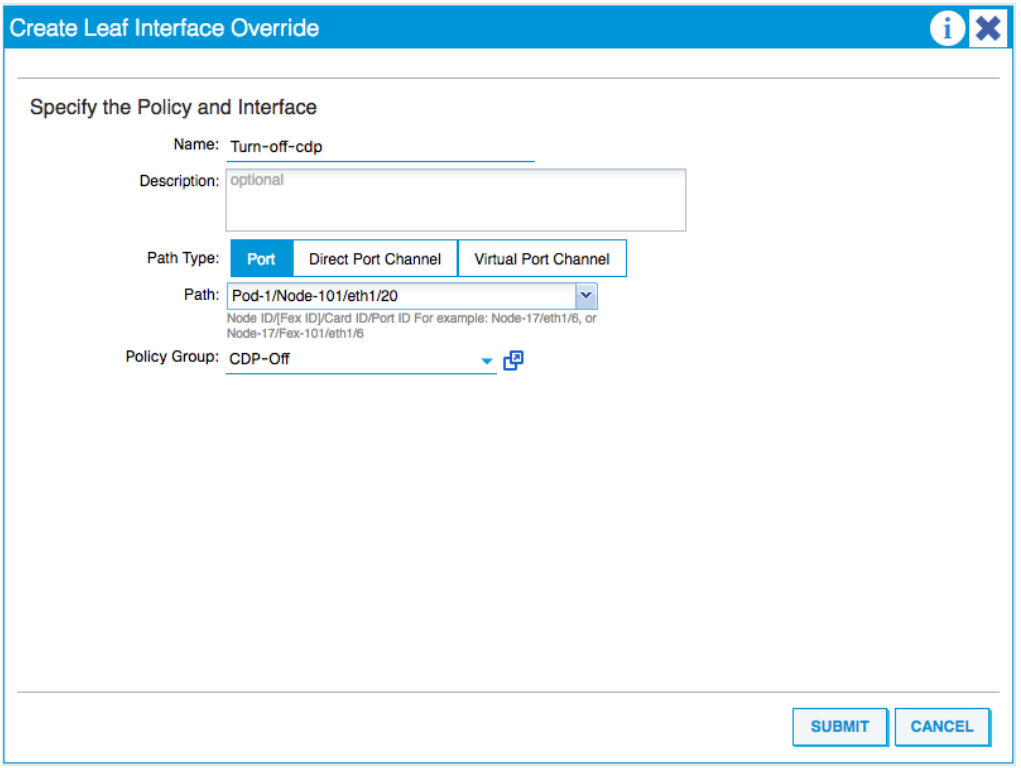
13. Enter a name for the override and select the port from the drop-down list.



14. From the Policy Group drop-down, select the CDP-Off policy created earlier.



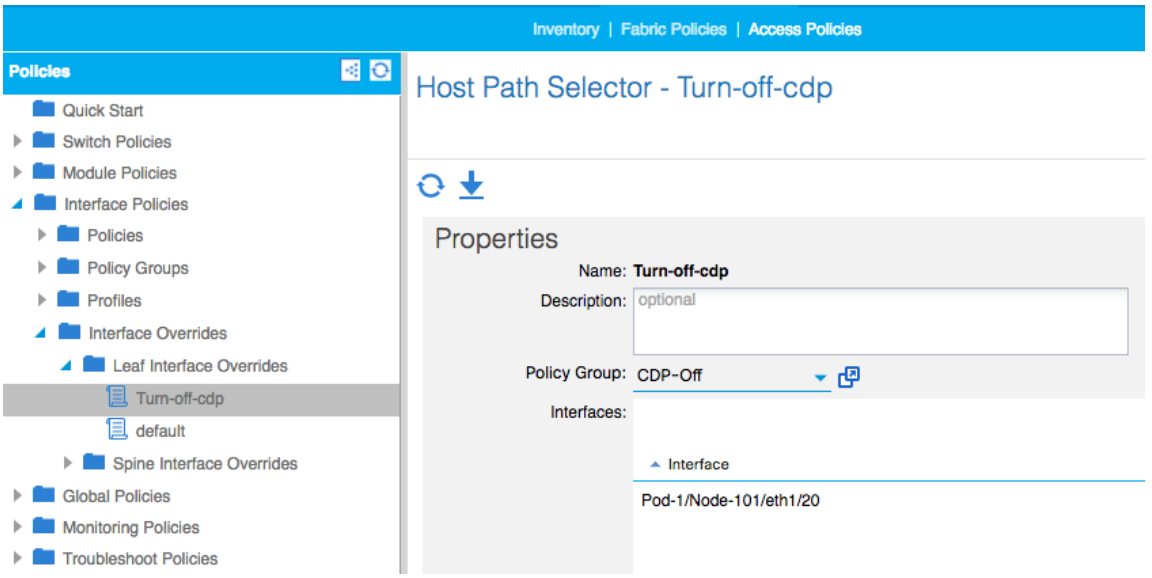
15. Click Submit.



How it works

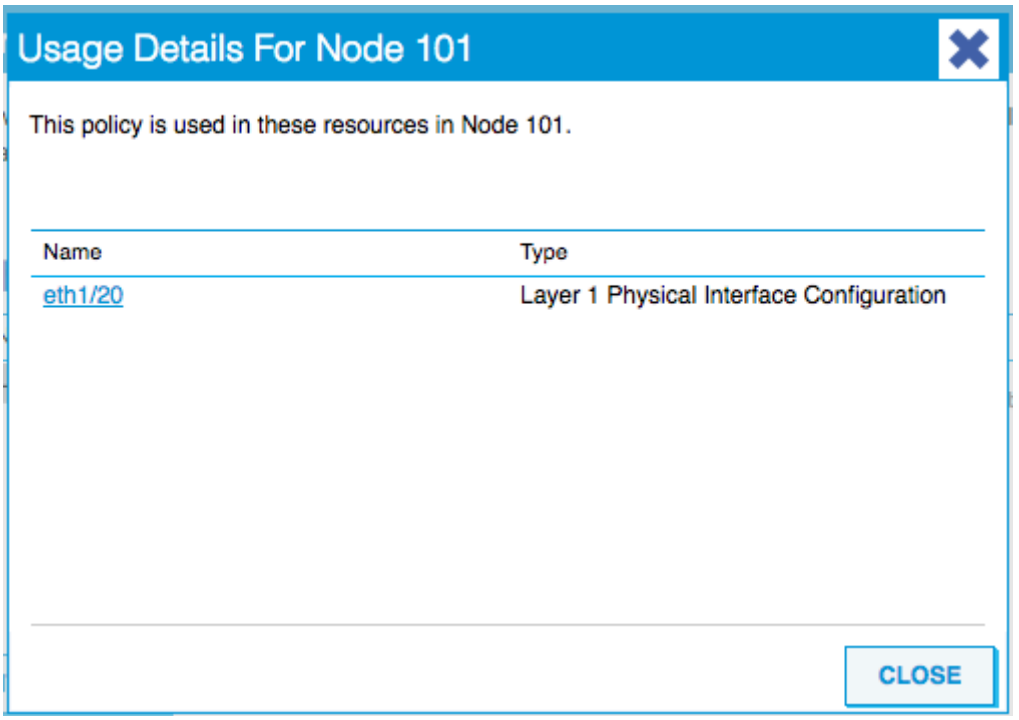
There are many pre-configured policies, which will cover the majority of the day-to-day configuration changes you will need to support your infrastructure. Many of these will already be enabled, or disabled, allowing you to tweak them as needed. Not all the defaults will suit every environment. Taking CDP as the example, we may need to turn it off on a single port, as we did above by creating an override.

We can check that the override is working as expected, by looking at the policy.



If we click on "Show Usage" down in the bottom right-hand corner, we can see that the policy is being used by Leaf-1

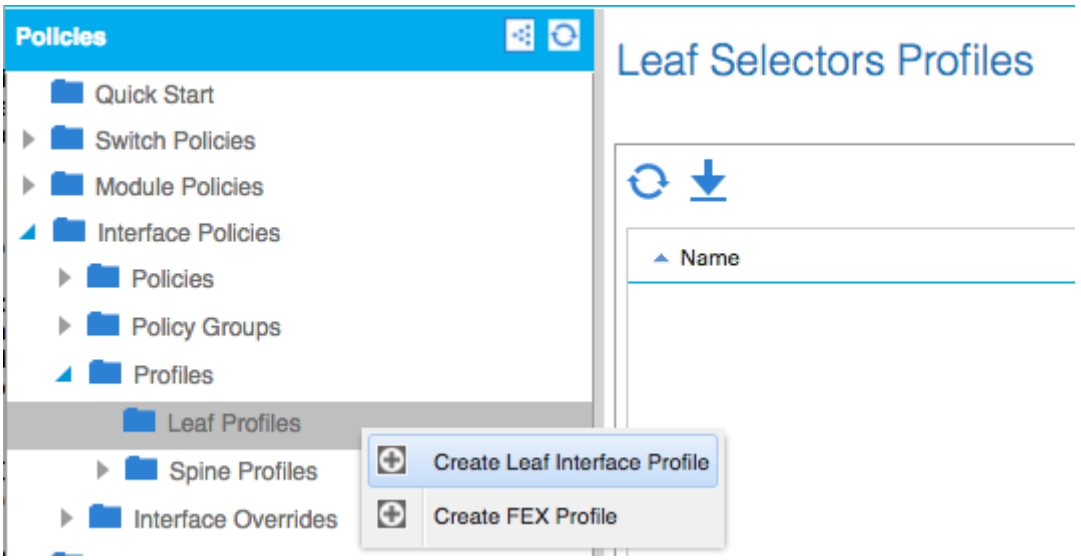
Clicking on "Click to Show Details" will show is the port that is the object of the override.



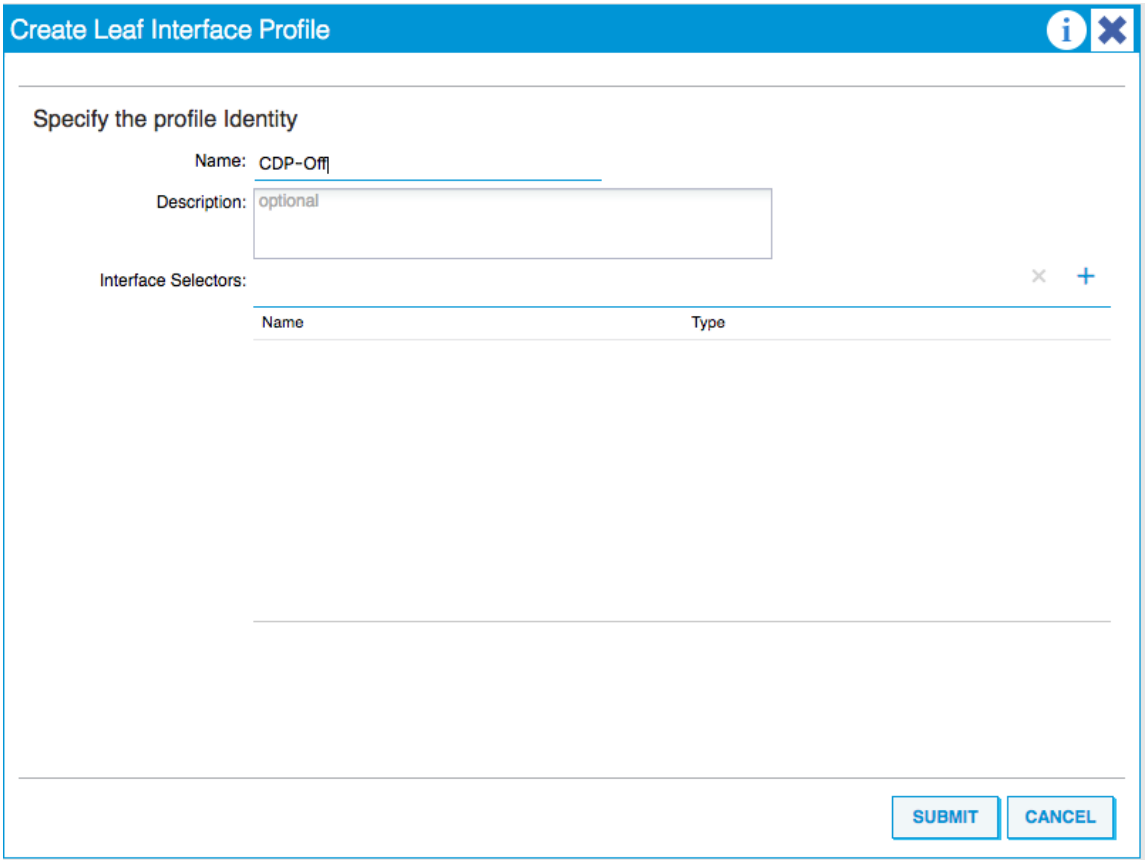
There's more…

The solution above, would not ba appropriate, though, if we wanted to turn CDP off for a range of ports.

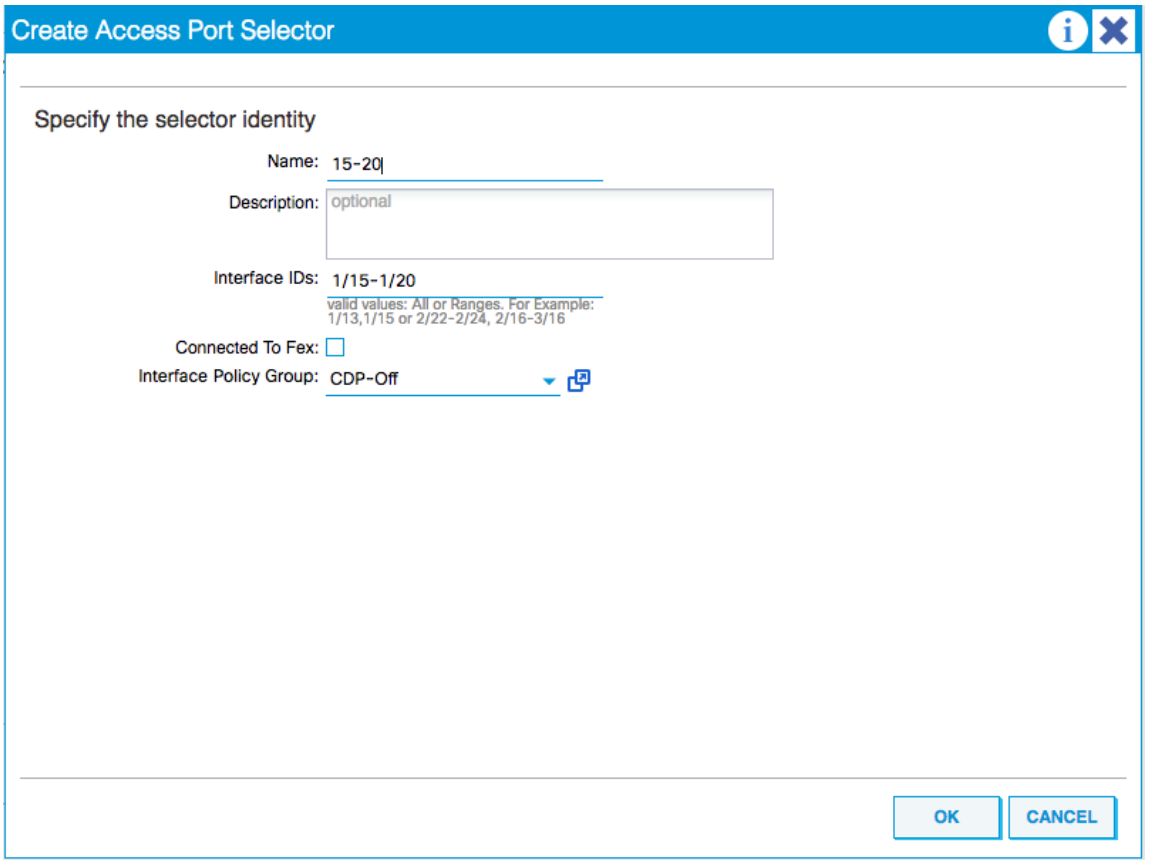
To turn off CDP for a range of ports, we would need to create a Leaf Interface Profile. To do this we navigate to Interface Policies > Profiles > Leaf Profiles and right-click on it and select "Create Leaf Interface Profile".



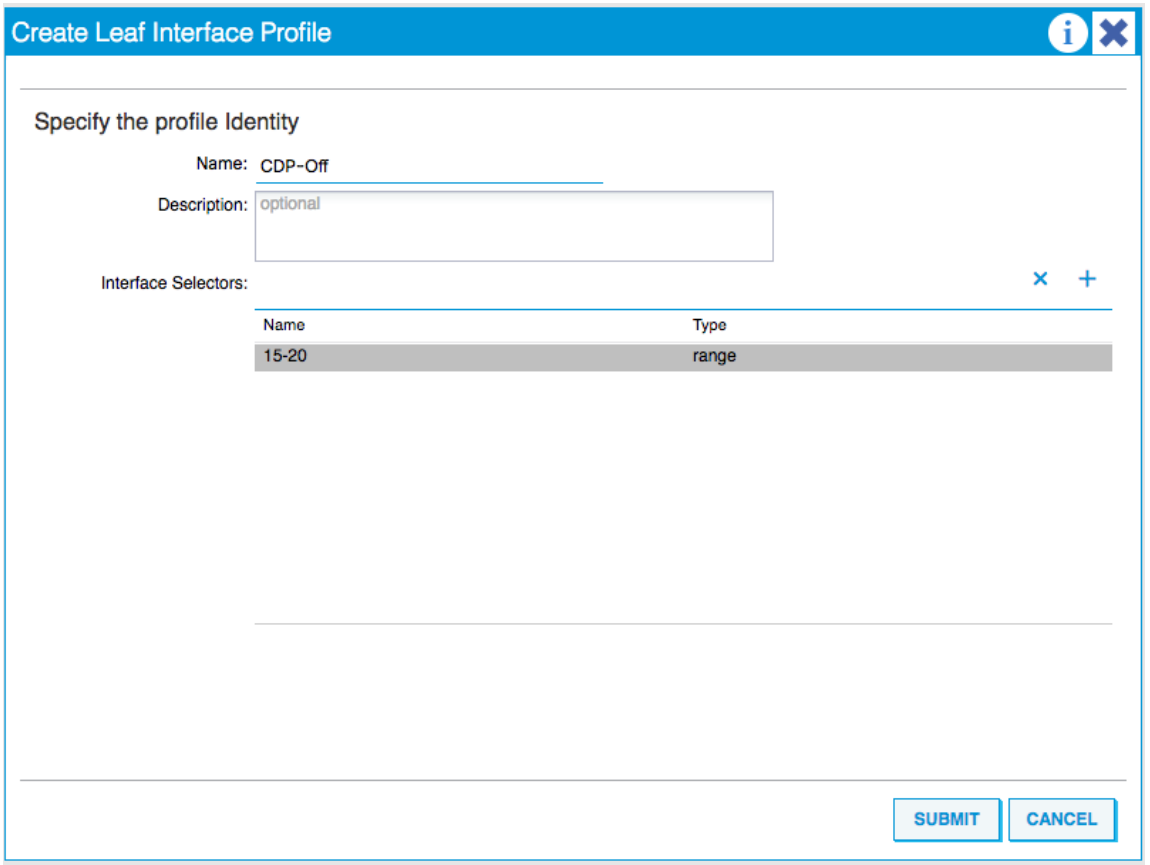
Name the profile and click on the plus sign next to Interface Selectors.



Enter a name for the port selector, and enter the ports in the Interface IDs row. Select CDP-Off as the Interface Policy Group from the drop-down list.

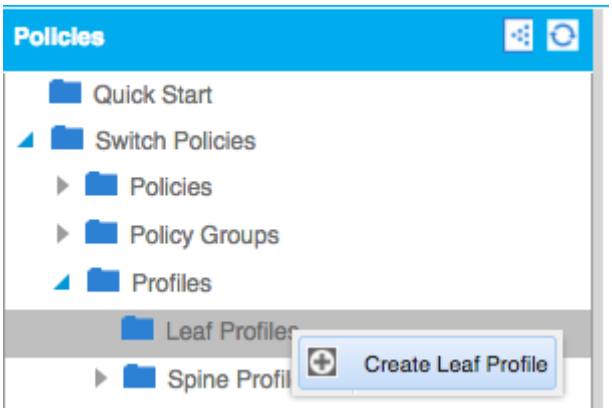


Click OK.

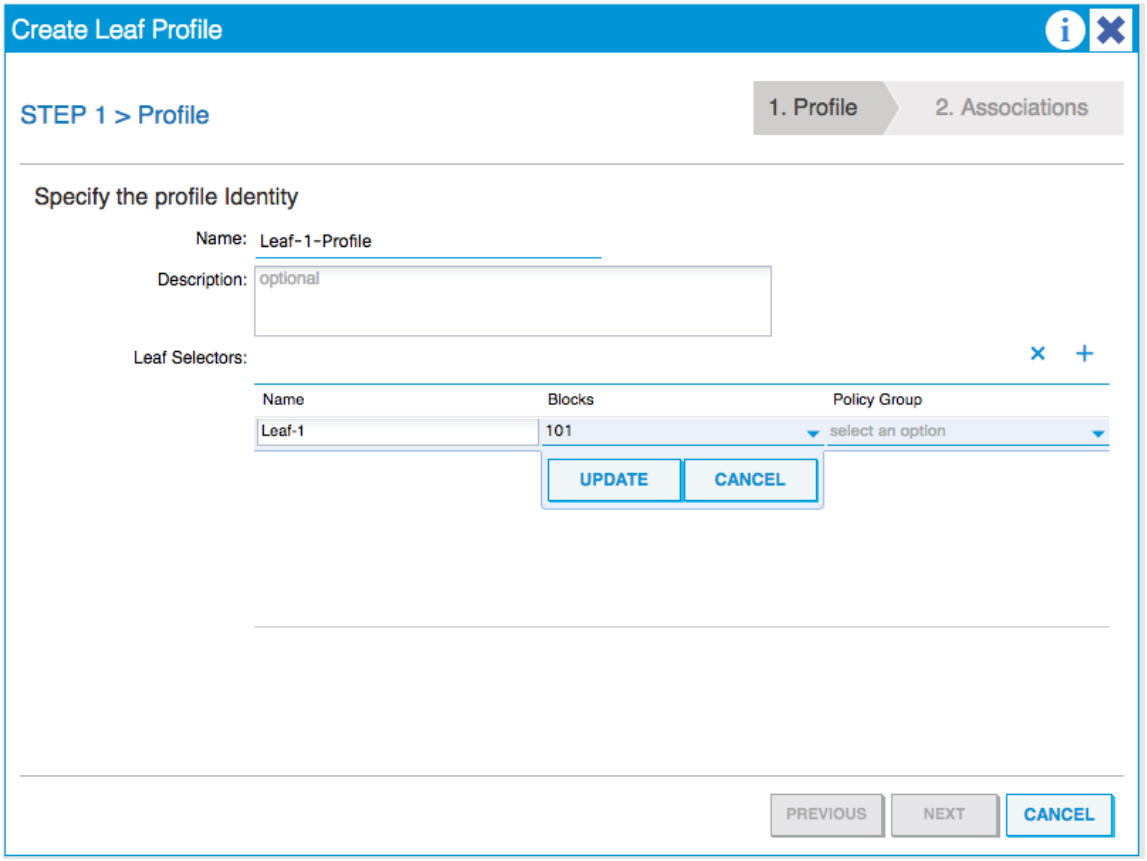


Click Submit.

The next step is to create a switch profile. Navigate to Switch Policies > Profiles. Right-click on Leaf Profiles and select “Create Leaf Profile”.

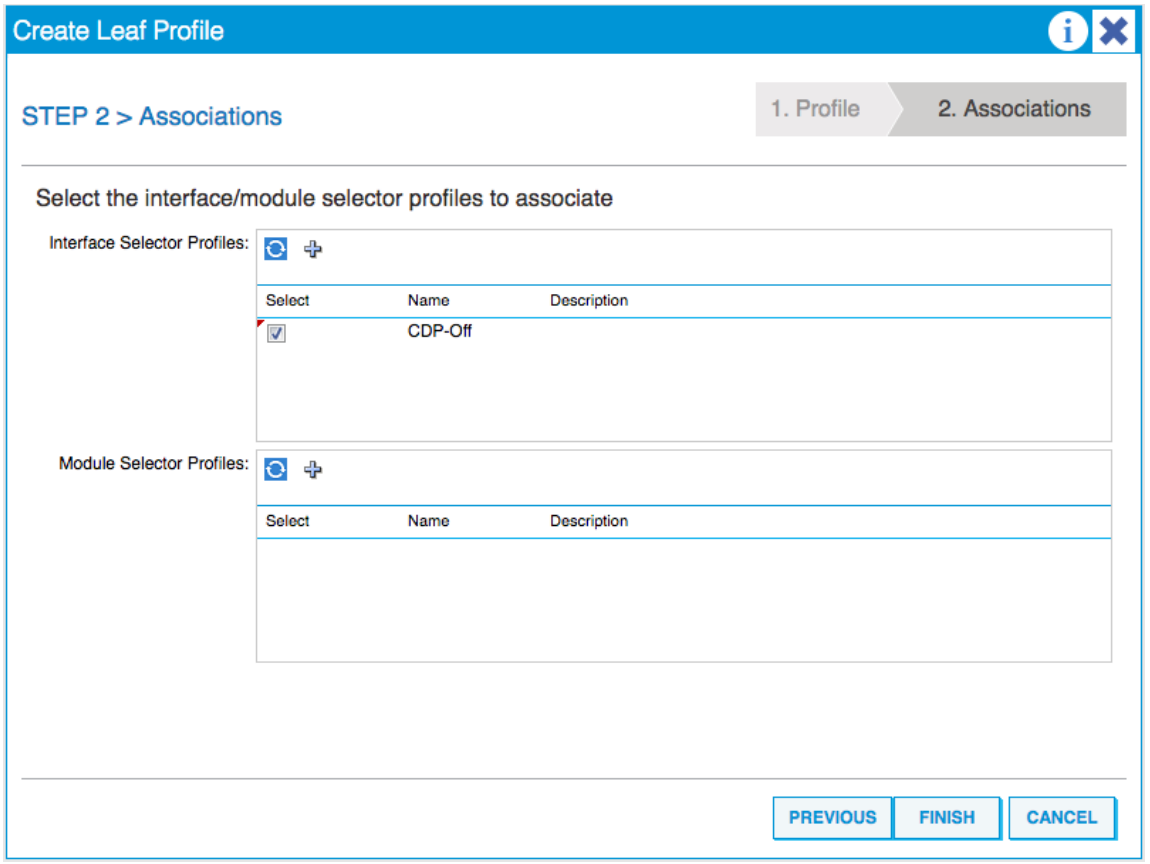


Name the profile and then click on the plus sign next to Leaf Selectors. Select Leaf-1 and click on Update.



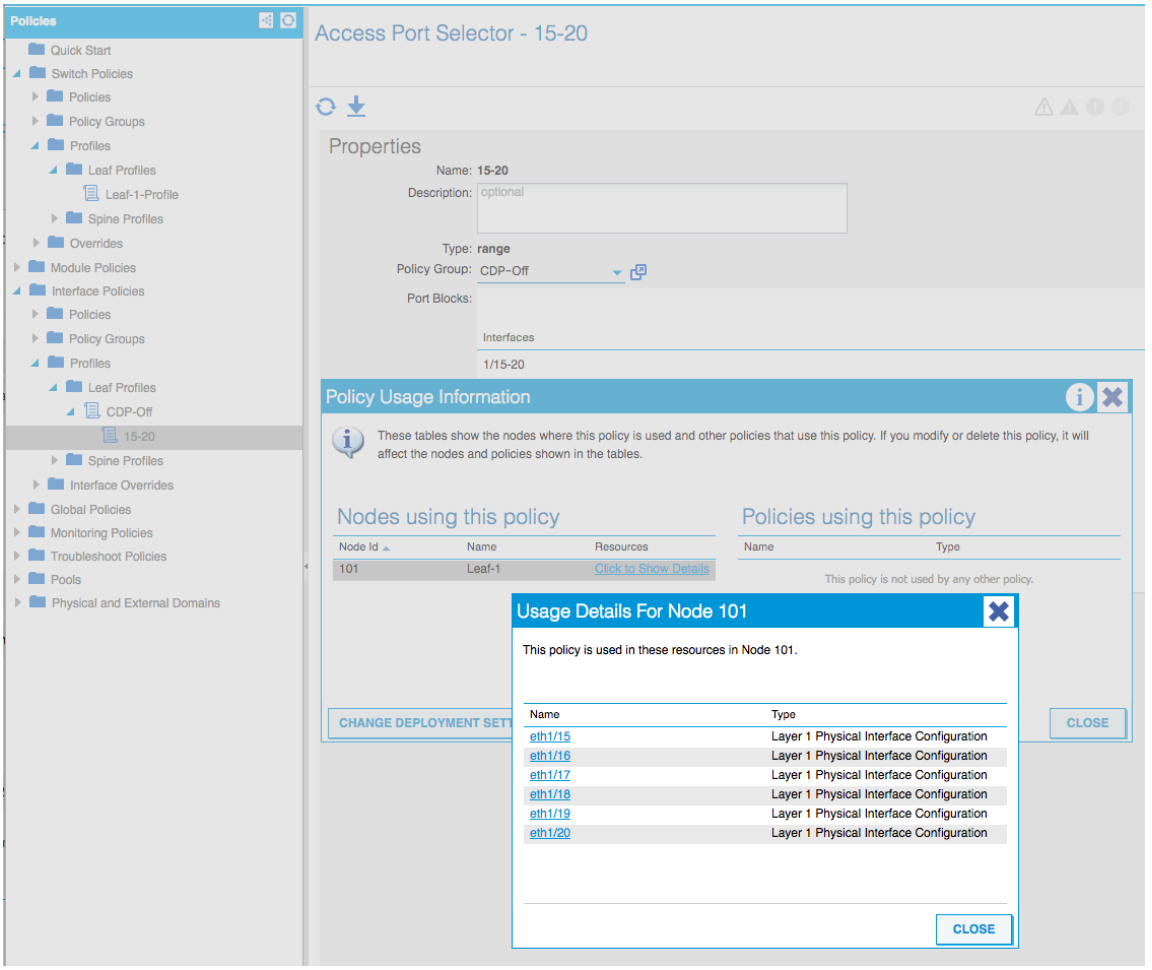
Click Next.

In the next window, select the CDP-Off profile.



Click Finish.

To check the deployment, navigate to Interface Policies > Profiles > Leaf Profiles > CDP-Off and select 15-20. Click on “Show Usage” in the bottom right-hand corner. We can see that the policy is being used by the Leaf-1 node, and clicking on the link under Resources, will show that the 6 interfaces (1/15 to 1/20) are using the CDP-Off policy group.

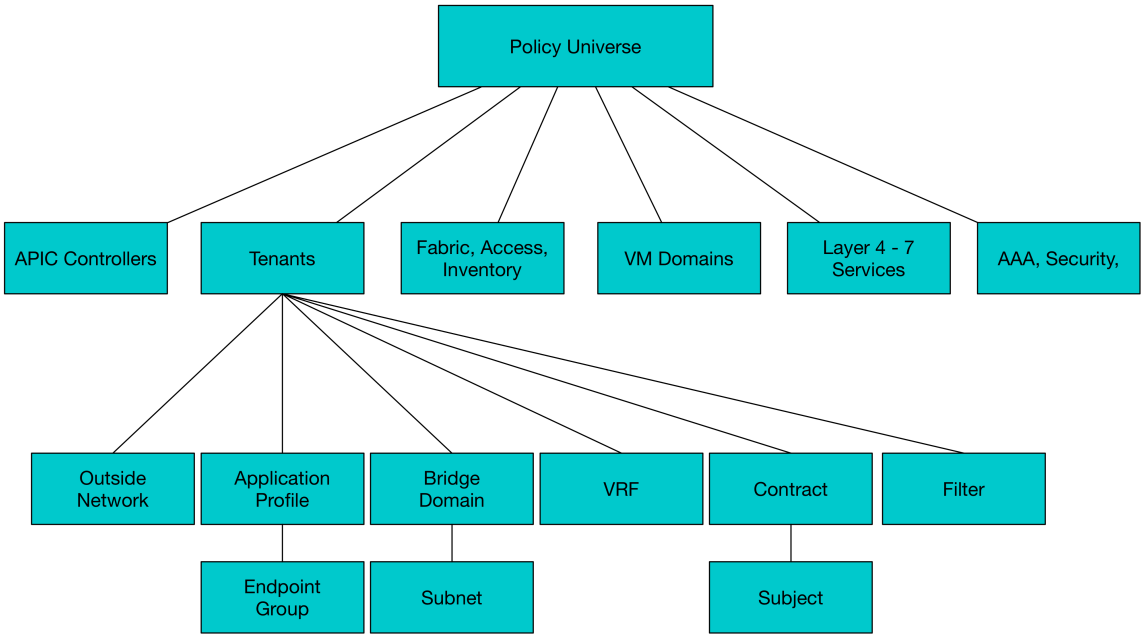


This methidology allows us to have very granular control over the fabric.

Creating Tenants

Tenants can be anything we want them to be (within reason), they can be a customer or a business unit within an enterprise, or a grouping of policies. The term “tenant” is flexible, but each tenant is (by default) an isolated unit within the fabric. It is a logical container, one that can remain self-contained, or, through contracts, share resources with other tenants.

The MIT for the Tenant is shown below.



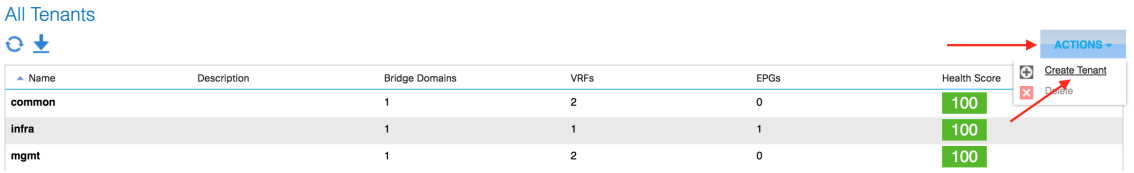
Tenant MIT.

As you can see from the diagram above, tenants contain some different components, including application profiles, bridge domains, VRFs (also referred to as "Contexts"), and contracts. Some of these components, such as bridge domains have their own components, such as subnets.

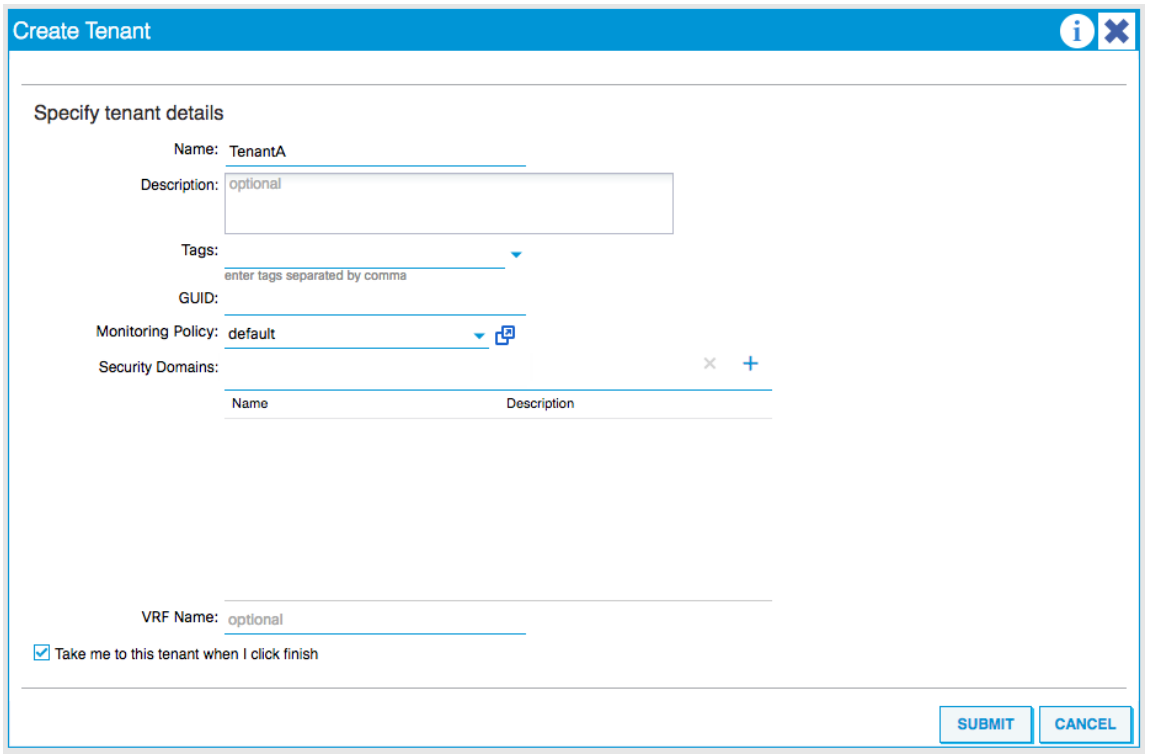
We have a couple of tenants preconfigured. These are the "Common" tenant, which holds policies for shared services, such as firewalls, DNS settings, the "Infrastructure" tenant, which holds policies and VXLAN pools, and the "Mgmt" tenant, or management tenant, which is used for out-of-band access, fabric discovery. The tenants we configure fall under the heading of "User" tenants, and in this recipe, we will create our first tenant.

How to do it…

1. From the Tenants menu, click on the **Actions** menu and select **Create Tenant**.



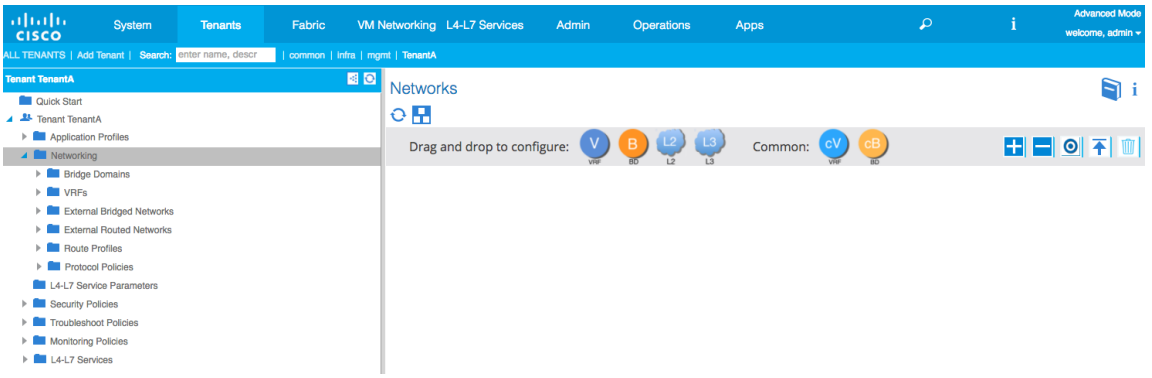
2. In the popup window, give the tenant a name, and click on **Submit**.



We do not need to enter anything for the other fields. Leaving these empty will not prevent us from creating the tenant.

How it works…

By creating a tenant, we are creating a container. If we look at it from the Tenant menu, we can see that we can now drag and drop the components needed into it.



We do not have any components yet, so let's start by configuring a bridge domain.

Configuring Bridge Domains

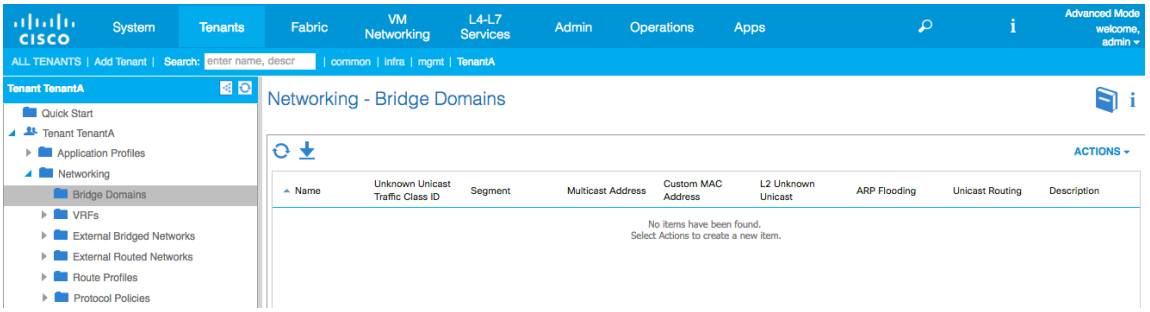
Bridge domains (**BD**) provide layer 2 forwarding within the fabric, as well as a layer 2 boundary. A BD must be linked to VRF (also known as a context) and must have at least one subnet associated with it. BDs define the unique layer 2 MAC address space and also the flood domain (if flooding is enabled).

Bridge domains can be public, private, or shared. Public bridge domains are where the subnet can be exported to a routed connection, whereas private ones apply only within the tenancy. Shared bridge domains can be exported to multiple VRFs within the same tenant, or across tenants when part of a shared service.

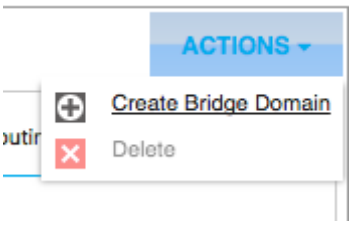
In this recipe, we will create a bridge domain and, along with it, define a VRF and a subnet for communication within the tenancy.

How to do it…

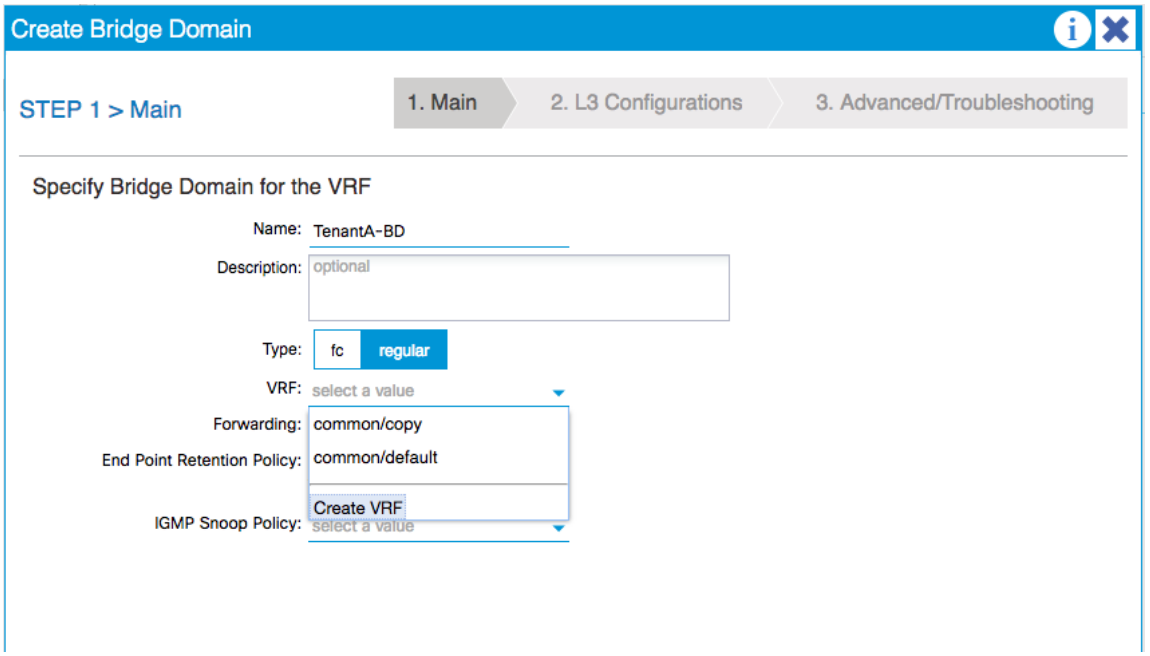
1. We start by going into the tenant we created in the previous recipe and clicking on **Networking and then Bridge Domains**.



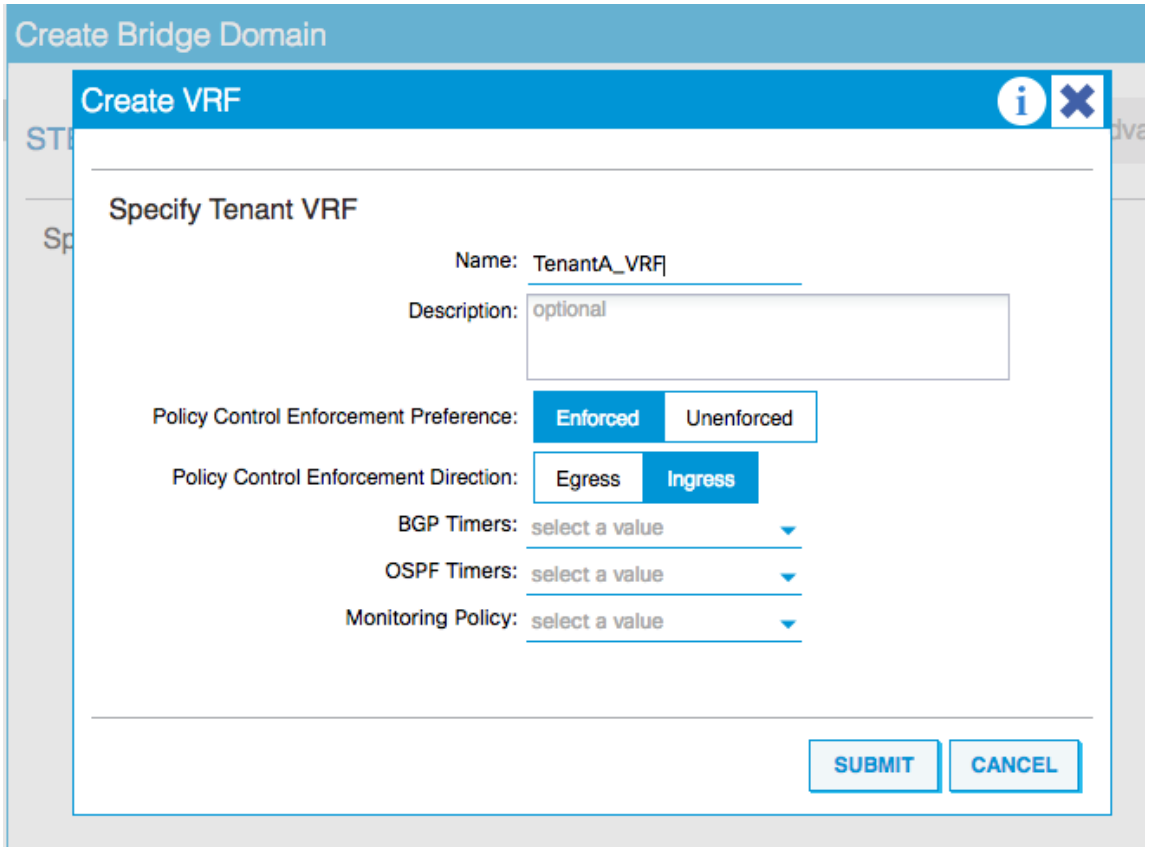
2. Click on **Actions**, then on **Create Bridge Domain**.



3. This launches a new window. Here we name our bridge domain and assign a VRF to it if we have already created one, or create a new VRF.

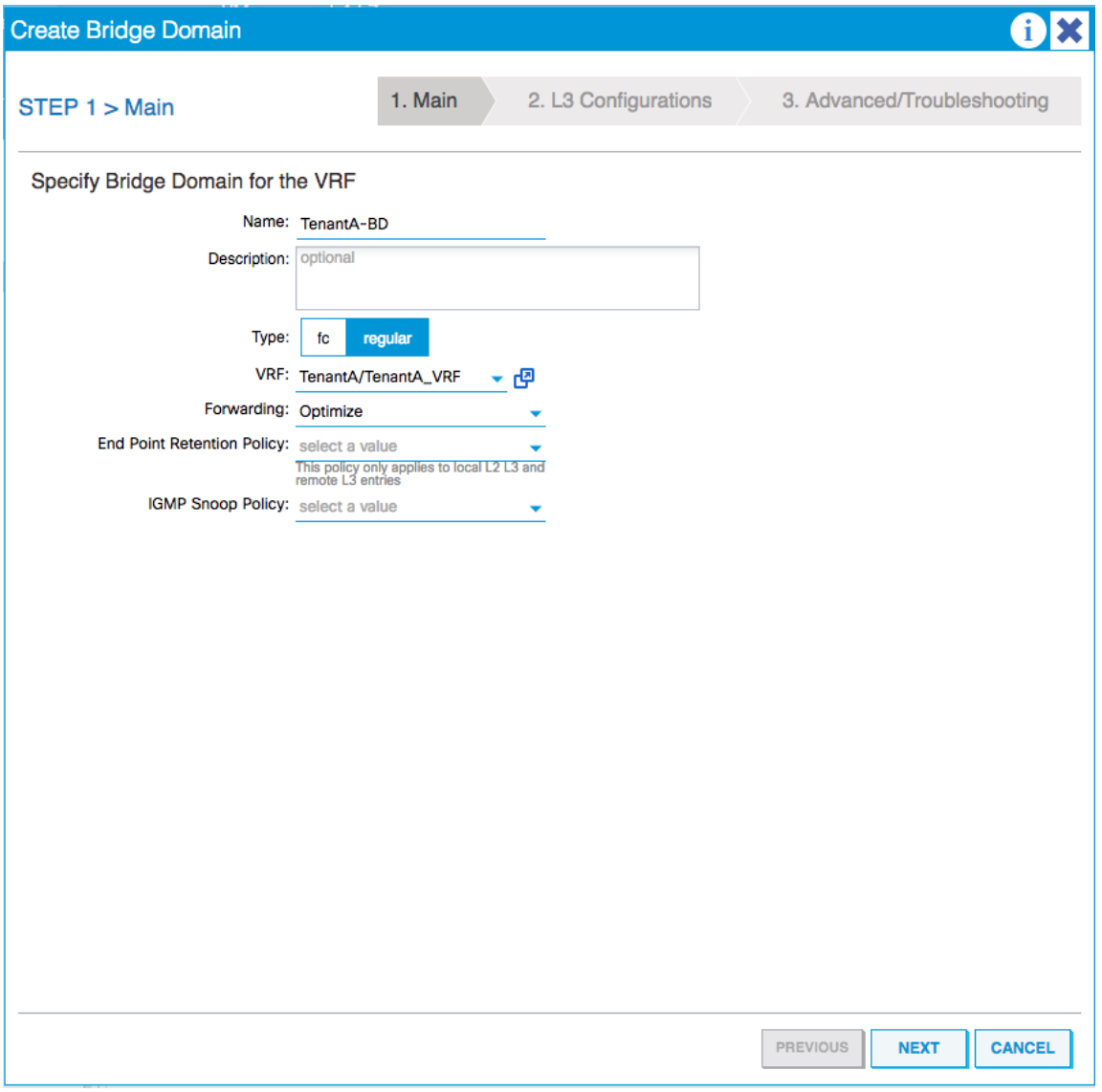


If we choose **Create VRF**, this brings up another window:

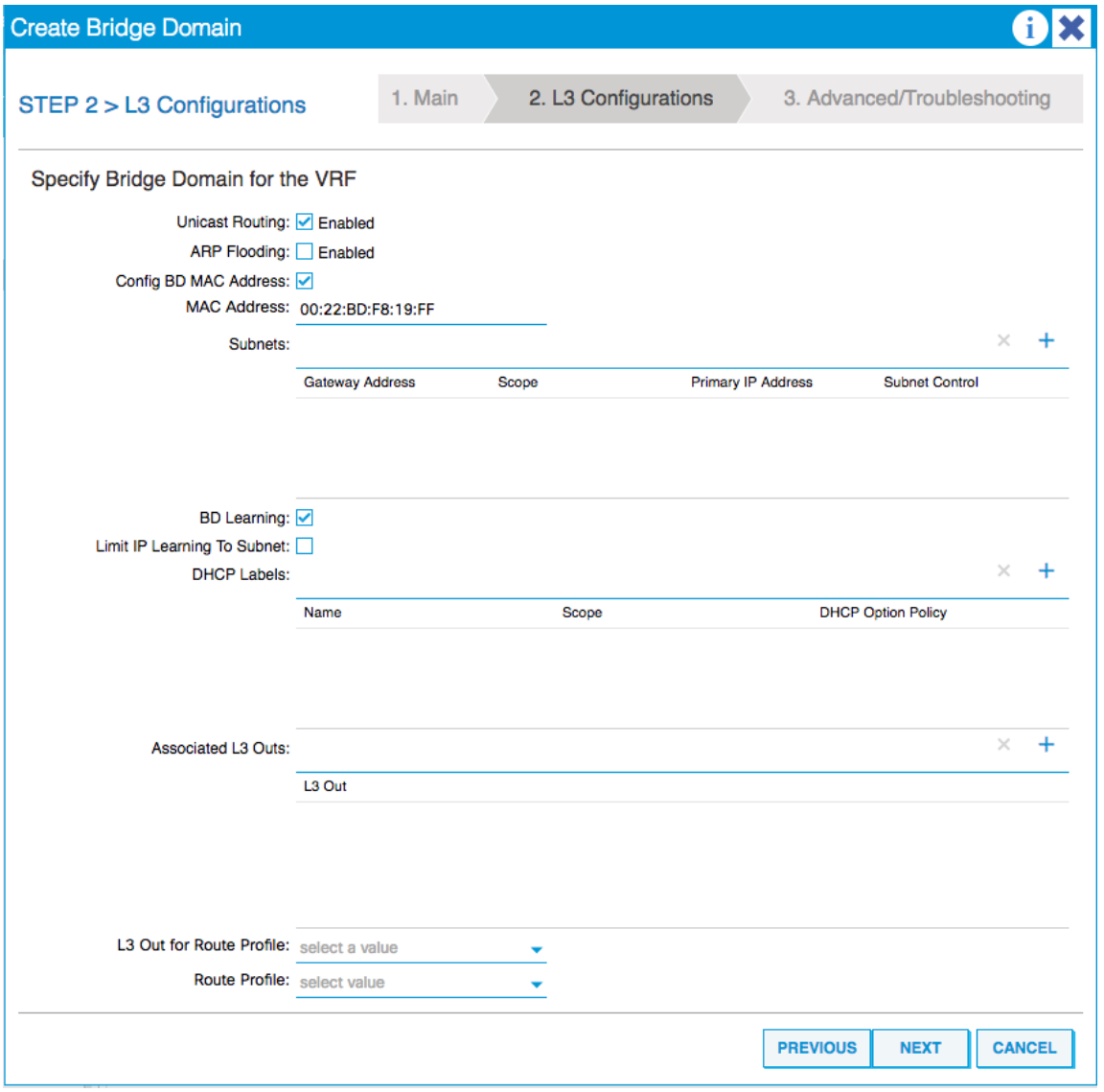


Because we have not created any timer policies for BGP or OSPF or a monitoring policy, we can leave these fields empty and use the default values.

4. Once we click on **Submit** the new VRF is selected:



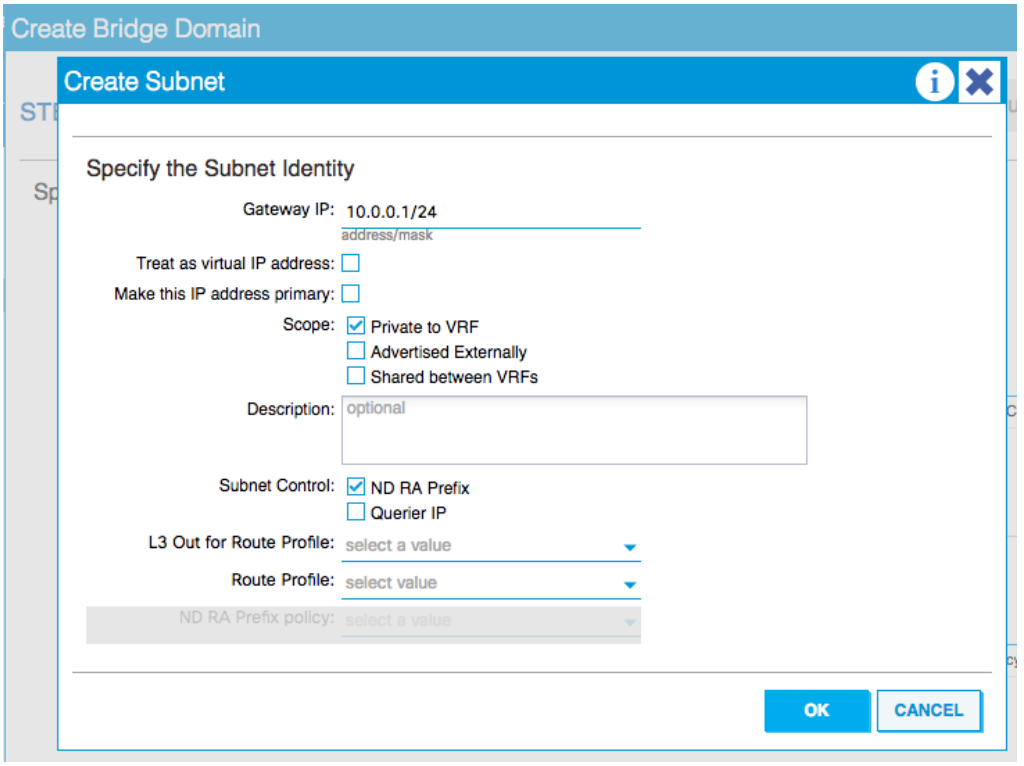
5. We can set the forwarding to **Optimize**, and leave the End Point Retention Policy and IGMP Snoop Policy to the defaults and click **Next** to take us to the L3 Configurations window.



This is where we enable Unicast Routing, ARP flooding (if we want to), specify a MAC address, and create a subnet.

ARP flooding is disabled by default. The fabric will convert any ARP broadcast traffic into unicast traffic and push it to the destination leaf node. If we want to enable traditional ARP flooding behavior, this is where we would enable it.

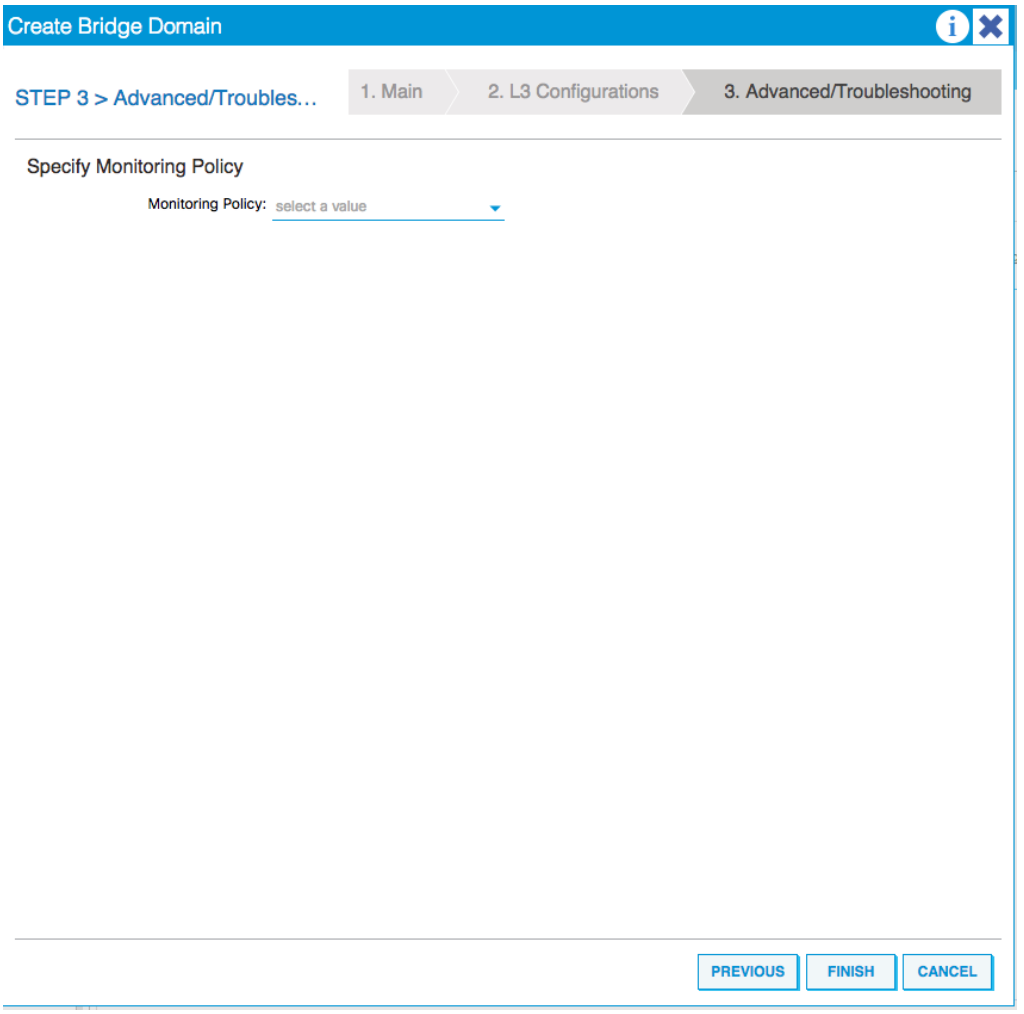
6. To create a subnet click on the plus sign, which brings up another window:



7. Here we specify the subnet and subnet mask for the network and set the scope to be private, public ("Advertised Externally") or shared.

6. **Private to VRF** means that the subnet will not be advertised externally (outside of the VRF). **Advertised Externally** means just that, and will be flagged for advertising through a routing protocol to an external device. **Shared between VRFs** is similar to advertising externally, but is kept within the fabric. Because we are only concentrating on TernantA, at this stage, we will use the Private to VRF scope.

7. Click **OK** which will take us back to the L3 Configurations window.
8. Click next to take us to the final window, where we can select a monitoring policy.

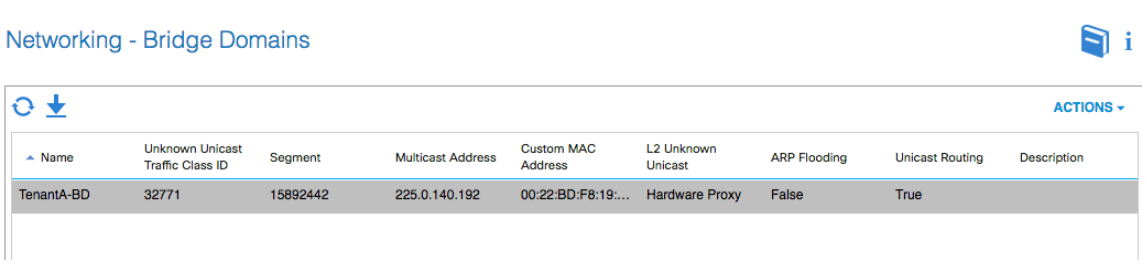


9. Click Finish.

How it works…

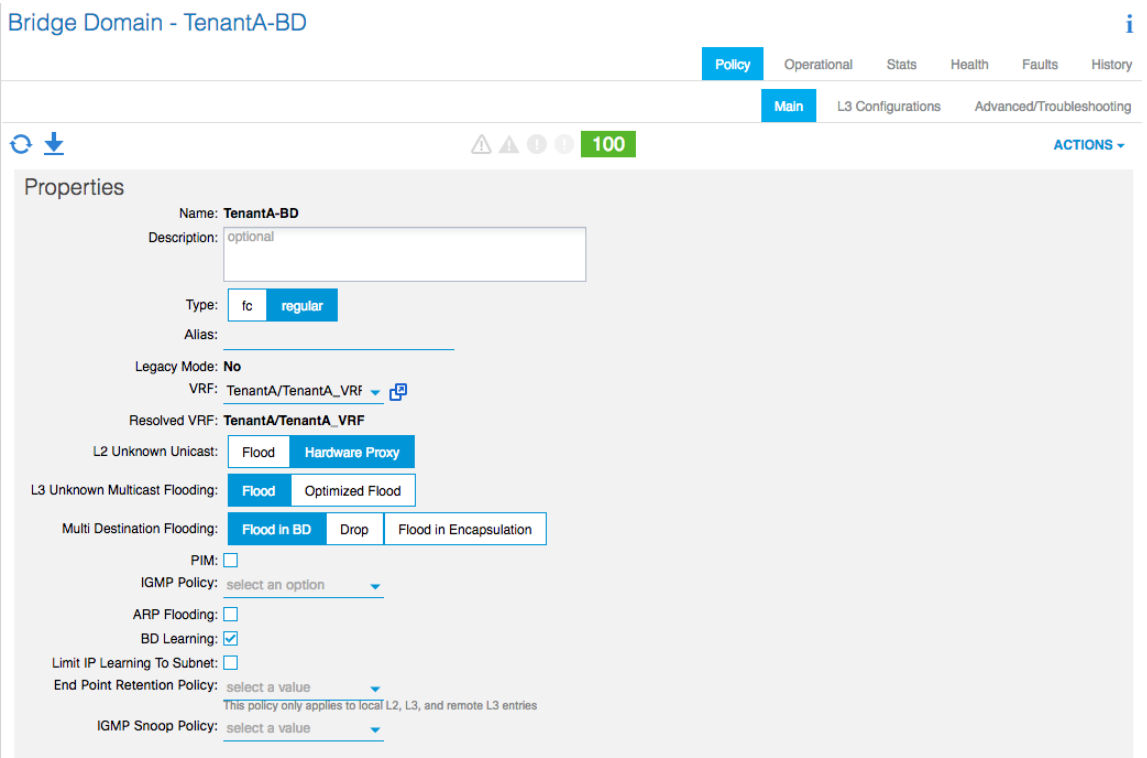
We can see the bridge domains we have created from the Networking / Bridge Domains menu within the tenant settings.

This page gives us an overview of all of the bridge domains associated with a tenant, including the multicast address.

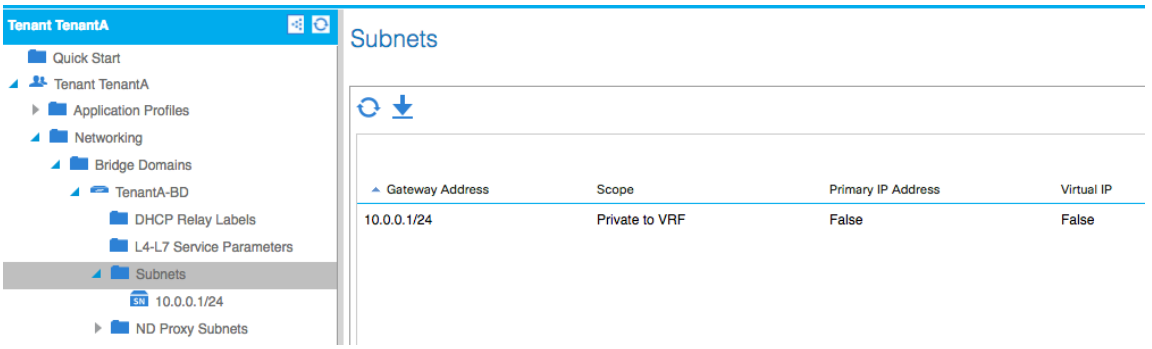


Because bridge domains permit multicast traffic, but at the same time, isolate it from other bridge domains, each bridge domain will get its own multicast address.

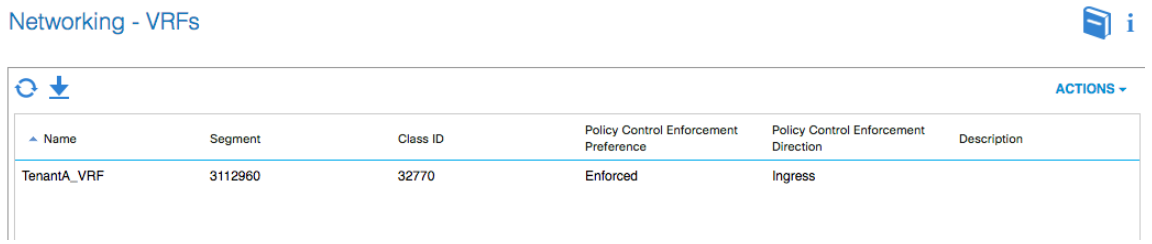
Clicking on a particular bridge domain takes us into all of the settings for it, and allows us to change the associated VRF, as well as changing the flooding settings and policies.



The subnet we created can be found within the Bridge Domains menu, under the configured bridge domain menu, and then Subnets menu:



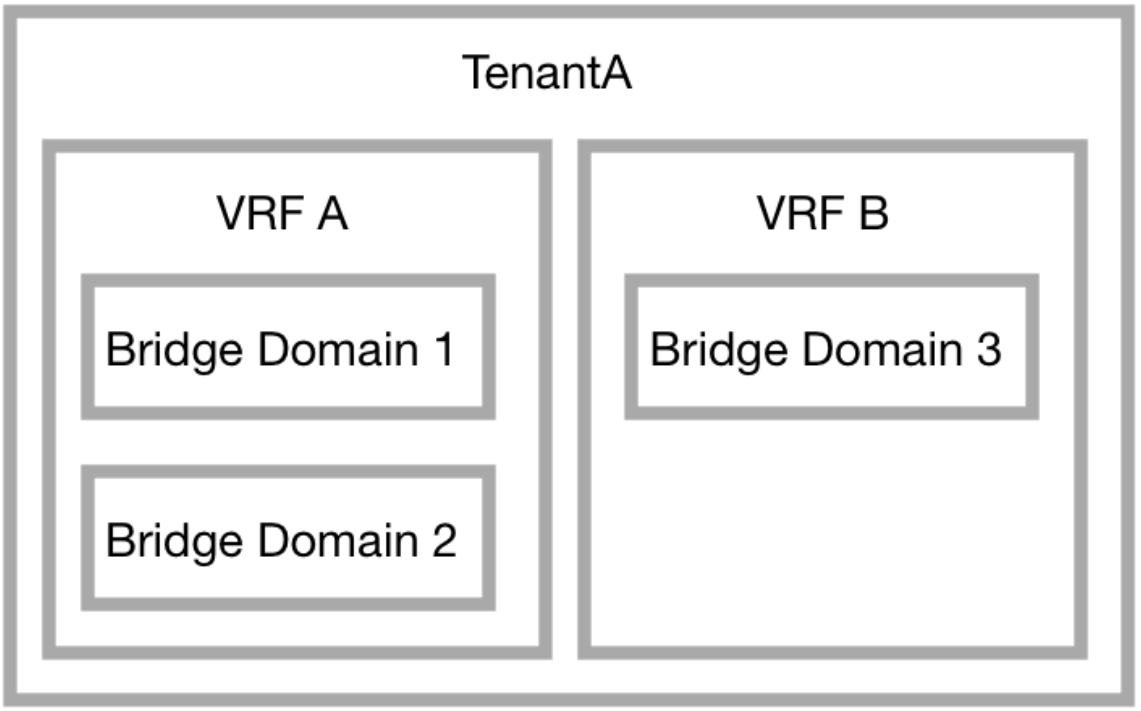
Lastly, the VRF we created can be found under Networking and then VRFs menu:



We can create additional VRFs if we want to, which we shall do in the next recipe.

Configuring Contexts

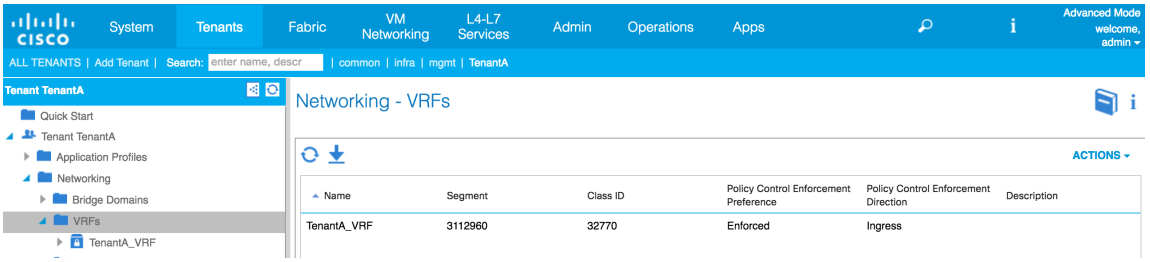
So far, we have configured a tenant and created a bridge domain and context for it. Contexts, also known as **VRF**s (**Virtual Routing and Forwarding**), are unique layer 3 forwarding domains. We can have multiple VRFs within a tenant, and VRFs can be associated with more than one bridge domain (but we cannot associate a bridge domain with more than one VRF).



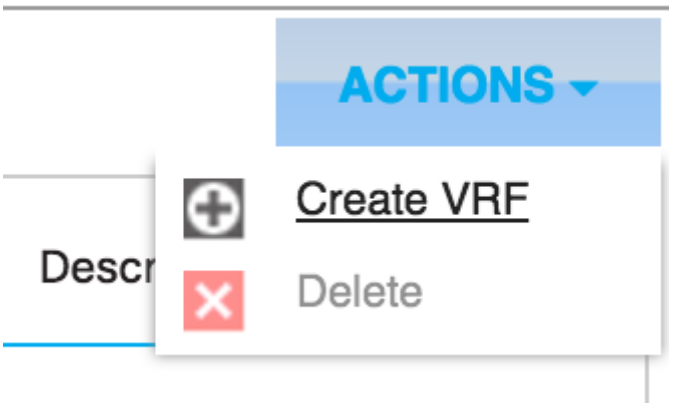
In this recipe, we will create a second VRF under TenantA and a new bridge domain.

How to do it…

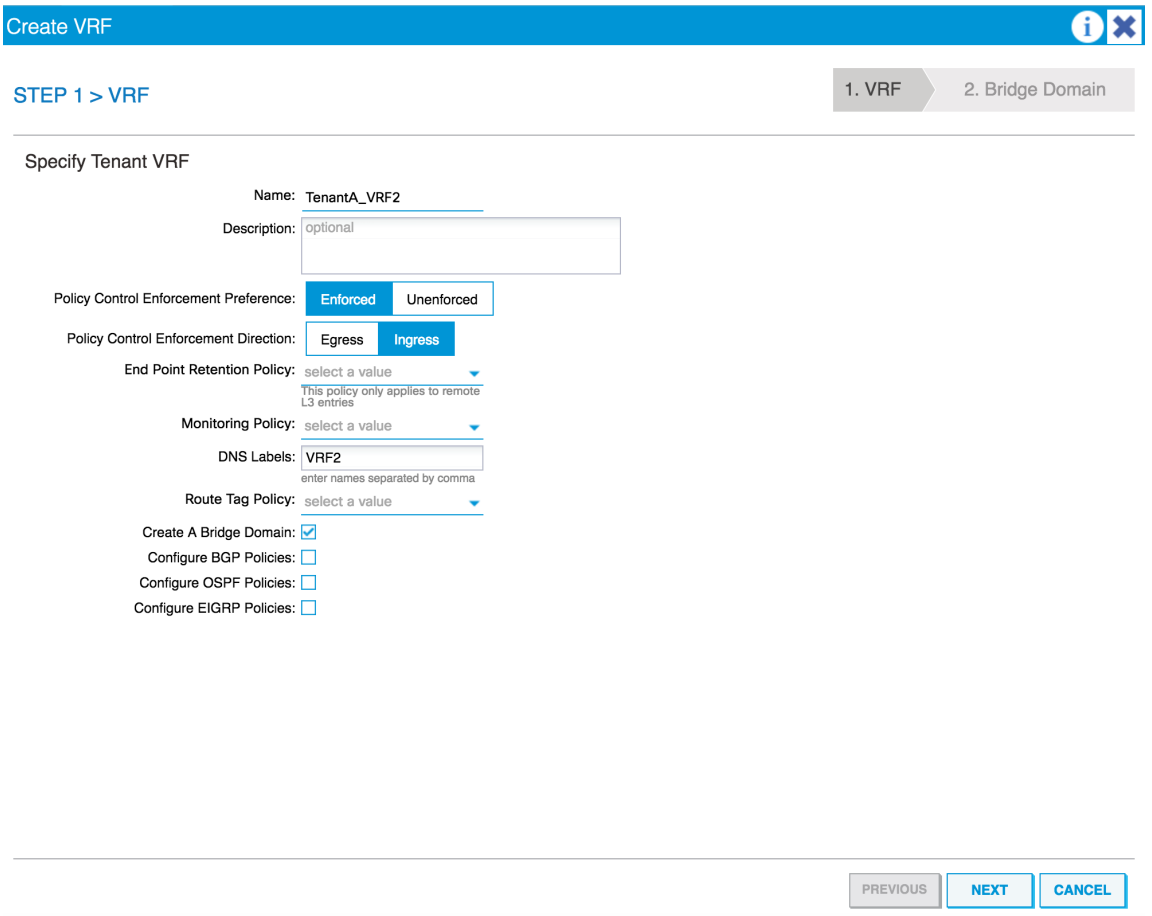
1. From the Networking menu under the tenant, select **VRFs**.



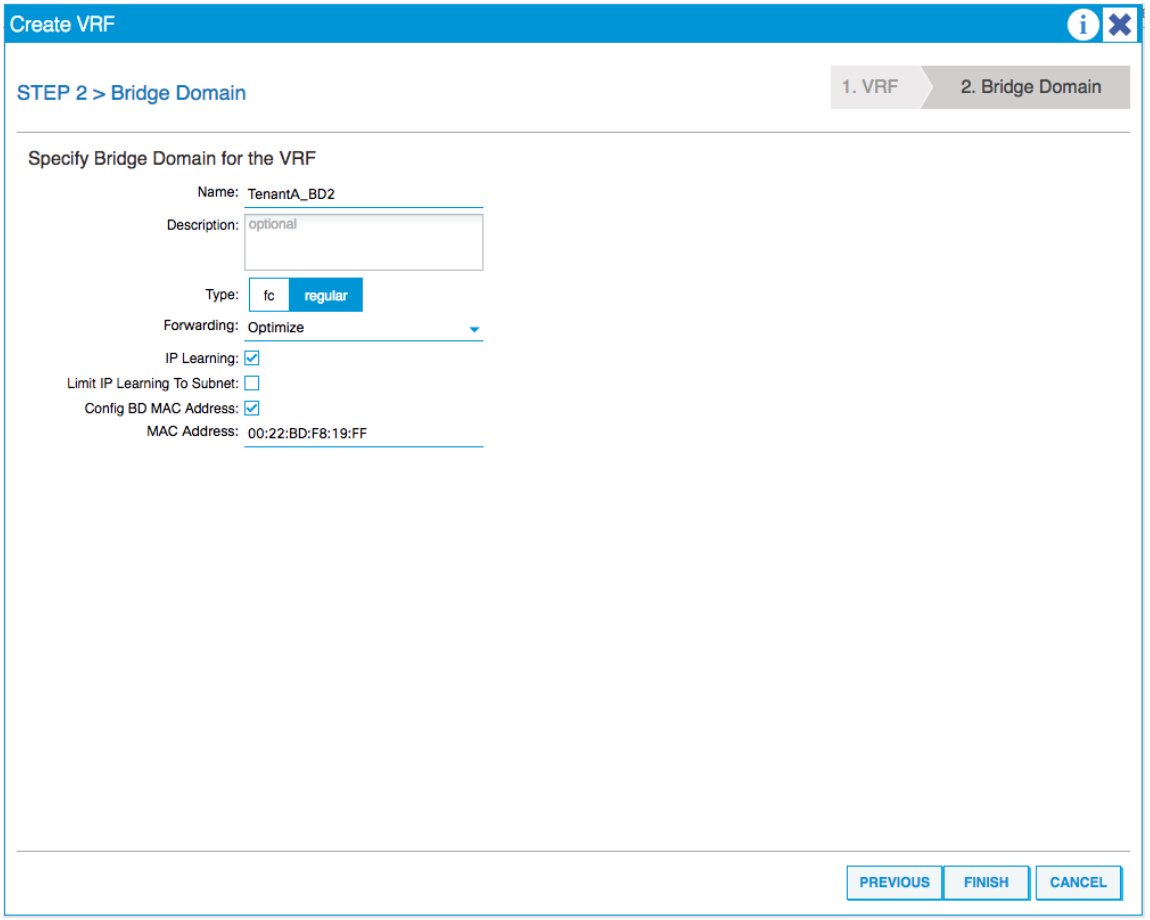
2. From the Actions menu, select **Create VRF**



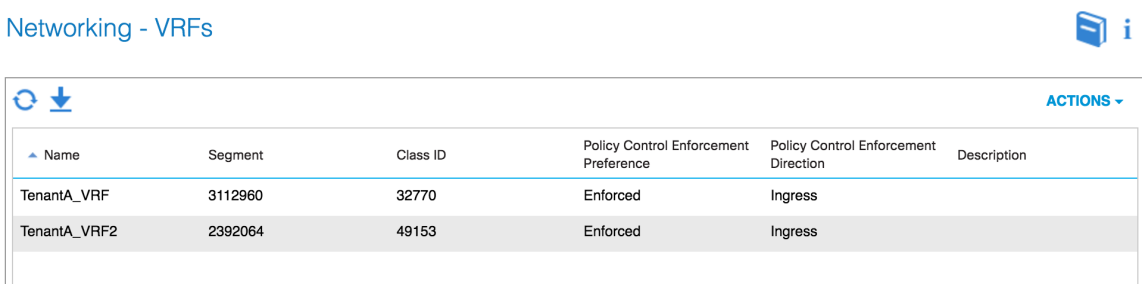
3. Give the VRF a name, and select any applicable options. Here I have chosen to add a DNS label of **VRF2**. We can create a new bridge domain (this is the default option) at this stage as well.



4. Click **Next**, and select the **Forwarding** settings (such as “Optimize”), and change any defaults (which I have not changed).

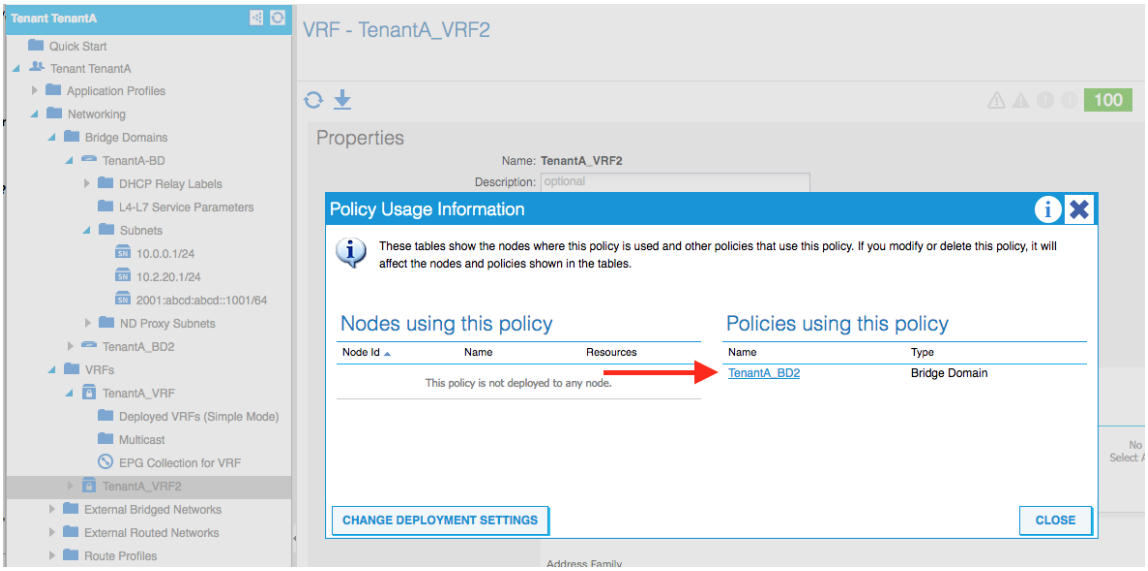


5. When you have done this, click **Finish**. The new VRF will be shown underneath the first VRF we created.



How it works…

Selecting the newly created VRF from the left-hand side, we can click on "Show Usage" to see which bridge domain it is associated to:

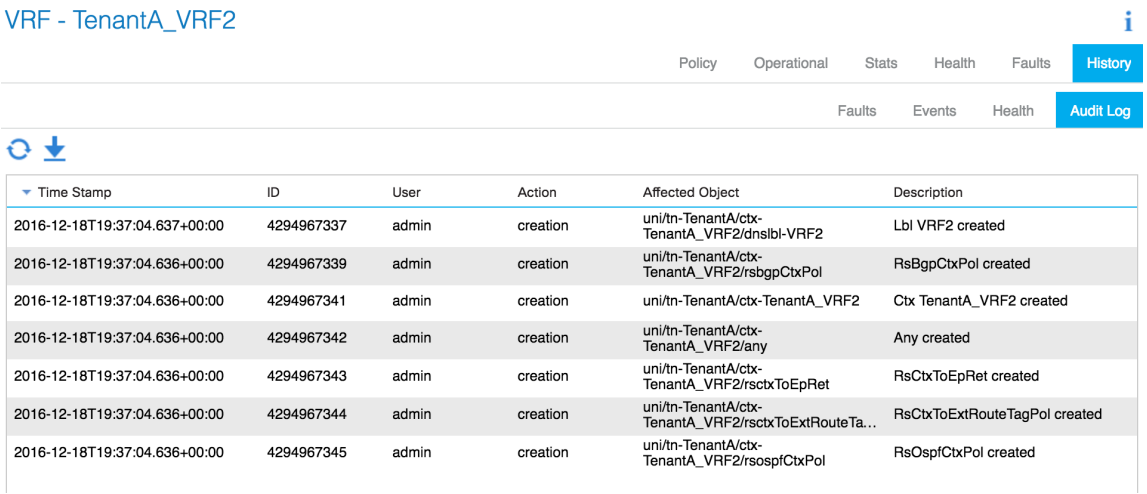


Going into the VRF, we can see the policy details, as we saw previously. We can also look at associated EPGs and external routed networks, from the Operational tab.

Using the Stats menu, we can see a graph of the traffic (unicast and multicast) that passes through the VRF.

From the Health menu, we can look at the health of the VRF and, as we get further with configuring our fabric, a diagram of the relationships between objects. This view will also allow us to troubleshoot the entire path.

We can also see any faults that may have occurred, as well as the history for the VRF, such as the audit log.

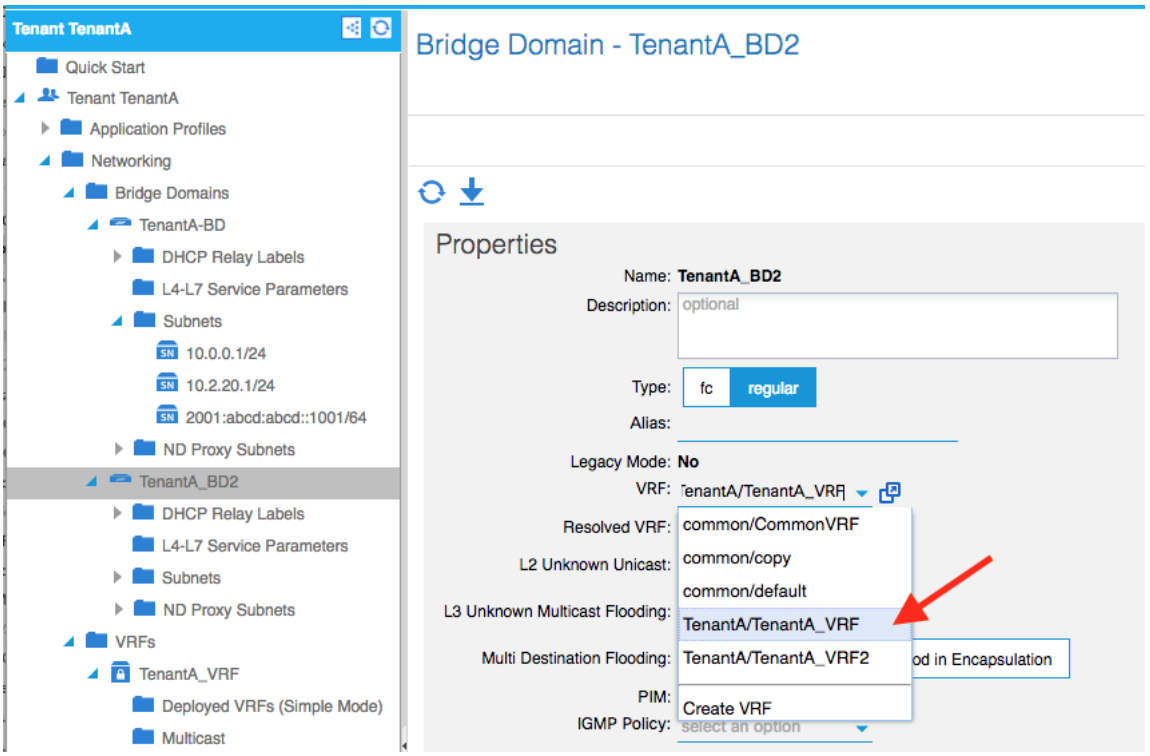


Time Stamp	ID	User	Action	Affected Object	Description
2016-12-18T19:37:04.637+00:00	4294967337	admin	creation	uni/tn-TenantA/ctx-TenantA_VRF2/dnslbl-VRF2	Lbl VRF2 created
2016-12-18T19:37:04.636+00:00	4294967339	admin	creation	uni/tn-TenantA/ctx-TenantA_VRF2/rsbgpCtxPol	RsBgpCtxPol created
2016-12-18T19:37:04.636+00:00	4294967341	admin	creation	uni/tn-TenantA/ctx-TenantA_VRF2	Ctx TenantA_VRF2 created
2016-12-18T19:37:04.636+00:00	4294967342	admin	creation	uni/tn-TenantA/ctx-TenantA_VRF2/any	Any created
2016-12-18T19:37:04.636+00:00	4294967343	admin	creation	uni/tn-TenantA/ctx-TenantA_VRF2/rsctxToEpRet	RsCtxToEpRet created
2016-12-18T19:37:04.636+00:00	4294967344	admin	creation	uni/tn-TenantA/ctx-TenantA_VRF2/rsctxToExtRouteTa...	RsCtxToExtRouteTagPol created
2016-12-18T19:37:04.636+00:00	4294967345	admin	creation	uni/tn-TenantA/ctx-TenantA_VRF2/rsospfCtxPol	RsOspfCtxPol created

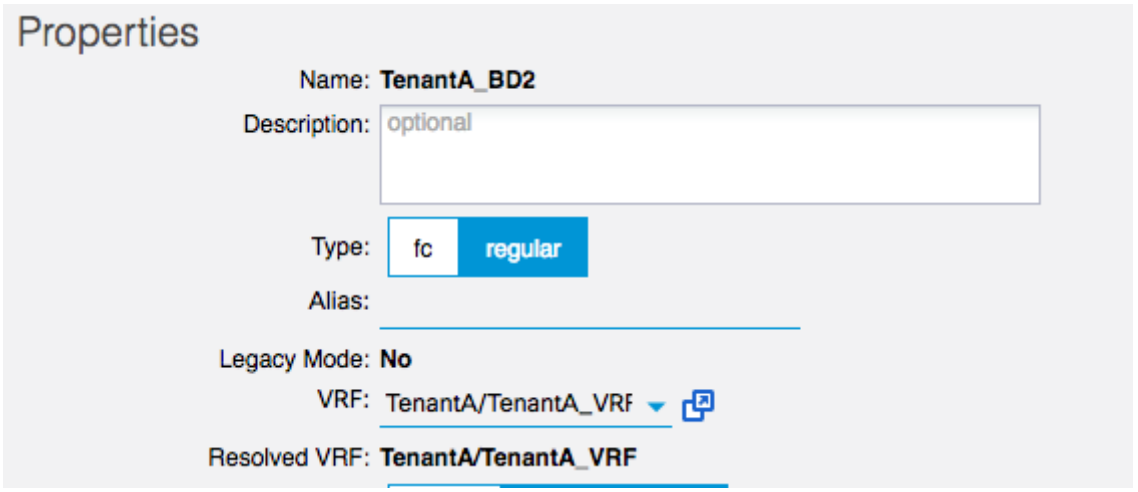
Looking at the audit is useful for two reasons. Firstly, we can see that we will have full tracking of actions through RBAC. Each action is time stamped along with the user who made the change. Secondly, we can start to see the kind of commands used by the API.

There's more…

If we wanted our new bridge domain (TenantA_BD2) to be associated with our first VRF (from the bridge domains recipe), we could do this. We select the bridge domain, and from the VRF drop-down, select TenantA/TenantA_VRF.

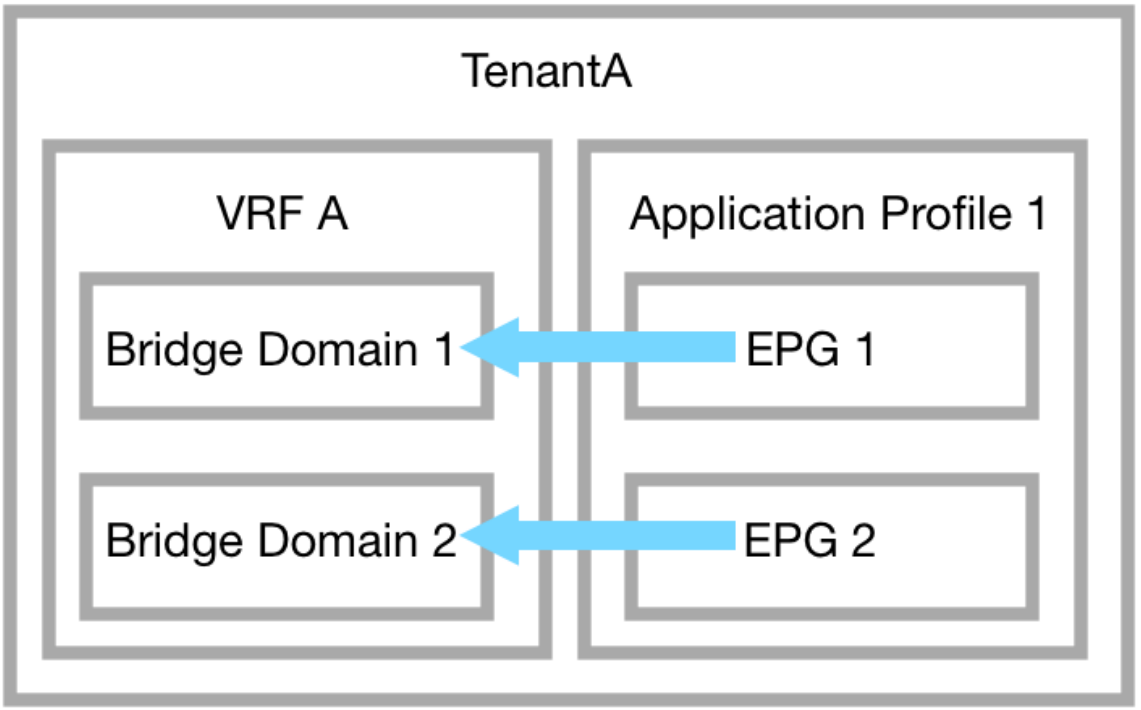


Click Submit and accept the policy change warning popup. The VRF will change along with the resolved VRF.



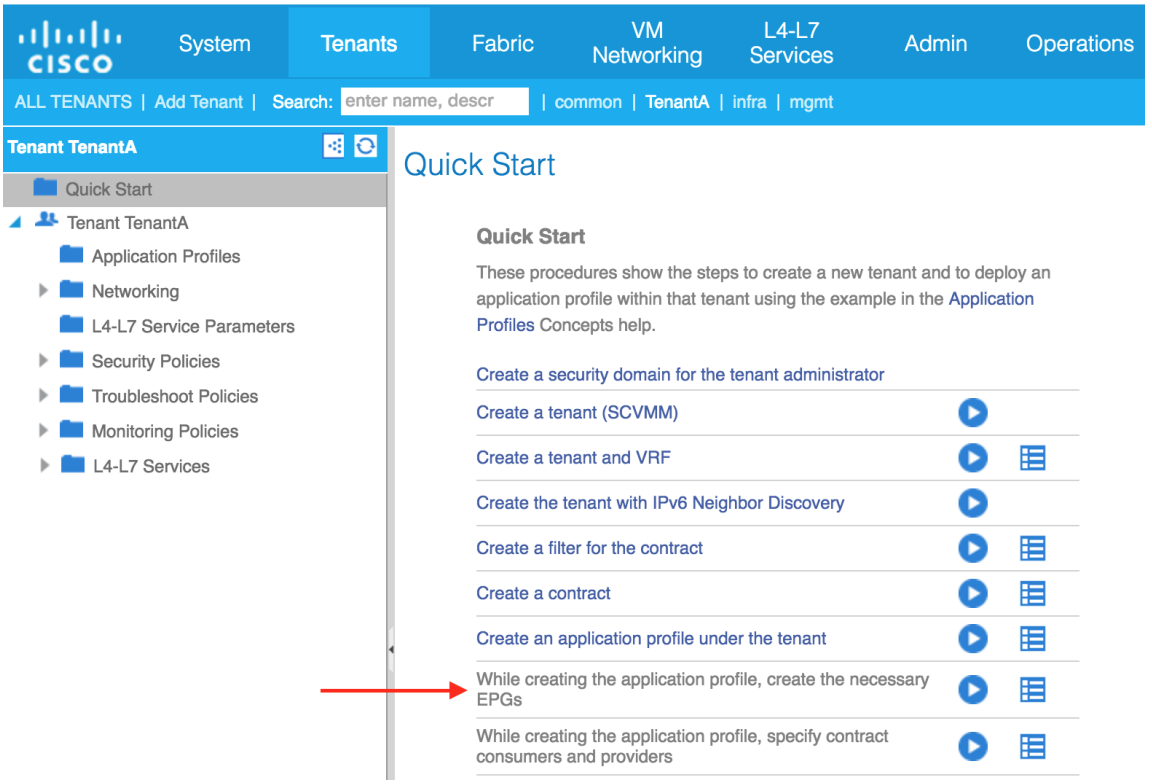
Creating Application Network Profiles

Application Profiles (**APs**) are containers for the grouping of **Endpoint Groups** (**EPGs**). We can have more than one EPG with an AP. For example, an AP could group a web-server with the backend database, with storage, and so on. EPGs are assigned to different bridge domains.



Application Profiles define different aspects to the tenancy, governing security, **Quality of Service** (**QoS**), **Service Level Agreements** (**SLA**s) and layer 4 to layer 7 services.

APs are so intrinsically linked to EPGs (and contracts to a lesser extent) that it is harder to create these as separate tasks. For this reason, we will create them in one recipe. As you can see from the picture below, we are even guided in the quick start to create the EPGs and contacts when we create the application profile.

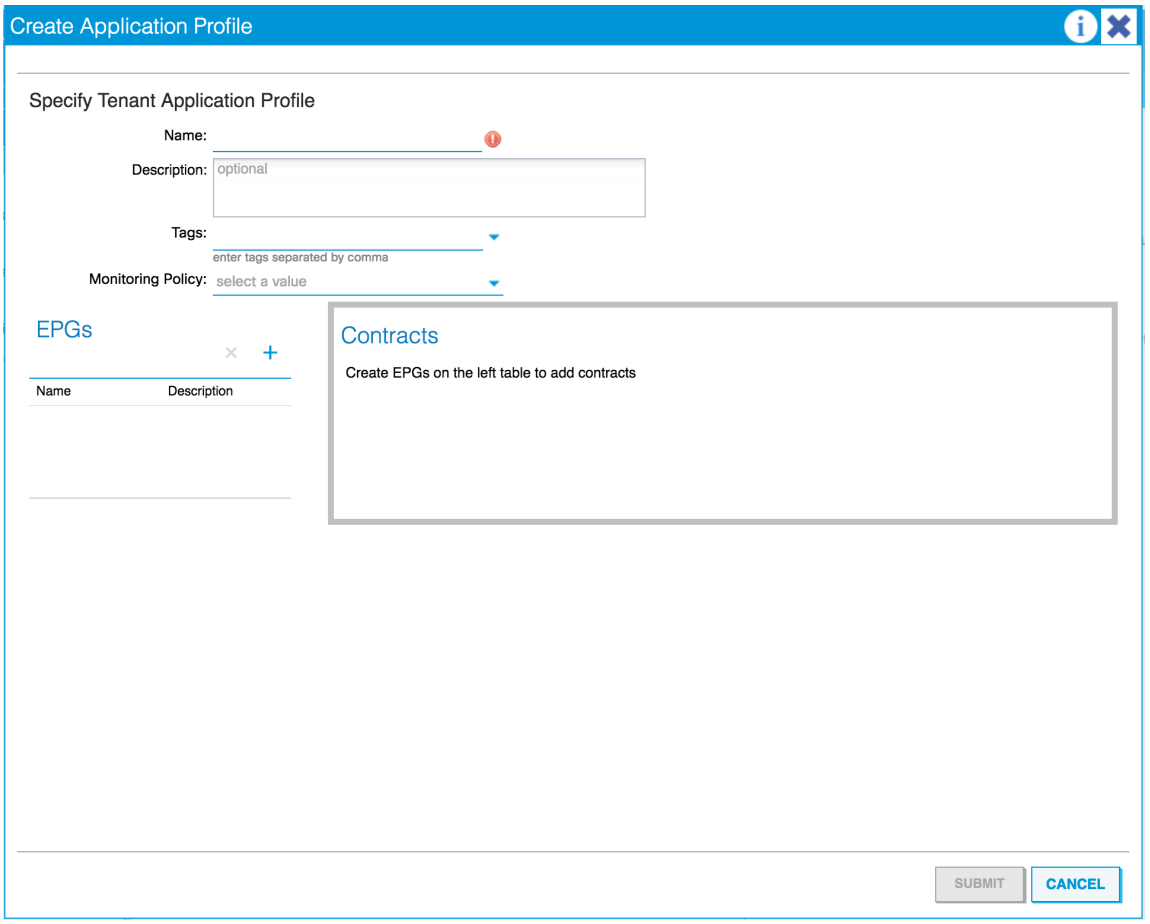


If we click on the grid-style icon next to the play button the help window for the task will pop up.

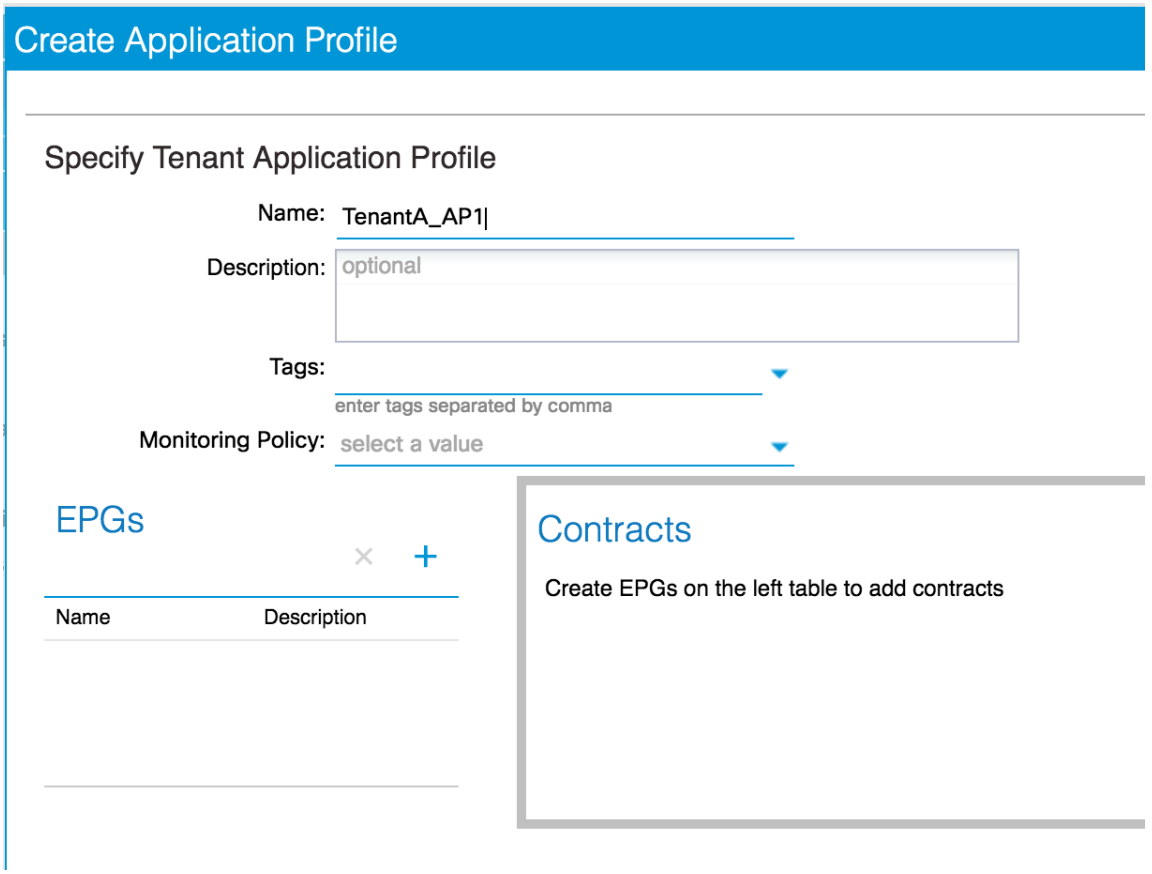
How to do it…

1. We can create an AP from the quick start menu by clicking on the **Create an application profile under the tenant** link. This method is slightly different to selecting Application Profiles from the left-hand side menu and then using the actions menu to select **Create Application Profile**. The end result is the same, however.

   The window that appears is shown in figure 74.



2. We need to enter a name (such as TenantA_AP1):



We also need to create an Endpoint Group, which we will do in the next recipe.

Creating Endpoint Groups

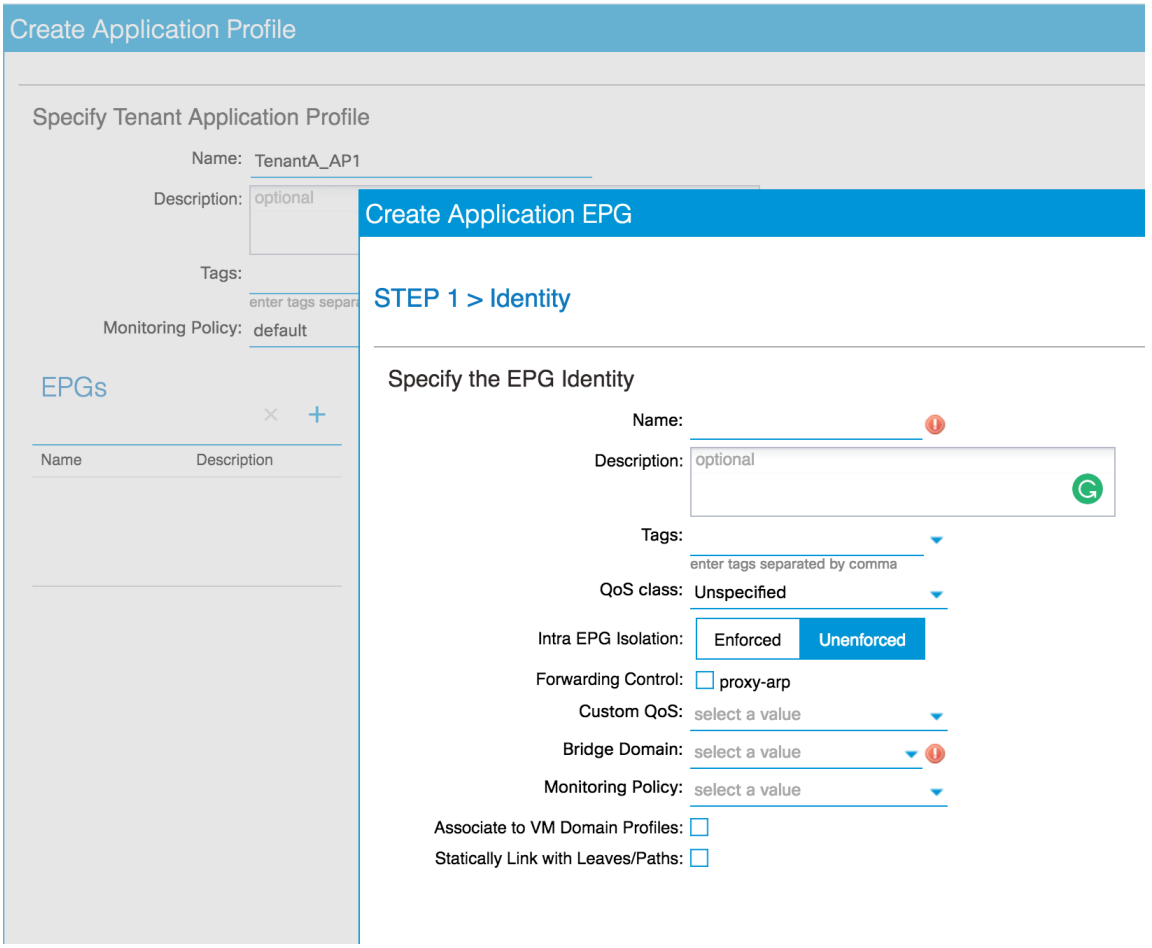
Endpoint Groups are managed objects that contain (unsurprisingly) endpoints. Endpoints are devices that are connected to the network, either directly or indirectly. Endpoints have certain attributes, such as an address, a location; they can be physical or virtual. Endpoint groups are a logical grouping of these, based on common factors. The factors are more business-related, such as having common security requirements, whether the endpoints require virtual machine mobility, have the same QoS settings, or consume the same L4-L7 services. Therefore, it makes sense to configure them as a group.

EPGs can span multiple switches and are associated with one bridge domain. There is not a

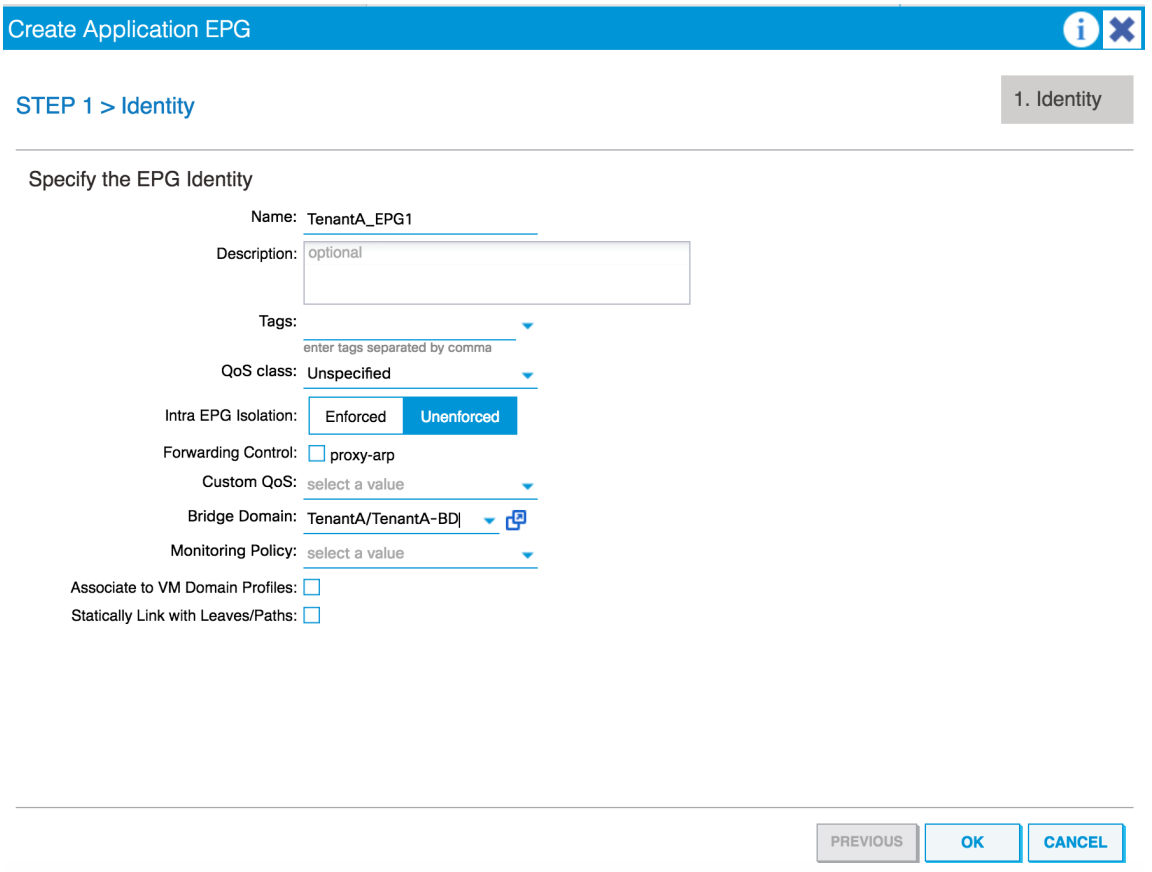
one-to-one mapping between an EPG and particular subnets, and one cool thing about membership in an EPG is that it can be static for physical equipment, or dynamic when we use the APIC in conjunction with virtual machine controllers, again this will cut down on the number of manual configuration steps.

How to do it…

1. From within the Create Application Profile window, click on the plus sign next to "EPGs."

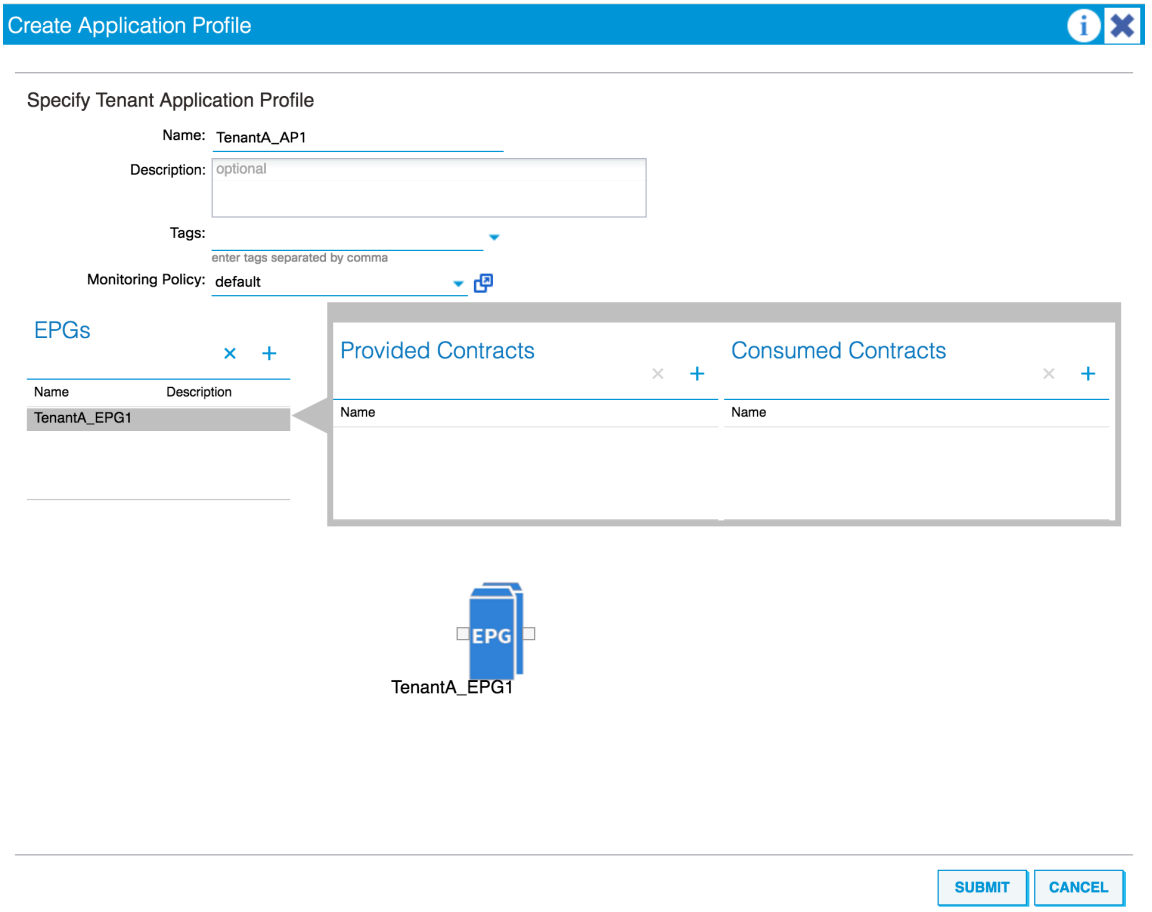


2. Give the EPG a name, and select the bridge domain associated with it:



3. We can click **OK** to be returned to the previous window.

Intra-EPG isolation is designed to prevent endpoints in the same EPG from communicating with each other. Useful if you have endpoints belonging to different tenants in the same EPG



4. We can click **Submit** here and our AP and EPG will be created, or we can specify some contracts to provide, consumer, or both. For the moment, we will just click on Submit and create the contract separately in the next recipe.

How it works…

We can start to see the different components of the ACI fabric starting to merge together now, giving us an understanding of how they all tie in together. A virtual machine (endpoint) can be assigned to an endpoint group, which is tied to an application profile. The endpoint group is associated with a bridge domain, which contains the subnet (or subnets) and, in turn, the bridge domain is linked to a VRF instance. The VRF controls our routing, and all of these components go to make up the tenant.

The tenant is, at the moment, very isolated. If we were to add another tenant into the mix, the two would not be able to communicate. We can permit inter-tenant communication by using contracts. Which is what we will start to set up next.

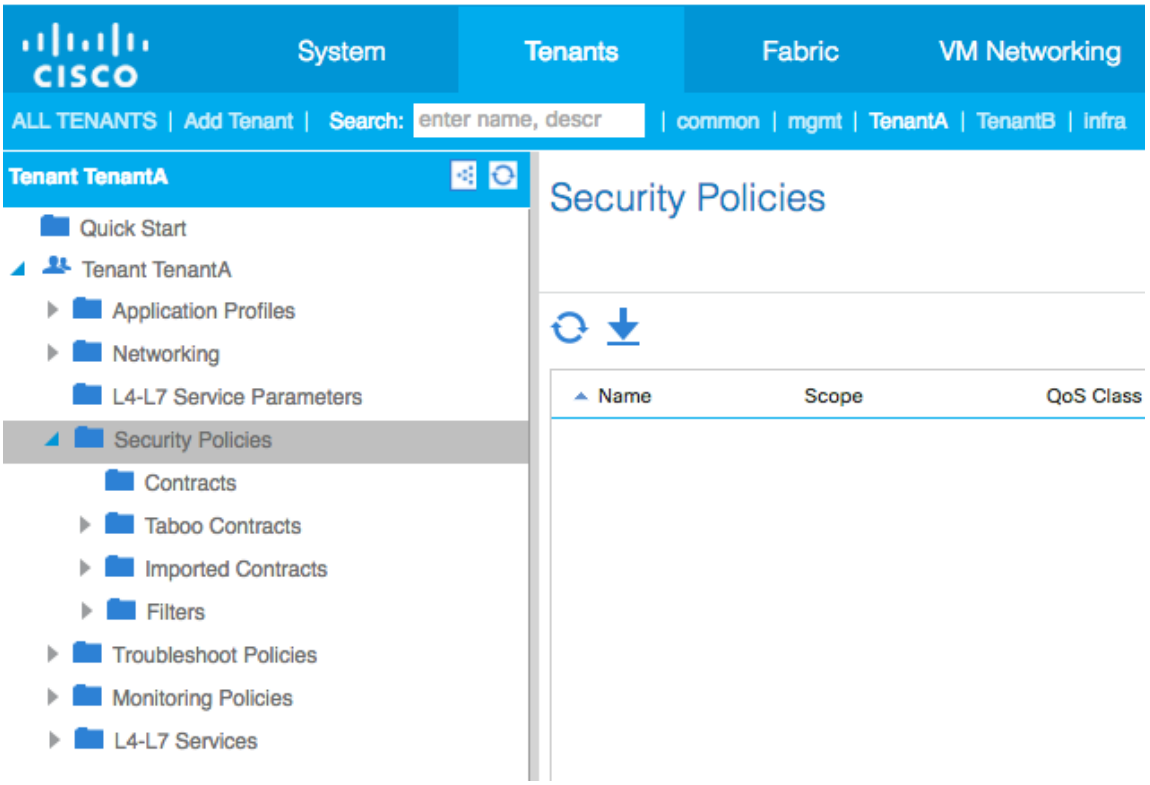
Using contracts between Tenants

Contracts allow EPGs to communicate with each other, according to the rules we set. Contracts can be very granular, including the protocol, port, and direction of the traffic. We do not need a contract for intra-EPG traffic, this is implicitly permitted, but a contract is essential for inter-EPG traffic.

An EPG can be a provider of a contract, a consumer of a contract, or can perform both functions; providing and consuming at the same time. We can also provide or consume multiple contracts simultaneously. Contracts are (to simplify them) access lists. However, they are not bound by the same limitations that access lists are. To read about why contracts are better than access lists, refer to this link: `http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/aci-fundamentals/b_ACI-Fundamentals/b_ACI_Fundamentals_BigBook_chapter_0100.html#concept_0DEE0F8BB4614E3183CD568EA4C259F4`. To try and simplify the definition of provider and consumer, we have two contracts. One opens up HTTP access to a particular destination (it provides), the other permits access from the other EPG to the HTTP server (consuming). We can also be less stringent and have full TCP and UDP access between two EPGs, so would have two contracts and both EPGs would consume one and provide the other, allowing the full bi-directional connectivity.

How to do it…

1. We need to create another Tenant for this recipe. Repeat the previous recipes from this chapter using the following settings: **Name:** TenantB **Bridge Domain Name:** TenantB-BD **VRF Name:** TenantB_VRF **Subnet:** 10.0.1.1/24 **Application Profile Name:** TenantB_AP1 **EPG Name:** TenantB_EPG1
2. This has created another tenant, but at the moment, the two will be unable to communicate. We need to edit the subnets we have created and set them to "Shared between VRFs". Navigate to Tenants > TenantA > Networking > Bridge Domains > TenantA-BD > Subnets > 10.0.0.1/24 and tick the Shared Between VRFs checkbox. Click submit and apply the changes. Repeat the process for the TenantB 10.0.1/24 subnet.
3. We are going to create a very basic contract. TenantA will be the provider and TenantB will be the consumer. We start by selecting the Security Policies option from the left-hand side menu for TenantA:



4. From here, we select "**Create Contract"** from the "Actions" drop down.

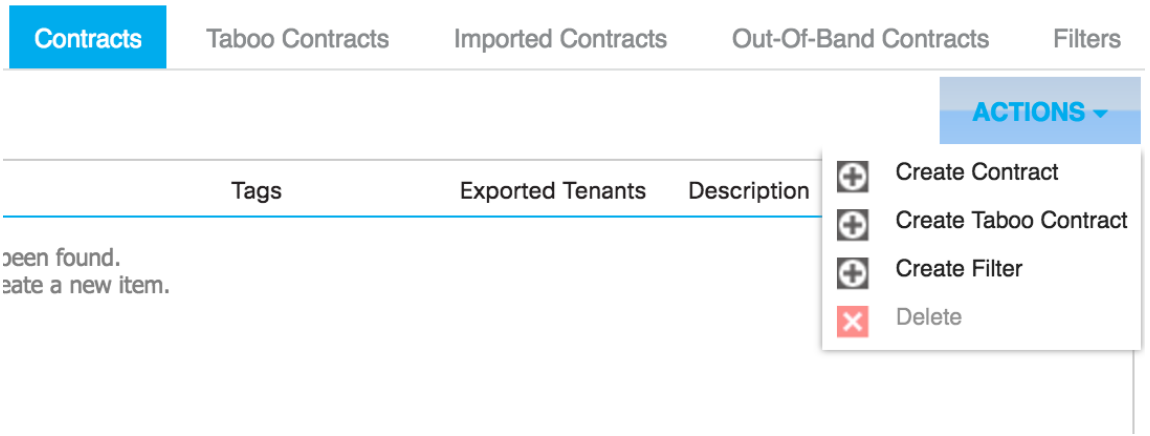


Figure 80.

5. We need to give the contract a name and click on the plus sign to create a new subject of the contract:

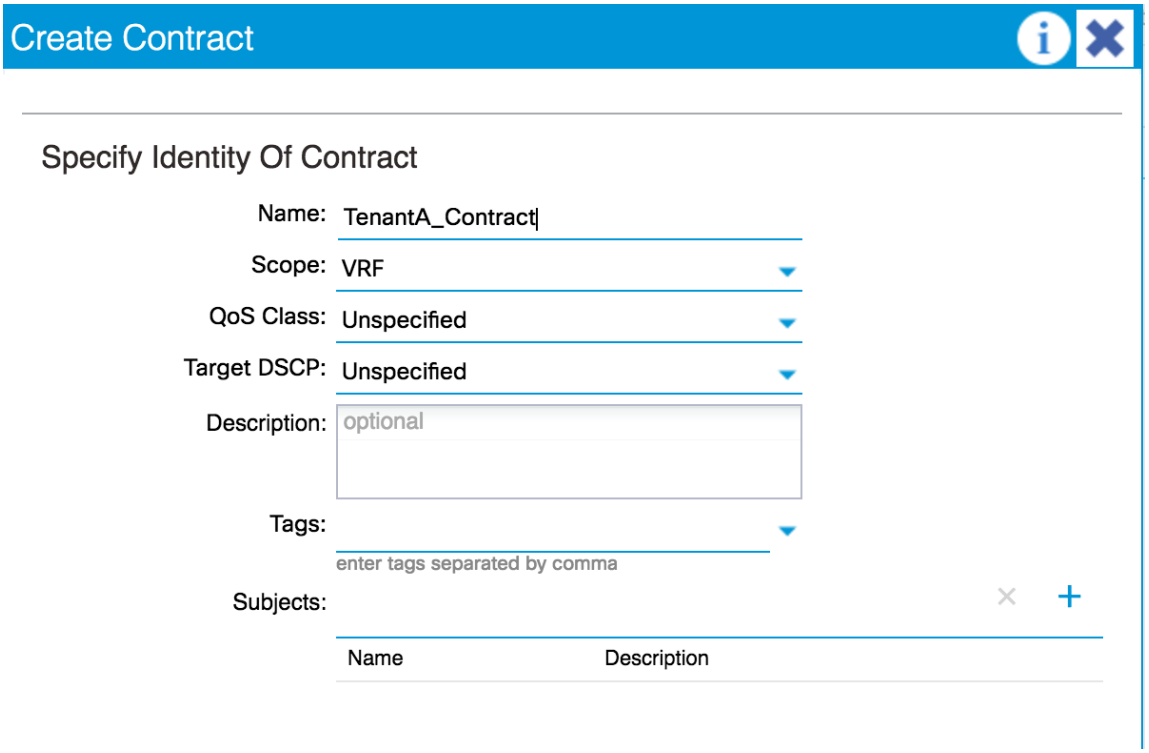
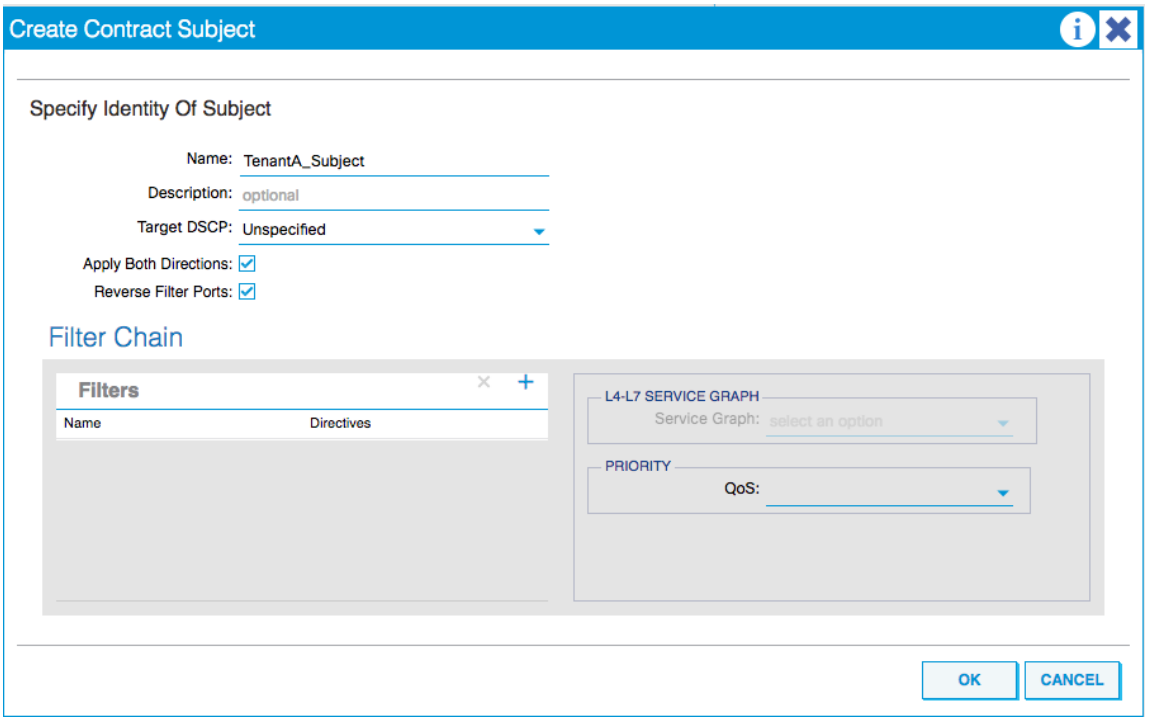
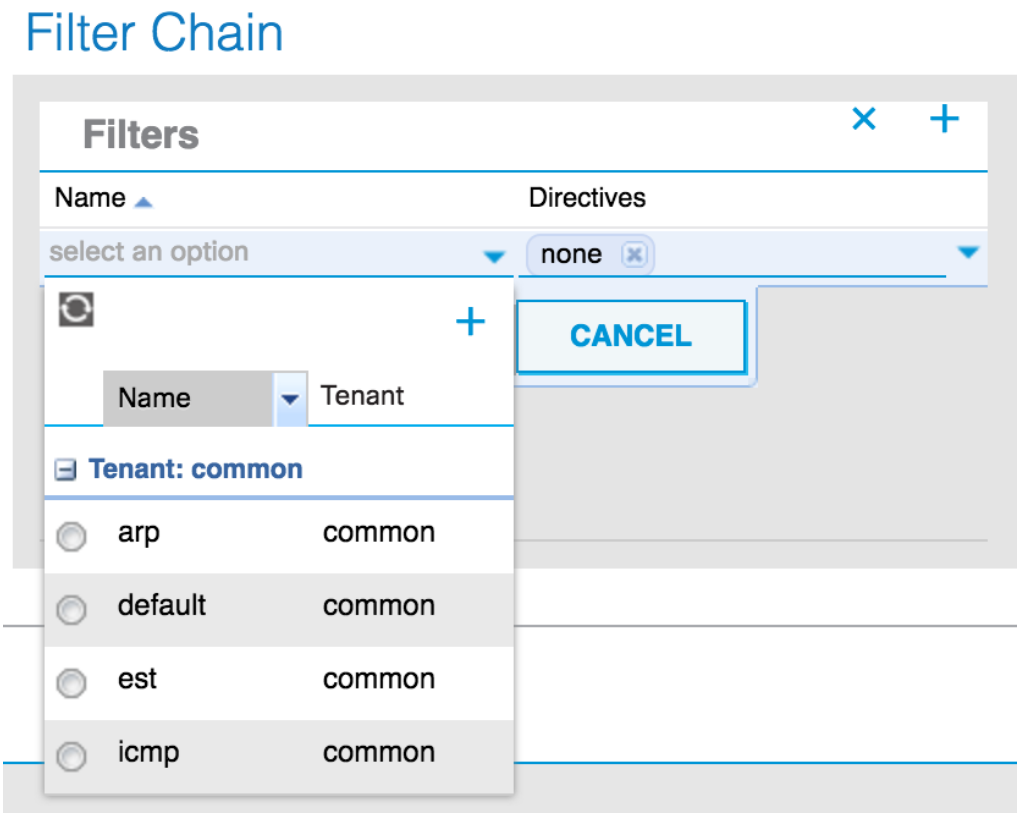


Figure 81.

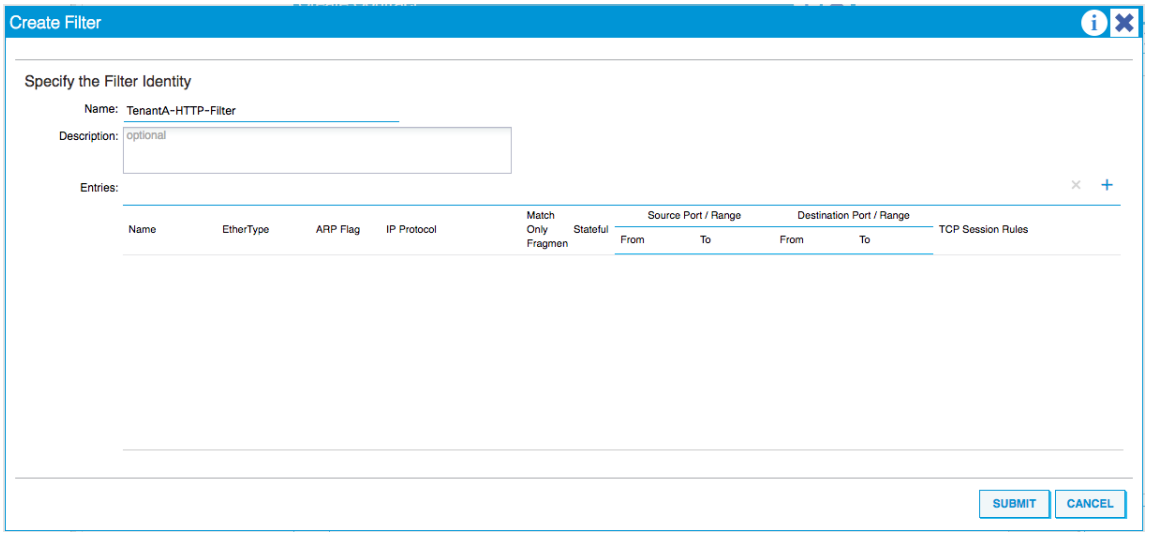
6. In the new window, we need to specify the subject. We assign it a name:



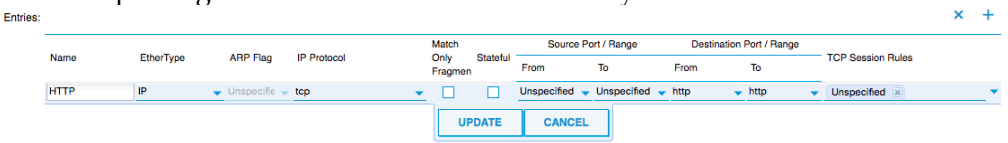
7. The next step is to create a filter chain. Filter chains are where we classify our traffic (according to which attributes between layer 2 and layer 4 we decide upon). Clicking the plus sign next to Filters gives us a list of filters that exist within the "common" tenant.



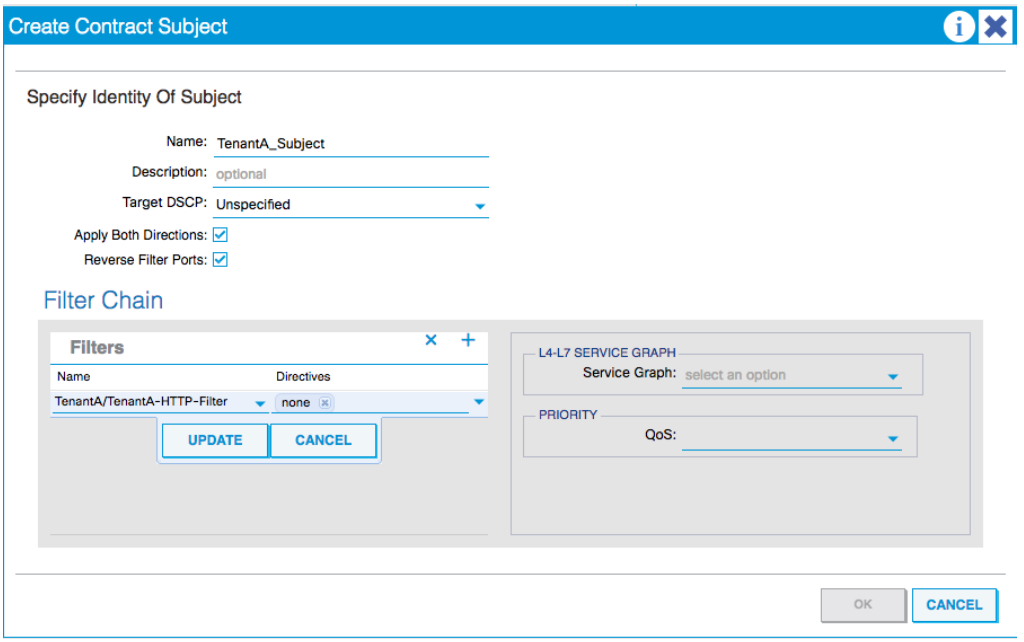
Clicking on the plus sign above the word “Tenant” will allow us to create a custom one.



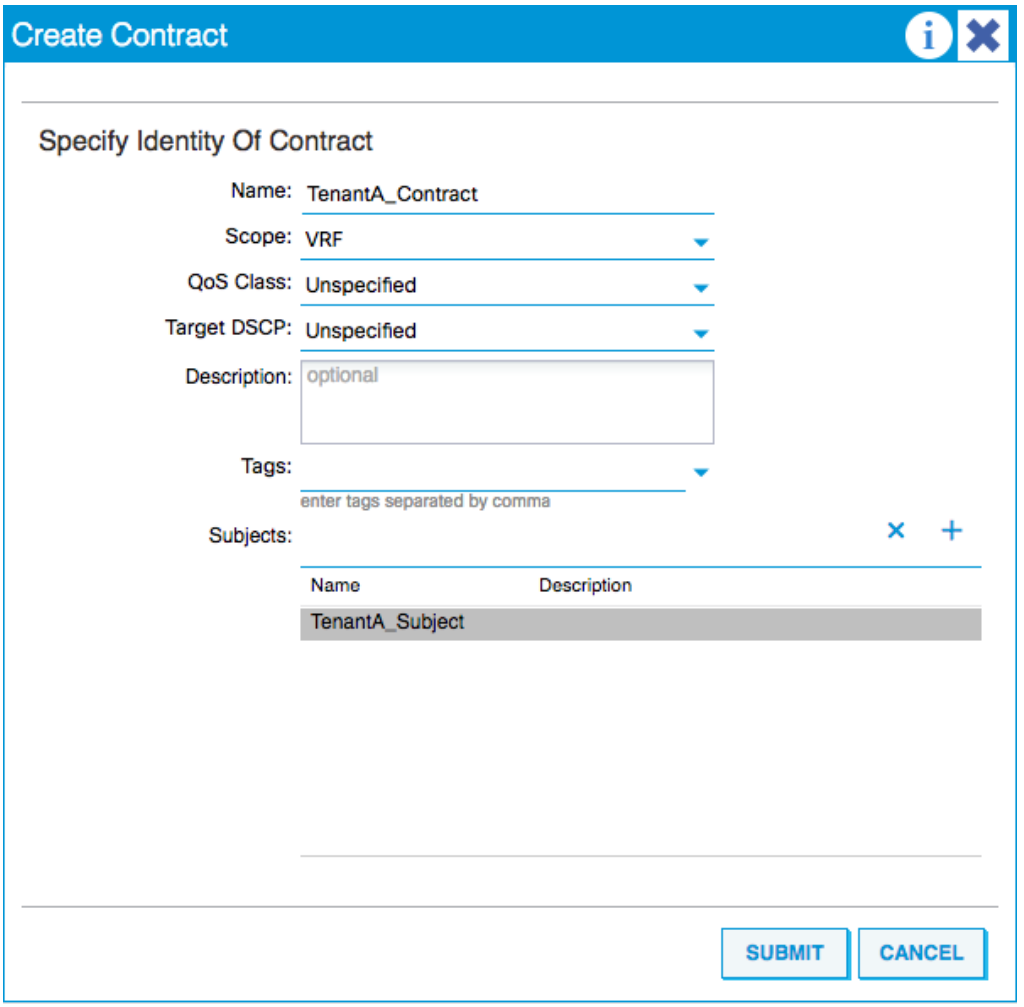
8. Click the plus sign next to Entries to create an entry for HTTP:



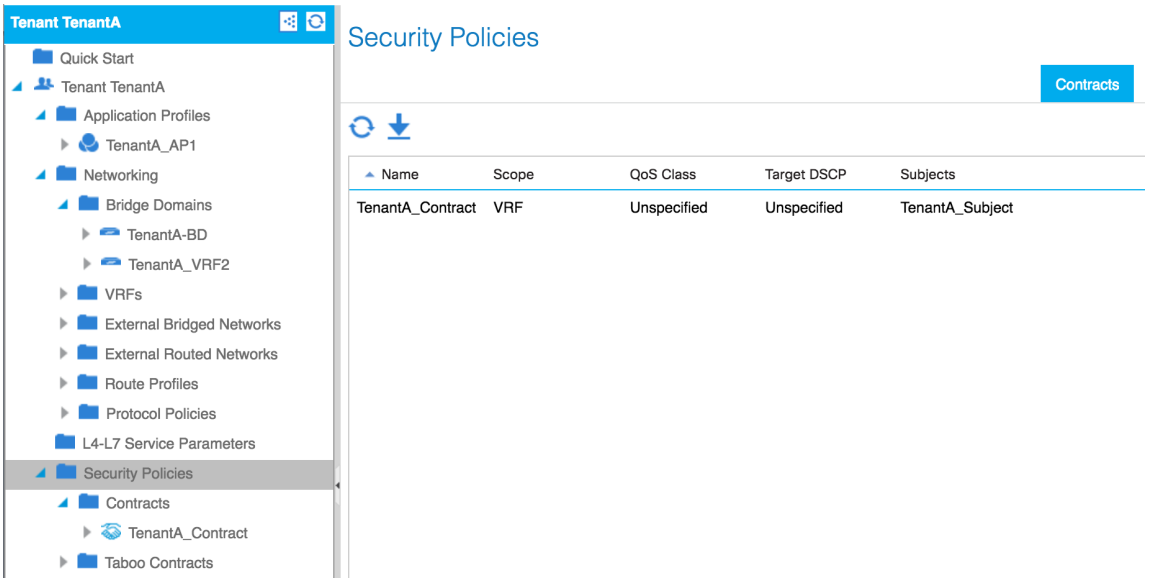
   Name the entry and set the EtherType to “IP”, the IP Protocol to “tcp” and set the destination port range to “http”.
9. Click Update.
10. Click Submit.
11. Back on the Create Contract Subject window, click Update



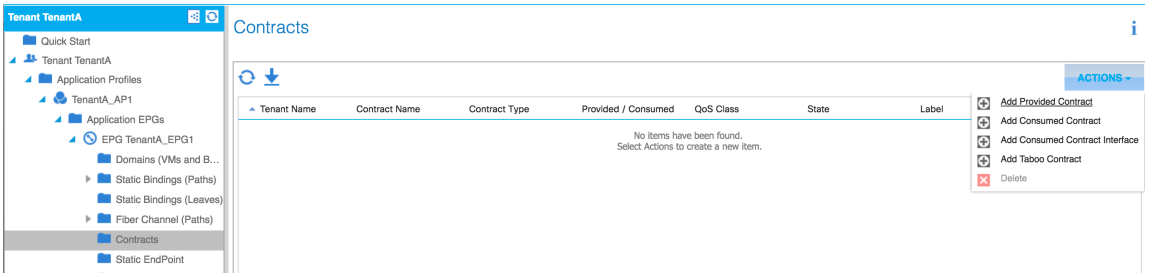
12. Click OK.
13. Click Submit.



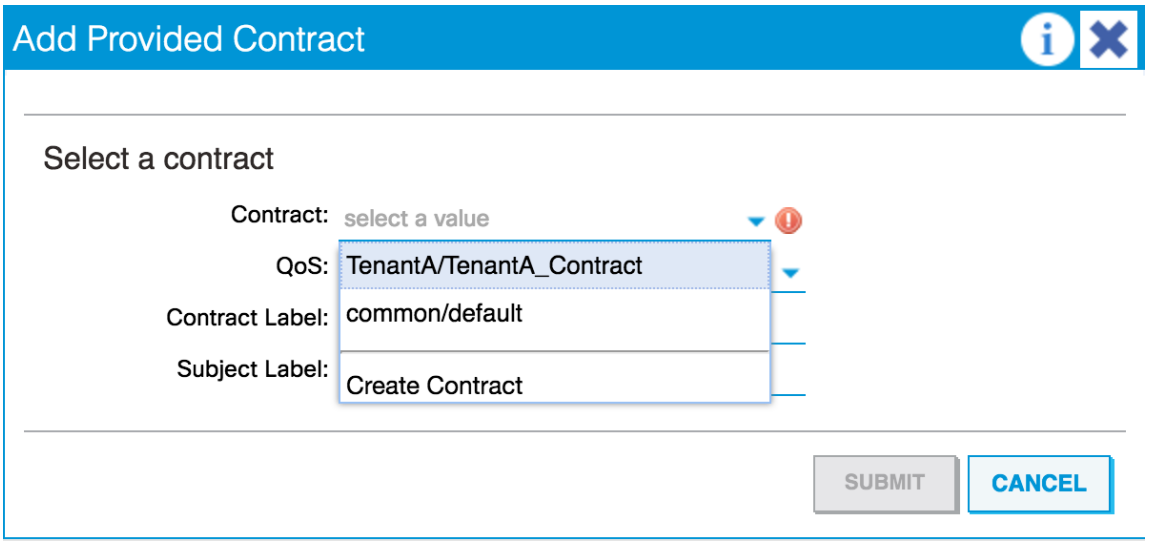
14. Once we click on **Submit**, we can see the contract listed in the security policies.



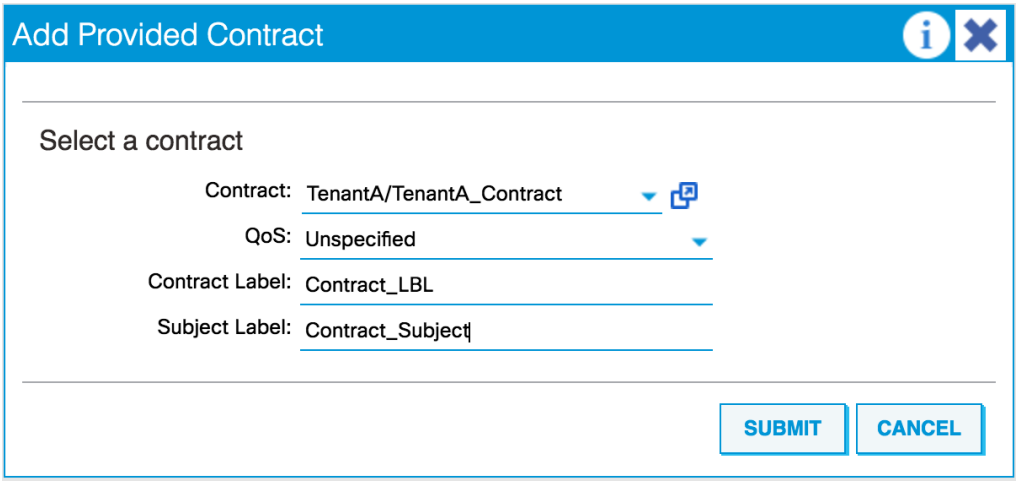
15. The next step is to attach it to the EPG. We do this from the Contracts option under the tenant application profile TenantA > Application profiles > TenantA_EPG1 > Contracts.



16. We click on Actions, then on "**Add Provided Contract"**, and select the contract we previously created.

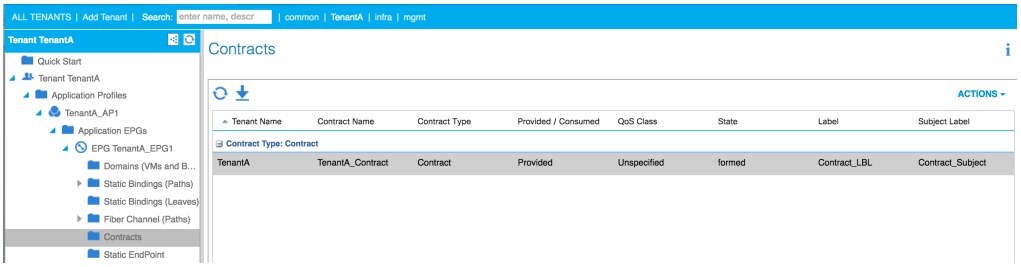


17. We can add contract labels and subject labels.

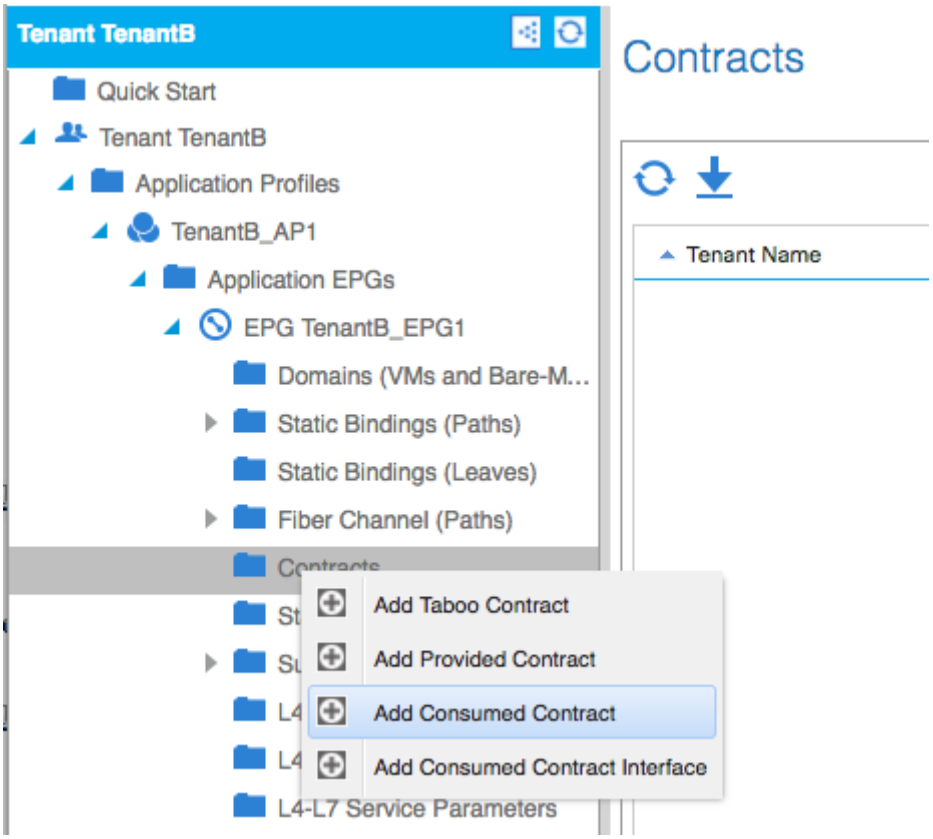


These labels are optional and are used to increase granularity during policy enforcement.

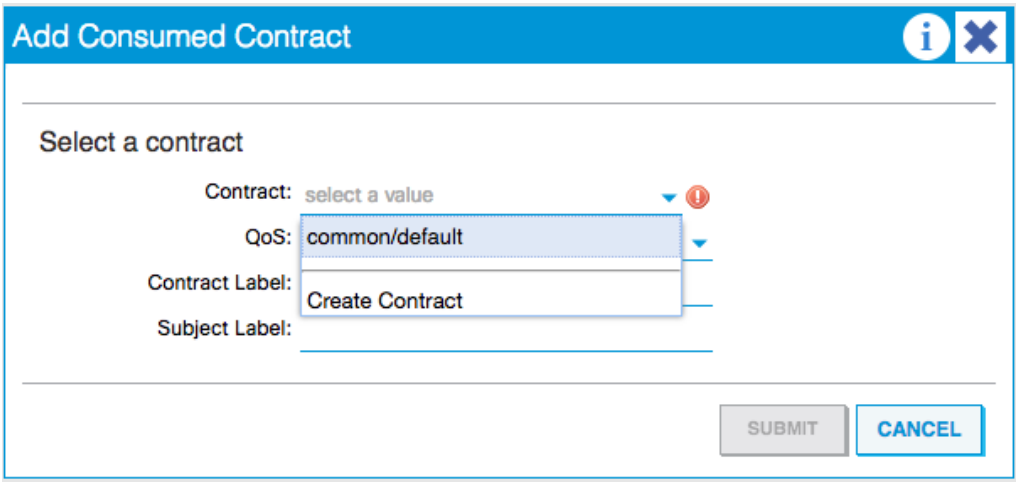
18. Once we hit **submit**, our contract is connected to our EPG.



19. We need to do the same with TenantB, this time setting it as a Consumed contract:

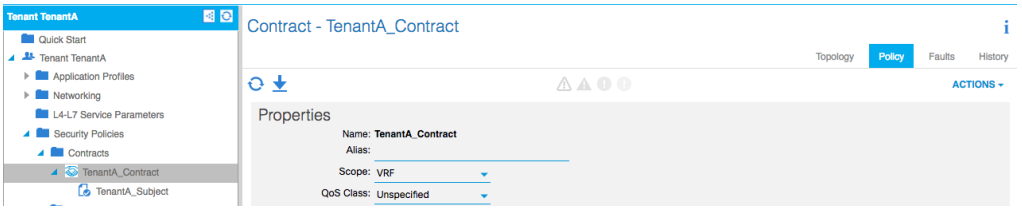


20. If you try and add the previously created contract, you will not find it in the drop-down list.

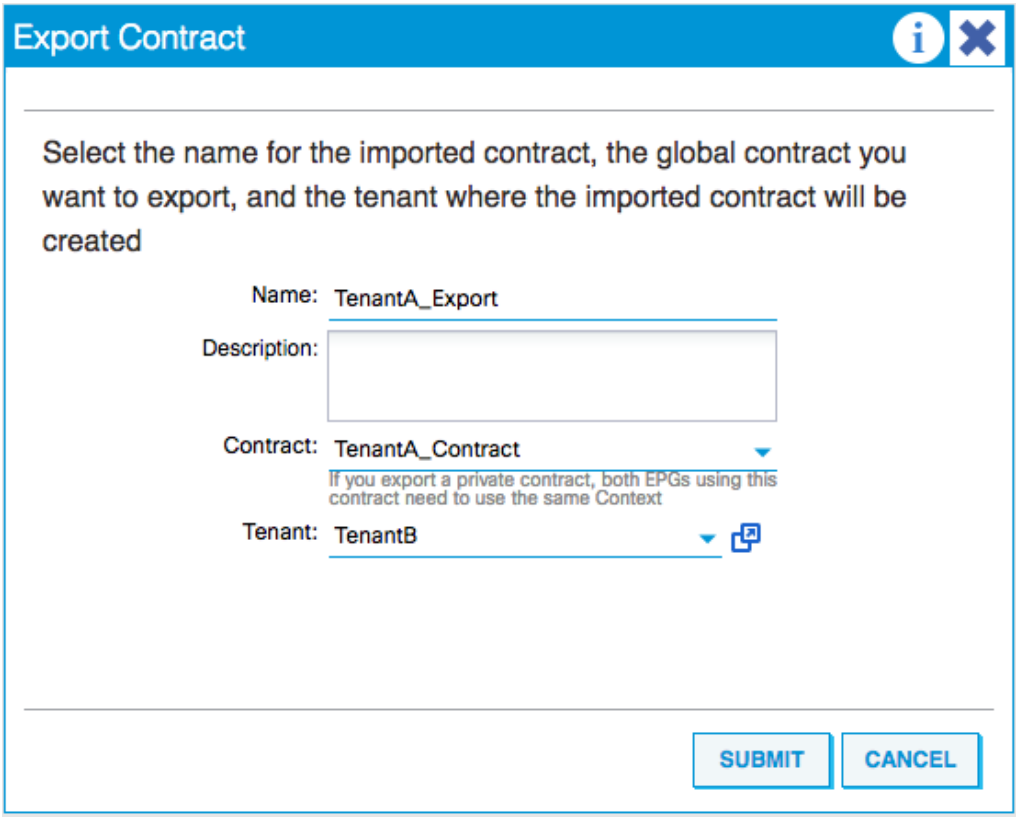


This is because the scope is set to "VRF". We need the scope to be set to "Global" so that other tenants can see it.

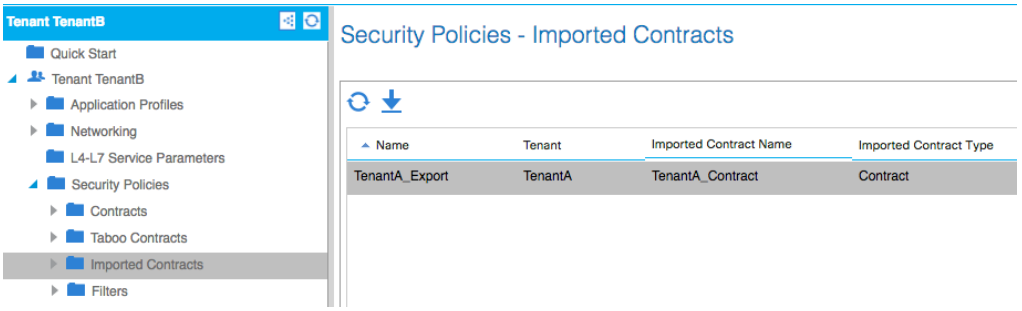
21. Return to TenantA, and navigate to Security Policies > Contracts > TenantA_Contract. Click the Policy tab on the right-hand side.



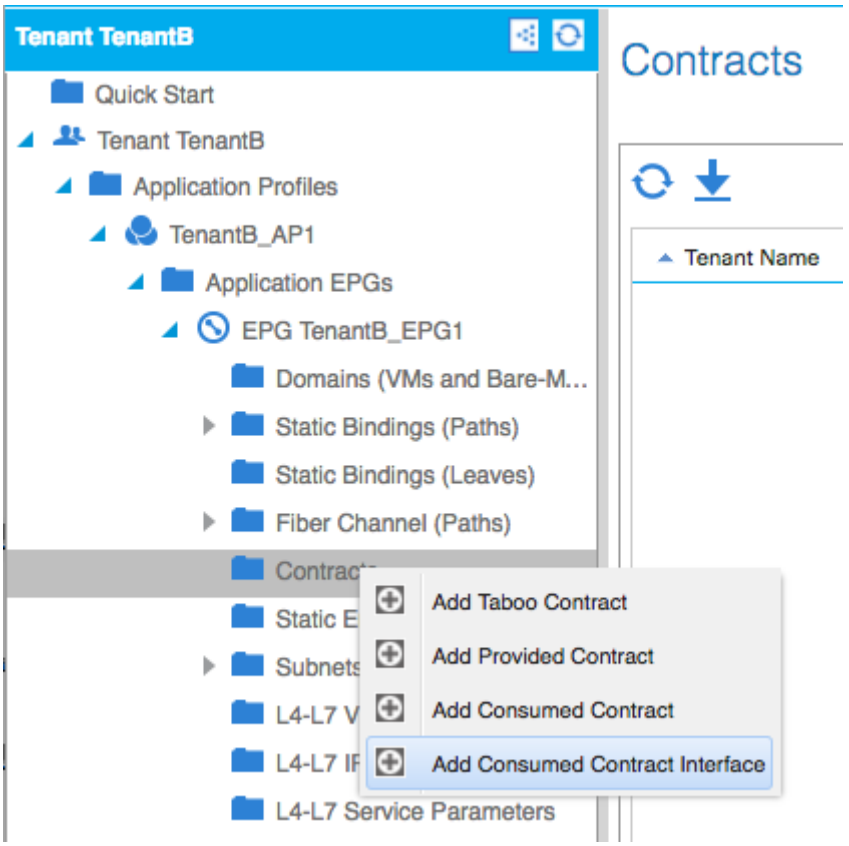
22. Change the scope to "Global" and click Submit at the bottom right-hand corner. Click on Submit Changes.
23. We need to export the contract now. From TenantA > Security Policies, right click on Contracts and select "Export Contract".
24. Set the name for the export, select the contract created earlier and select TenantB.



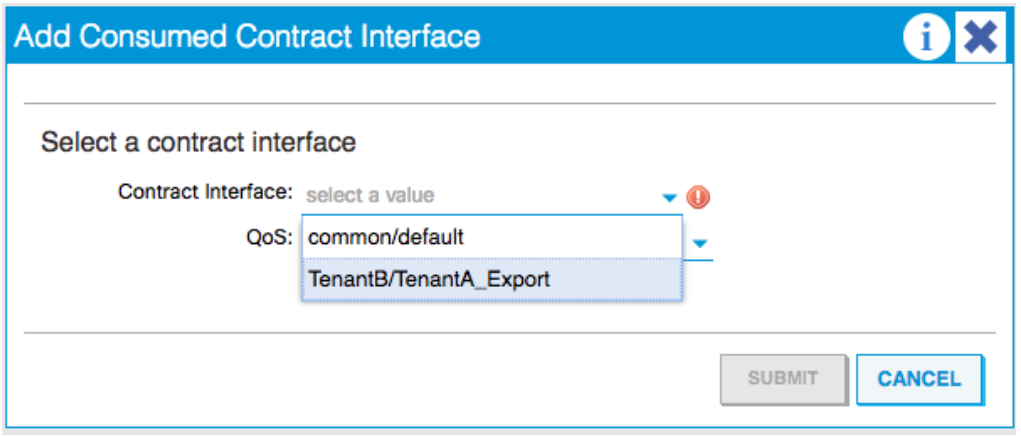
25. Click Submit.
26. We should now be able to see the exported contract being imported into TenantB.



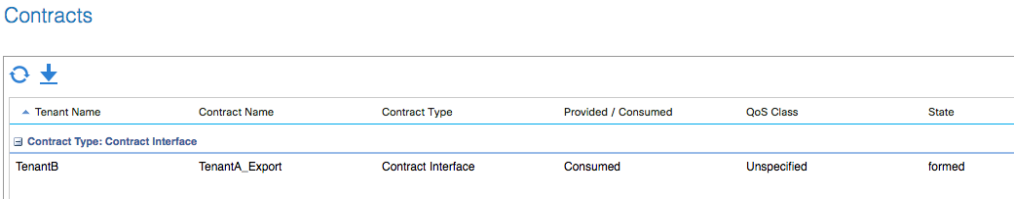
27. Navigate to Contracts, right-click on it ans select “Add Consumed Contract Interface”.



28. Select the TenantB/TenantA_Export.



29. Click Submit.
30. We can now see the contract listed.



How it works…

We have created a very basic regular contract to provide to another tenant. There are other types of contracts we can create. Taboo contracts are used to deny and log traffic. Like conventional access control lists to deny traffic, these need to come first. An example would be where we are permitting a large number of ports and wanted to deny one or two particular ports; we would do this with a taboo contract to deny the traffic, created before the regular contract permitting the entire range.

In the above recipe, we added a couple of labels. Labels allow us to classify what objects can talk to each other. Label matching is performed first, and if no label matches, then no other contract or filter information is processed. The label-matching attribute can be all, none, at least one or exactly one.

While filters specify the fields to match on between layer 2 and layer 4, the subject can specify the actual direction of the traffic (unidirectional or bidirectional).

The contract we created was not that exciting but offers a building block onto which we can add more filters.

Creating Filters

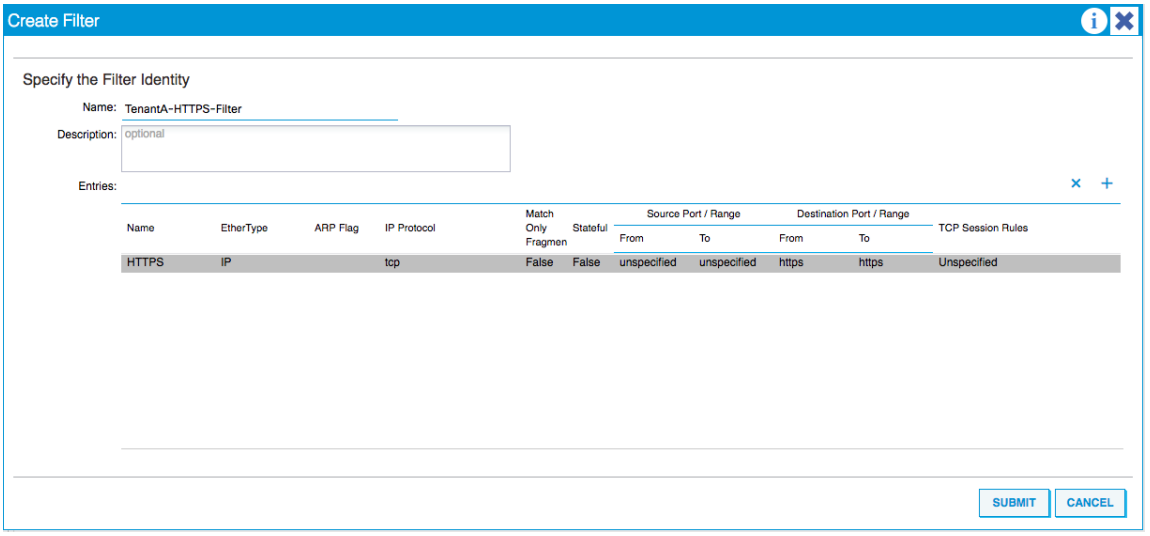
In this recipe, we will create a filter and apply it to the contract we created previously.

How to do it…

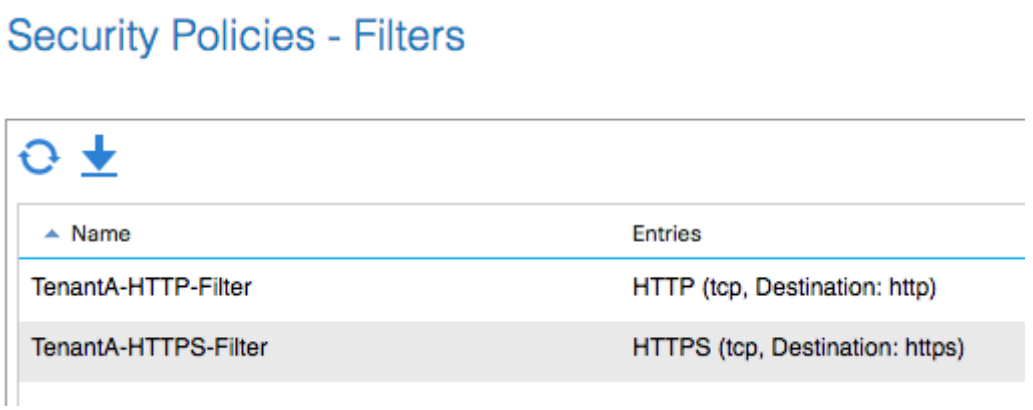
1. From the TenantA Security Policies menu, select the **Filters** option. Click on

Actions, and then click on **Create Filter.**

2. Give the filter a name, description (if you want to), and then click on the plus sign. The entries in the filter must have a name, but after that, you can be as permissive or restrictive as you need. Here we have created a filter called “https”, which sets a filter on the layer 3 EtherType of “IP”, the layer 4 IP protocol of “tcp” and the layer 7 protocol of https (as the destination port range). This follows the same steps as the previous recipe.

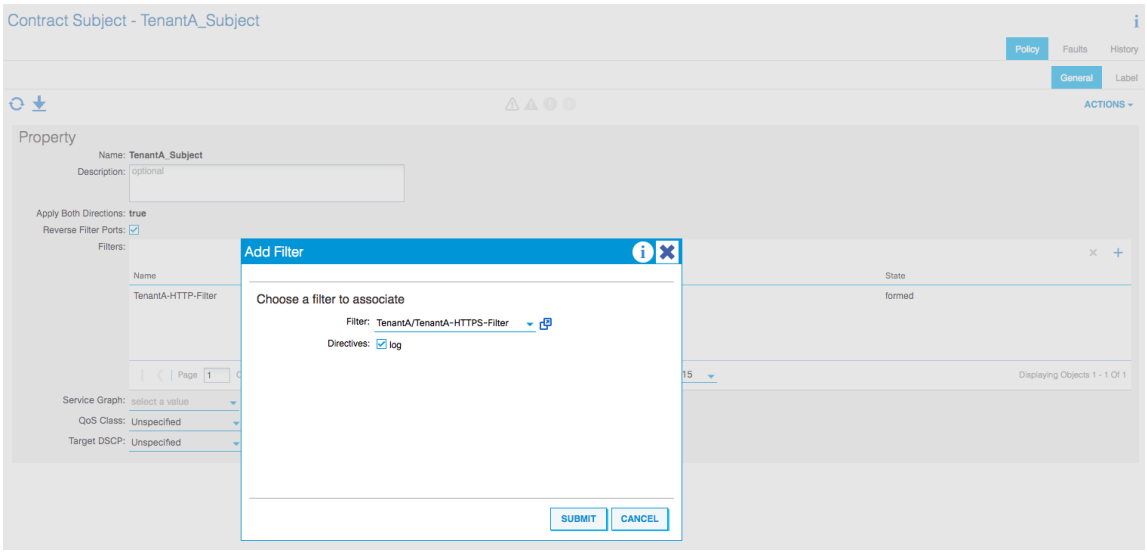


3. We can now click on **Submit**, and we can see the filter listed under the tenant’s filters:

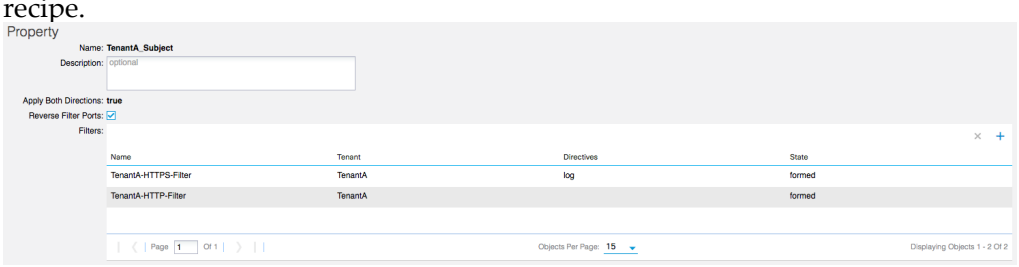


4. To attach this filter to the contract, we need to select the contract we created earlier, then, under the **Filters** window, click on the plus sign.

5. In the window that pops up, we can select the new filter from the drop-down menu, we can choose to log the activity, and click on **Submit**:



6. Finally, we see our filter sitting alongside the default filter from the previous recipe.



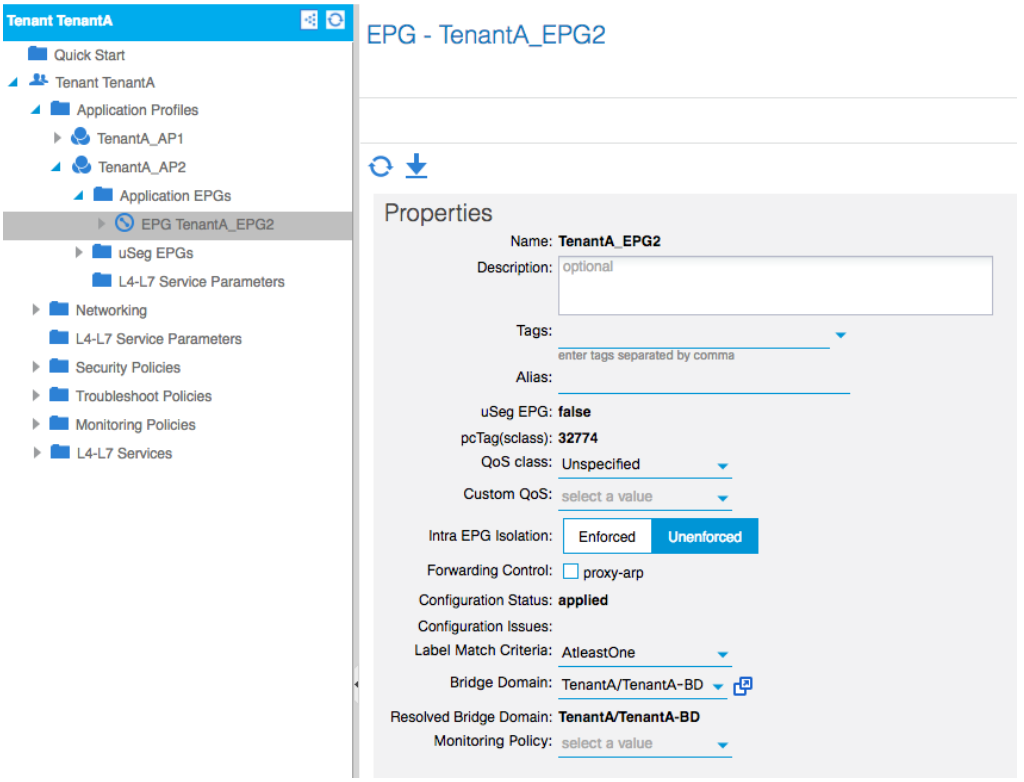
Configuring contracts between different tenants is the harder of the options. By contrast, configuring contracts between EPGs in the same tenant takes much fewer steps, as do management contracts. We will look at these next. This will also help show how contracts work so much more nicely than access-lists as you scale the number of APs, EPGs and tenants.

Creating contracts within Tenants

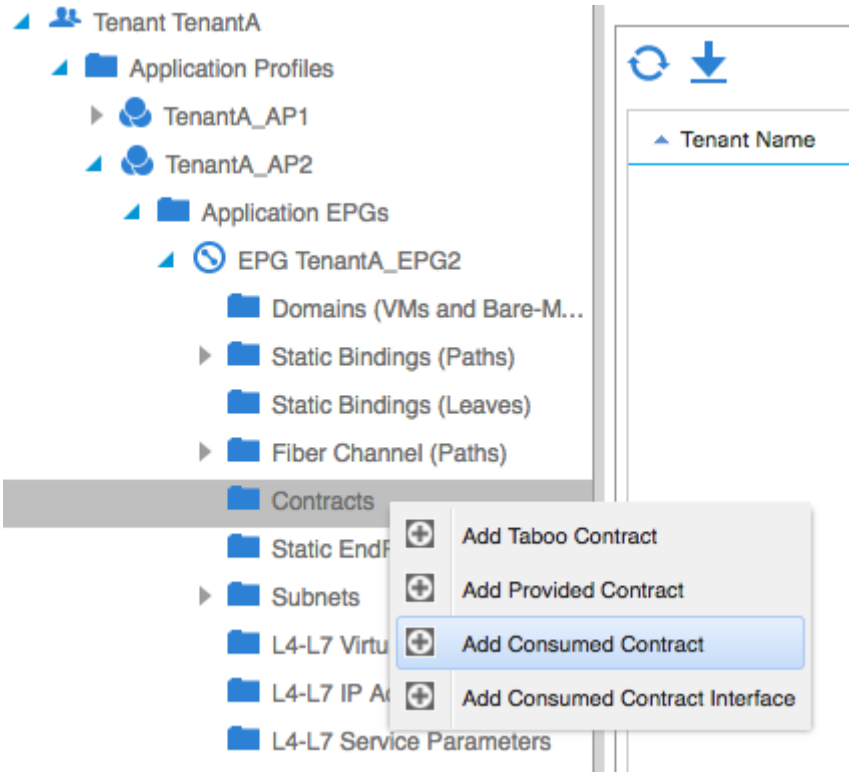
We will now create another Application Profile and EPG within TenantA, and provide a contract for the other to consume.

How to do it…

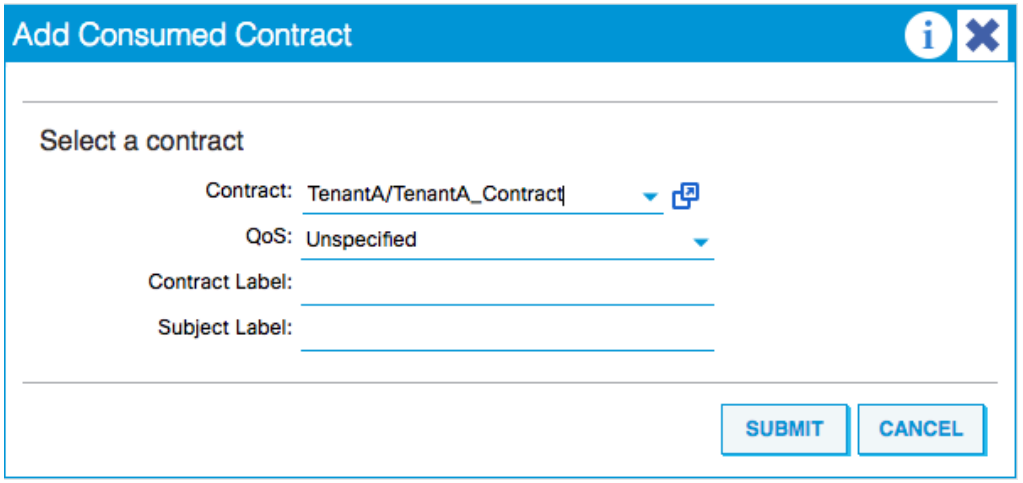
1. Create another Application Profile and EPG within TenantA. The end result should look like this:



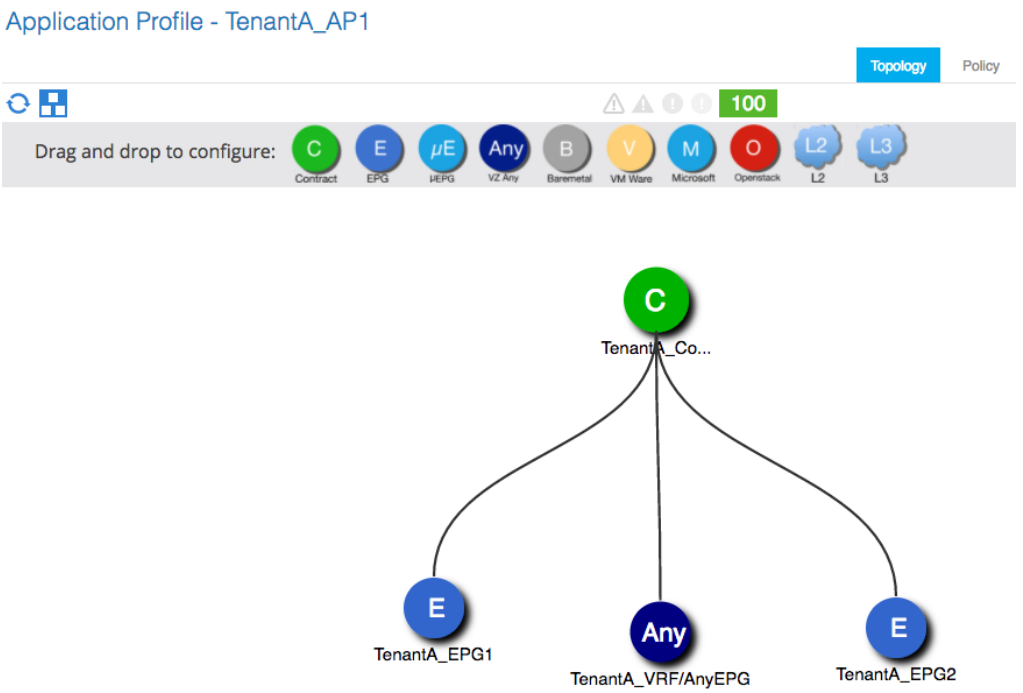
2. Expand the new EPG, right-click Contracts and select “Add Consumed Contract”.



3. From the drop-down list, select the contract.



4. Click Submit. We can get a visual representation of this as well.



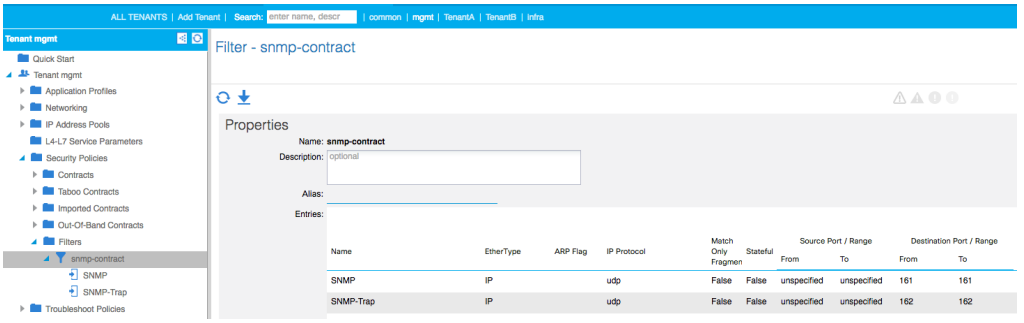
The same method would be used if we had another EPG within the same AP.

Creating Management contracts

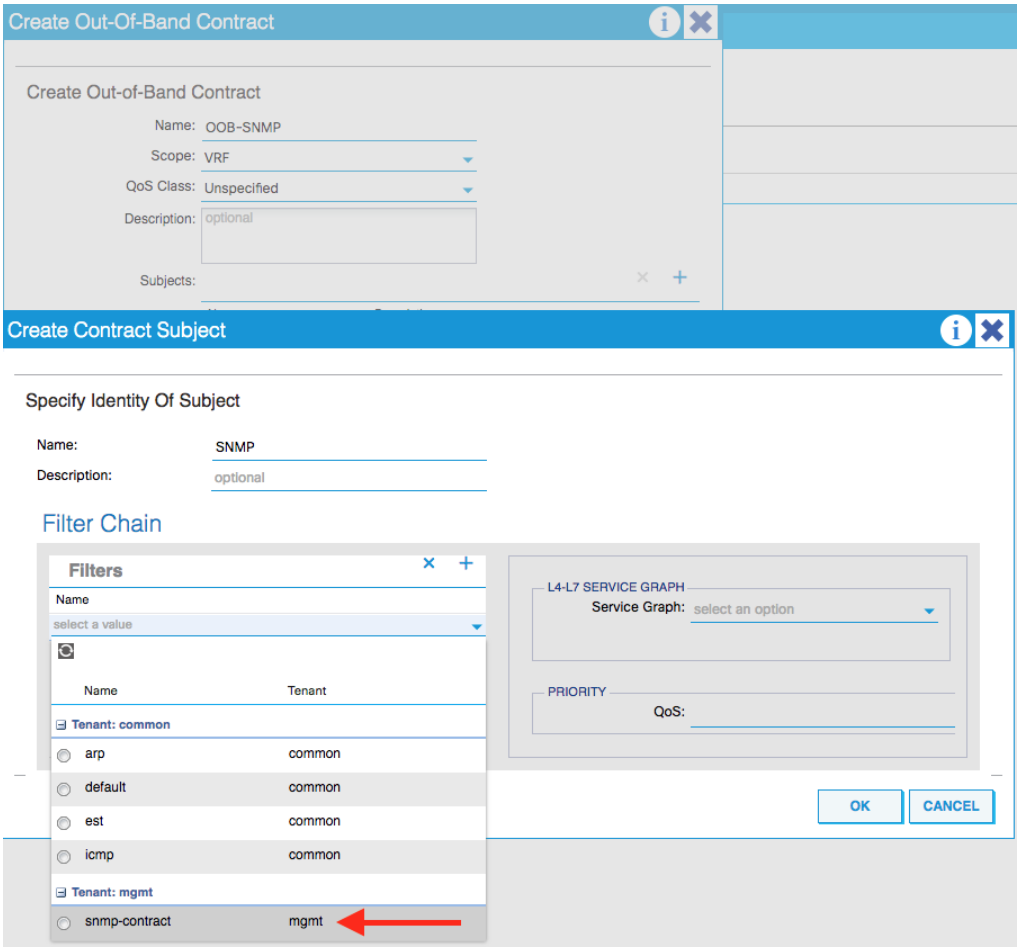
The final contract we are going to create is one in the mgmt tenant. This one will allow SNMP traffic between the APIC and the SNMP software, which we will be setting up in Chapter 8.

How to do it…

1. Create a filter (snmp-contract) in the mgmt tenant (Tenants > mgmt > Security Policies > Filters).
2. Create two entries, permitting UDP ports 161 and 162).



3. Right-Click on Out-Of-Band Contracts and select “Create Out-Of-Band Contract”.
4. Name the contract (OOB-SNMP) and click the plus sign next to Subjects. Select the snmp-contract created previously.



5. Click Update.
6. Click OK.
7. Click Submit.

How it works…

This is an out-of-band contract which we will be needing later on in the book. Earlier versions of the ACI software did not require this contract, but newer ones do. The contract is permitting traffic to the UDP ports used by SNMP and for the SNMP trap notifications.

3
Hypervisor Integration (and other 3rd Parties)

In this chapter, we will cover the following recipes:

- Installing device packages
- Creating VMM Domains and integrating VMWare
- Associating vCenter Domains with a Tenant
- Deploying the AVS
- Discovering VMWare Endpoints
- Adding Virtual Machines to a Tenant
- Tracking ACI Endpoints
- Integrating with A10
- Deploying the ASAv
- Integrating with OpenStack
- Integrating with F5
- Integrating with Citrix NetScaler

Introduction

ACI is highly extensible. Through device packages, we can add several different devices to our environment, which is referred to (in ACI terms) as “service insertion.”

The packages themselves are small zip files. Some require certain permissions from the manufacturer before you can download them (such as Citrix), whereas others just require registering your email address (A10, for example).

Inside the zip file, we have some different files. Taking the A10 APIC package as the example here, we have five Python files, one XML file and one GIF image in a folder called "Images." The zip file's size is a mere 65KB. The XML file is, for most, going to be the easiest file to understand. This file is called "device_specification.xml." It starts with defining the vendor (vnsMDev), along with a package name (which is one of the python scripts) and the version details (vmsDevScript):

```
<vnsDevScript name="A10" packageName="device_script.py" ctrlrVersion="1.1"
minorversion="1.0" versionExpr="4.[0-9]+.[-A-Za-z0-9]*"/>
```

Next, we define the device profiles (vnsDevProf), whether they are virtual or physical devices, and the number of Ethernet interfaces that they have (note that I have truncated the output):

```
<vnsDevProf name="vThunder" type="VIRTUAL" context="single-Context"
pcPrefix="trunk ">
   <vnsDevInt name="ethernet 1" />
   <vnsDevInt name="ethernet 2" />
   <!—truncated ->
   <vnsDevInt name="ethernet 7" />
   <vnsDevInt name="ethernet 8" />
</vnsDevProf>
<vnsDevProf name="vThunder-ADP" type="VIRTUAL" context="multi-Context"
pcPrefix="trunk ">
   <vnsDevInt name="ethernet 1" />
   <vnsDevInt name="ethernet 2" />
   <!—truncated ->
   <vnsDevInt name="ethernet 7" />
   <vnsDevInt name="ethernet 8" />
</vnsDevProf>
<vnsDevProf name="Thunder" type="PHYSICAL" context="single-Context"
pcPrefix="trunk ">
   <vnsDevInt name="ethernet 1" />
   <vnsDevInt name="ethernet 2" />
   <!—truncated ->
   <vnsDevInt name="ethernet 19" />
   <vnsDevInt name="ethernet 20" />
</vnsDevProf>
<vnsDevProf name="Thunder-ADP" type="PHYSICAL" context="multi-Context"
pcPrefix="trunk ">
   <vnsDevInt name="ethernet 1" />
   <vnsDevInt name="ethernet 2" />
   <!—truncated ->
   <vnsDevInt name="ethernet 19" />
   <vnsDevInt name="ethernet 20" />
</vnsDevProf>
```

After the interface declarations, we define some interface labels for “external” and “internal” (vnsMIfLbl) and some credentials (vnsMCred and vnsMCredSecret). The first big section we get to is next, which is the cluster configuration (vnsClusterConfig). This section covers core functionality, such as time and DNS, hostname, interface numbering, IPv4 and IPv6 functionality and NTP. We then move to vnsMDevCfg, which is for network interface settings, including Virtual Router Redundancy Protocol(VRRP).

The “vnsMFunc” tag takes up the bulk of the XML file. These are device-specific entries, so the contents of this tag will vary considerably between the different vendors. However, they must all follow the same schema.

The final tags are vnsComposite (comparisons between two values, such as “on” or “off,” “tcp” or “udp,” “Yes” or “No) and vnsComparisons (match a-z and A-Z, or match 0-9, is it an IP address or subnet mask?).

```
<vnsComposite name="TrueFalse" comp="or">
   <vnsComparison name="True" cmp="eq" value="true"/>
   <vnsComparison name="False" cmp="eq" value="false"/>
</vnsComposite>
<vnsComposite name="EnableDisable" comp="or">
   <vnsComparison name="enable" cmp="match" value="enable"/>
   <vnsComparison name="disable" cmp="match" value="disable"/>
</vnsComposite>
<vnsComposite name="onOff" comp="or">
   <vnsComparison name="on" cmp="match" value="on"/>
   <vnsComparison name="off" cmp="match" value="off"/>
</vnsComposite>
<!-- Basic comparison Objects -->
<vnsComparison name="isAlpha" cmp="match" value="[a-zA-Z]+"/>
<vnsComparison name="isNumber" cmp="match" value="[0-9]+"/>
<vnsComparison name="isIPAddress"
               cmp="match"

value="([01]?dd?|2[0-4]d|25[0-5]).([01]?dd?|2[0-4]d|25[0-5]).([01]?dd?|2[0-
4]d|25[0-5]).([01]?dd?|2[0-4]d|25[0-5])"/>
<vnsComparison name="isIPMask"
              cmp="match"

value="([01]?dd?|2[0-4]d|25[0-5]).([01]?dd?|2[0-4]d|25[0-5]).([01]?dd?|2[0-
4]d|25[0-5]).([01]?dd?|2[0-4]d|25[0-5])"/>
```

We also have tags that set the fault codes (vnsMDfcts) and function profiles (vnsAbsFuncProfContr). The function profiles are, again, device specific, and for the A10, specify whether the device is a web server or a web server with high availability.

While we, as engineers, do not need to be concerned with what the contents of these XML

files are, they do serve as a good reminder of the declarative nature of ACI. The XML files are all based on a common schema, and if the vendor can fit what they need around this schema, then the appliance should run very happily within the ACI framework.

For a more in-depth look at device packages, refer to the following link:`http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/L4-L7_Device_Package_Development/guide/b_L4L7_Package.html`

Let's find out how to add a device package.

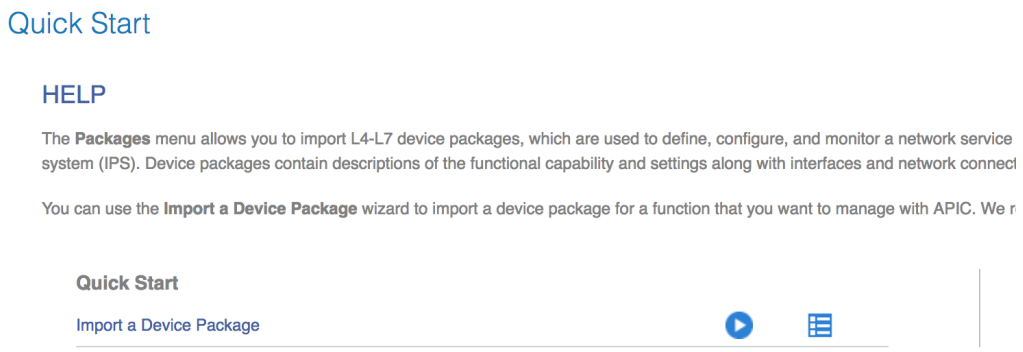
Installing device packages

Installing a device package from the GUI is very simple.

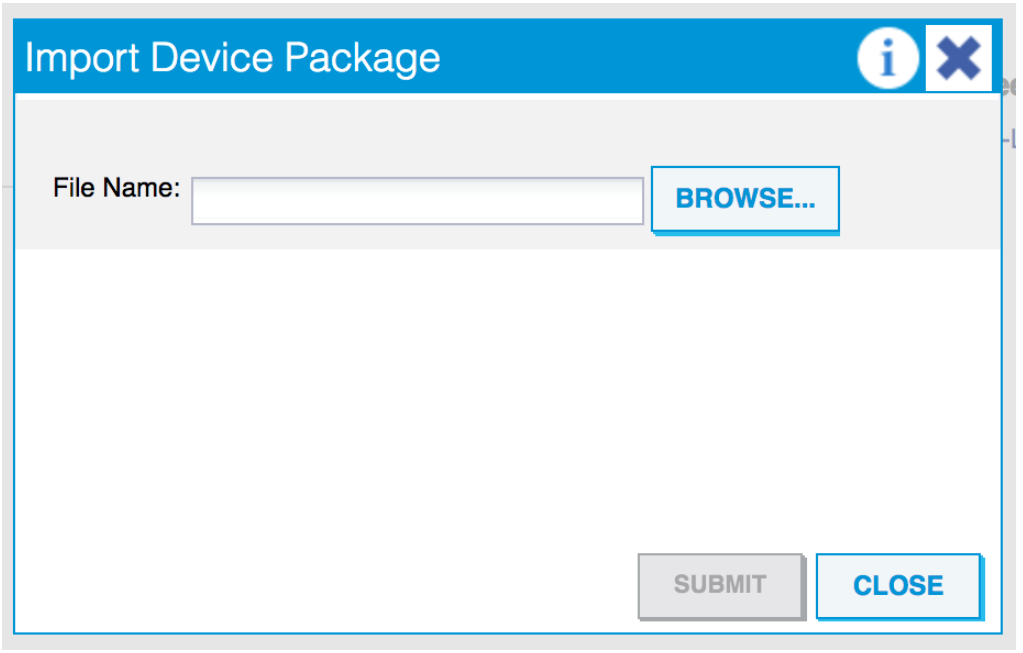
How to do it…

1. From the L4-L7 Services menu, click on Packages:

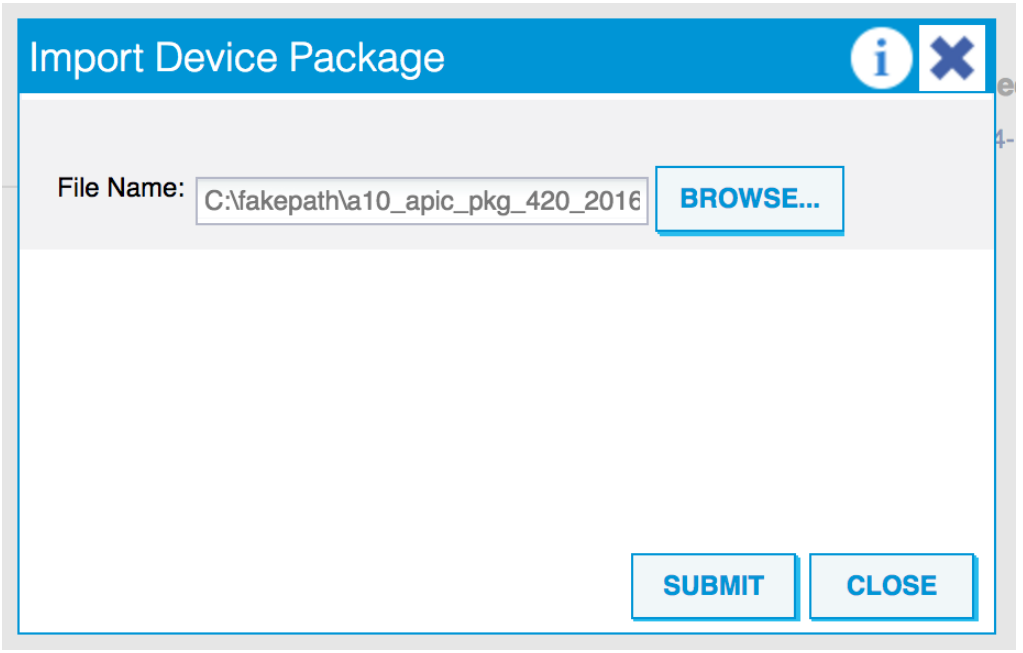
2.
3. The Quick Start menu gives us one option; "Import a Device Package."



4.
5. Click on this link to bring up the file open dialog box:



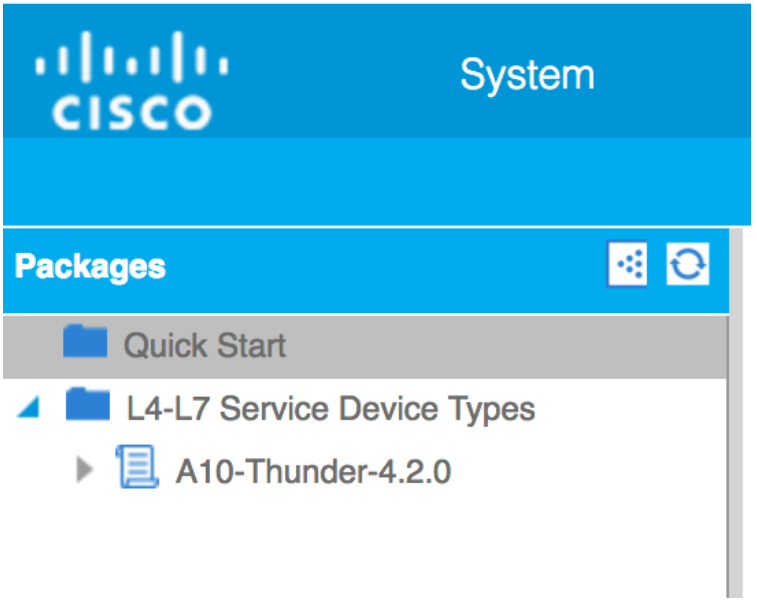
6.
7. Click on browse, and select the zipped package file you want to import. Do not extract the files.

8. 

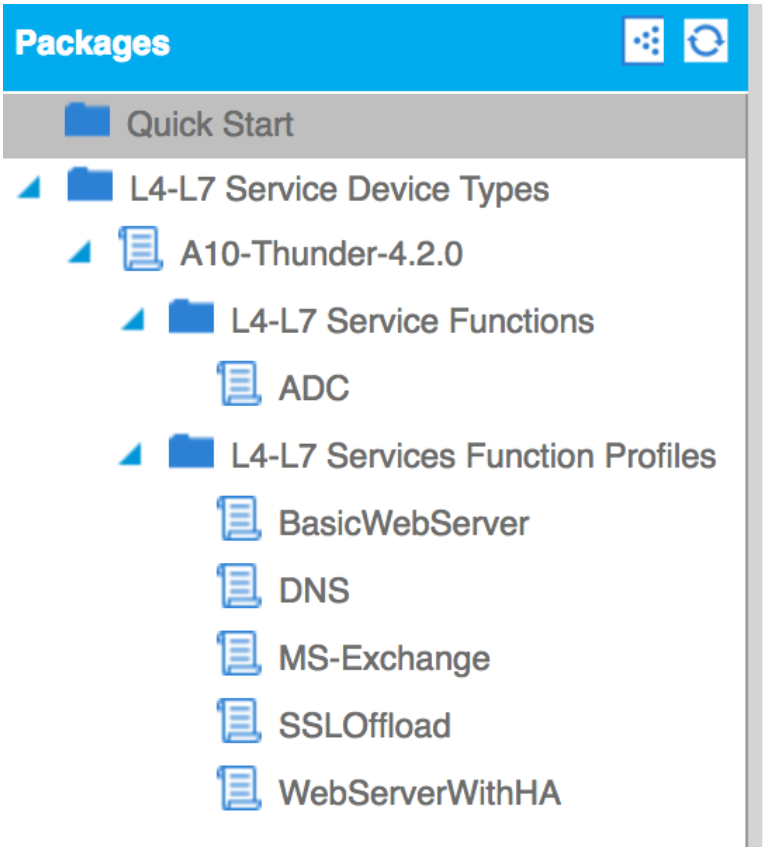
9. Click **Submit**. You will see another message briefly appear as the file is uploaded to the APIC.

How it works…

If we look at "L4-L7 Service Device Types", we can see the newly added device package.



If we expand out the package, we can see a number of options; these should be familiar if you have looked through the packages XML file.



We won't be going through these settings now, as we will look at A10 devices later in this chapter.

There's more…

Here is a link to a list of device packages (some require specific permissions to download – such as ASA):

```
http://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization
/application-centric-infrastructure/solution-overview-c22-734587.html
```

Next, we will move on to integrating VMWare with ACI.

Creating VMM domains and integrating VMWare

ACI uses Virtual Machine Manager (VMM) domain profiles to facilitate communication between virtual machine controllers and the ACI fabric. There are a handful of components that make up a domain, and these are:

- Virtual Machine Manager Domain Profile
- EPG Association
- Attachable Entity Profile Association
- VLAN Pool Association

The Virtual Machine Manager Domain Profile groups VM controllers together. Within this are the components "Credential" for connecting to the VM controller and the "Controller" which specifies how to connect to the VM controller.

The EPG Association allows the APIC to push endpoint groups into the VM controller as port groups and also permits the EPG to span across several VMM Domains.

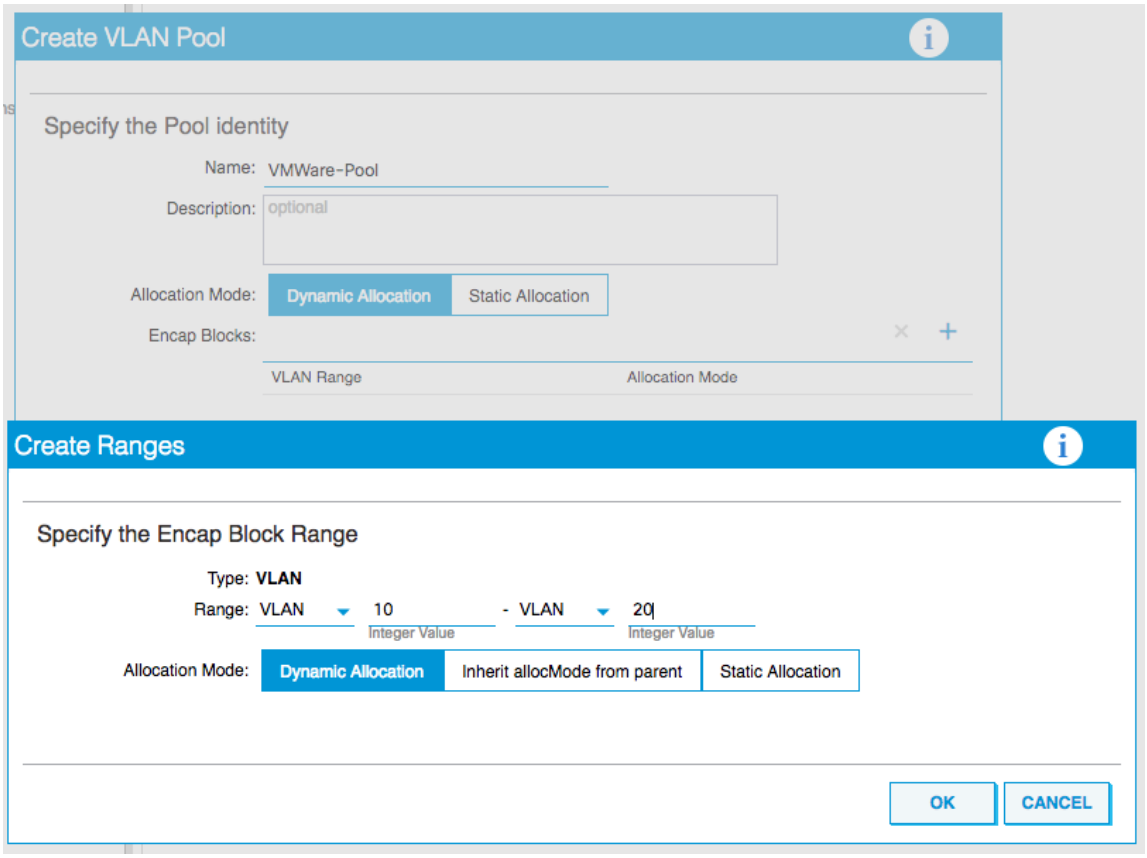
The Attachable Entity Profile Association associates a VMM domain to the physical network. Here, we use an **attachable entity profile** (**AEP**), which is a network interface template, to set policies on leaf switch ports.

Finally, the VLAN pool association specifies the VLAN ID, or range of IDs, for encapsulation.

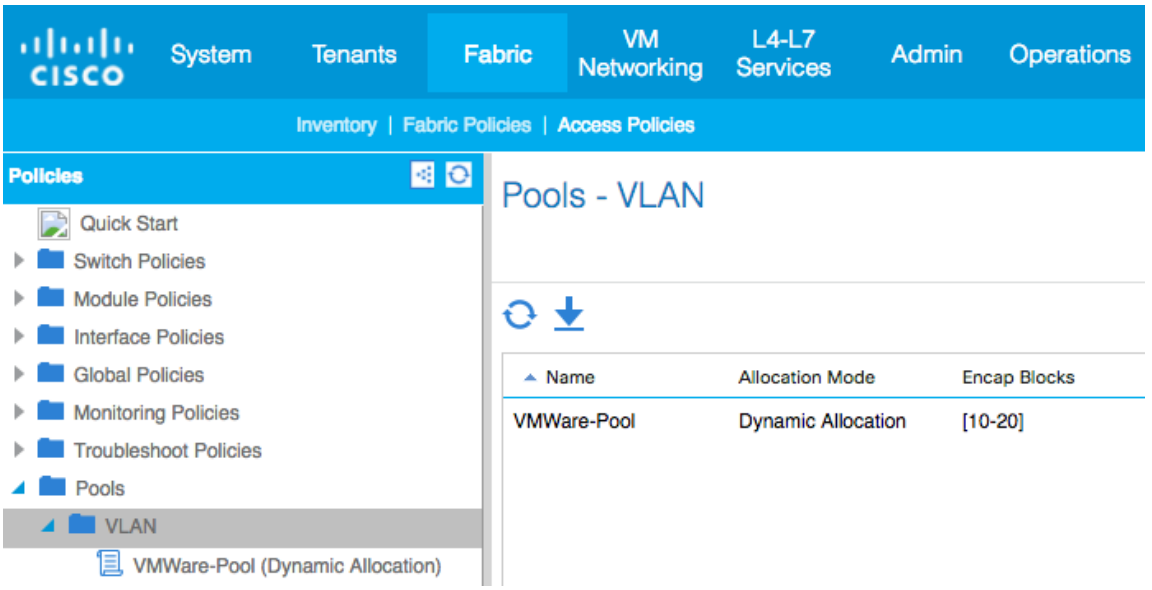
How to do it…

This recipe assumes that you already have a vCenter server already setup

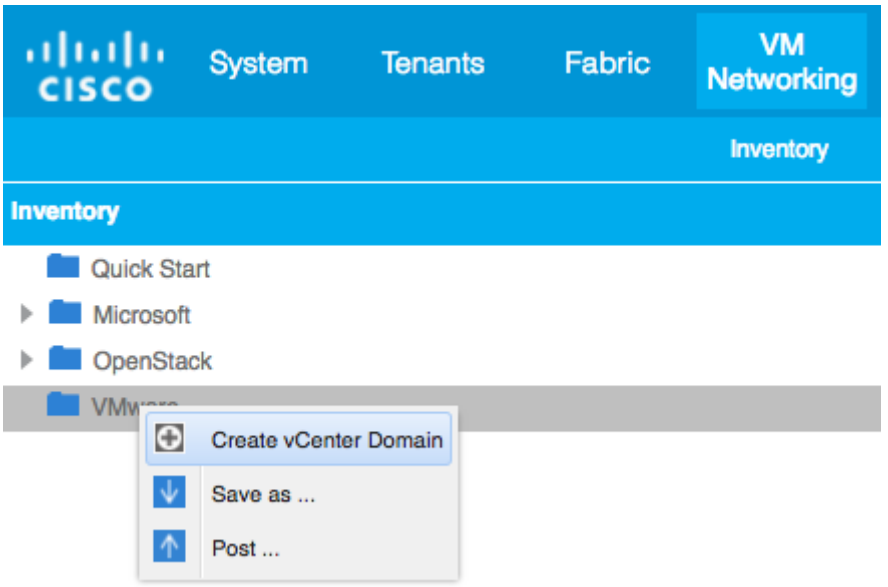
1. From **Fabric** > **Access Policies** > **Pools** > **VLAN**, Select "**Create VLAN Pool**" from the Actions menu
2. Give the pool a name and click the plus sign to set the block range:



3. Click **OK**. The new range should be listed under "Encap Blocks:." Click on **Submit**. The new VLAN pool will be listed:

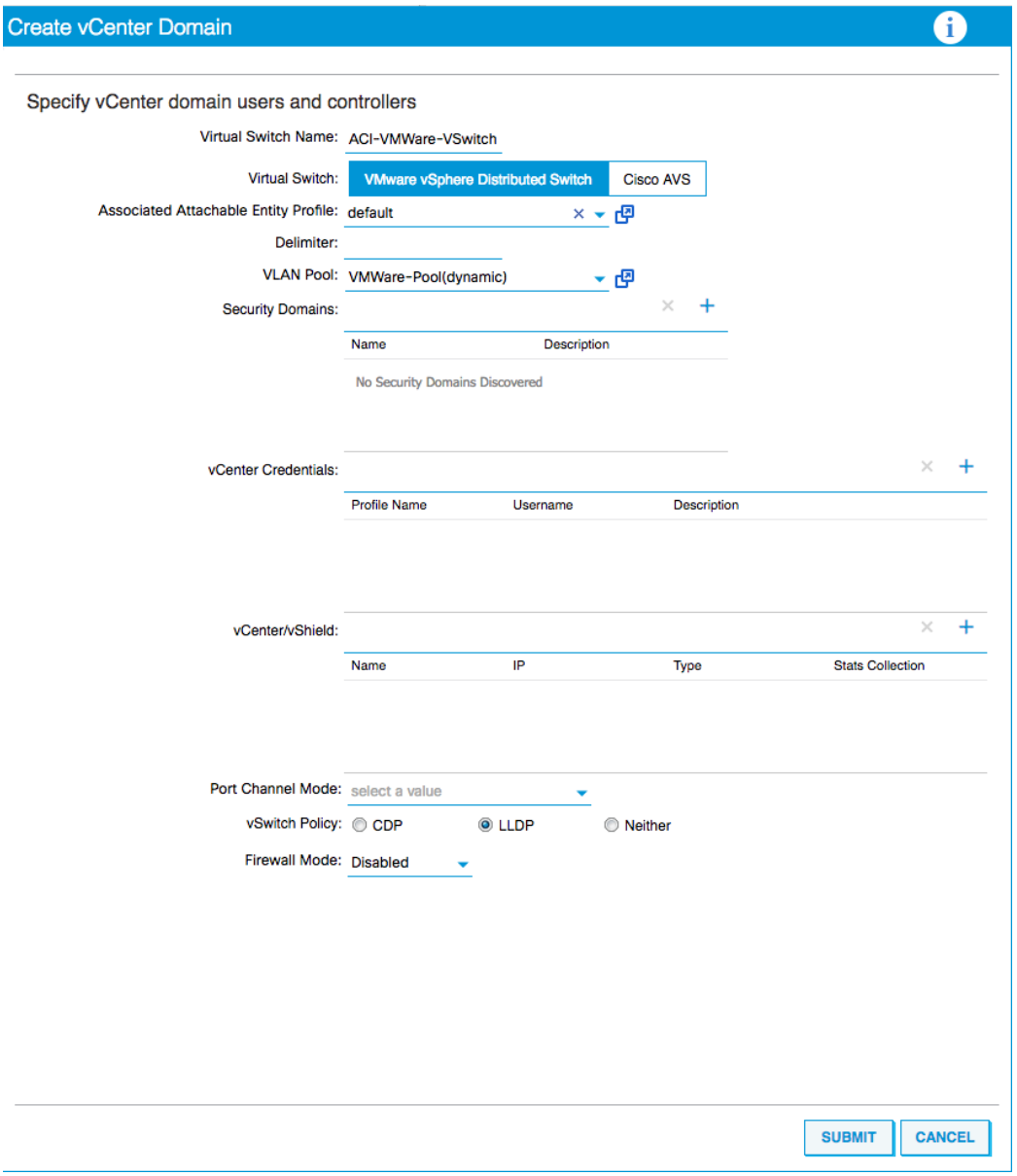


4. From the VM Networking menu, right click on VMware and select "**Create vCenter Domain**."



5. Fill in the details for the "**Virtual Switch Name**," and from the drop-down, select

the VLAN pool created earlier.



6. Fill in the vCenter Credentials with an appropriate username and password by clicking the plus sign.

7. Create the relevant credentials. Here will name the controller, set the IP address (or hostname), the **Distributed Virtual Switch** (**DVS**) version, the data center (as defined in vCenter), the associated credential (created in step 6).

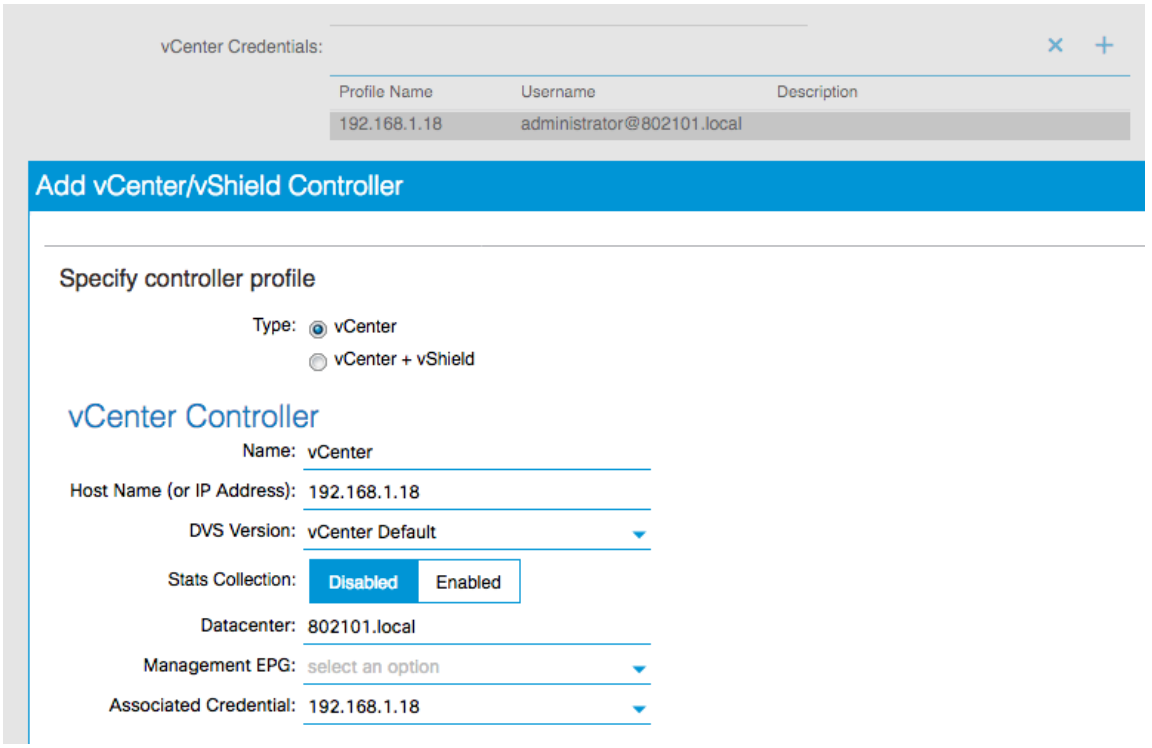
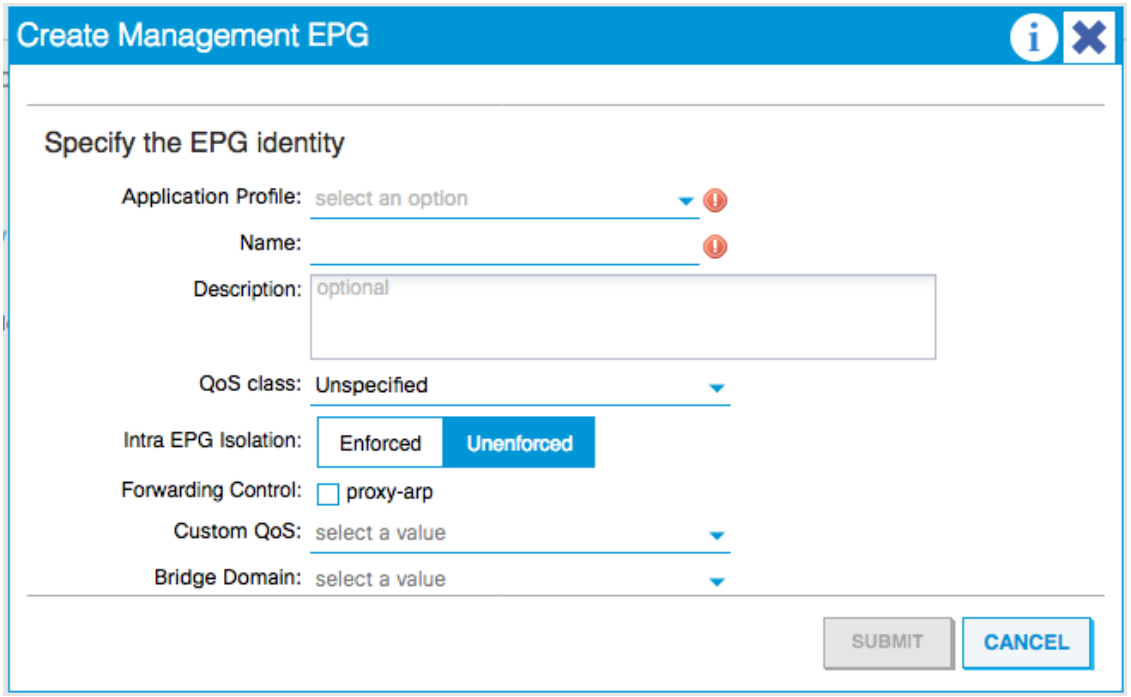
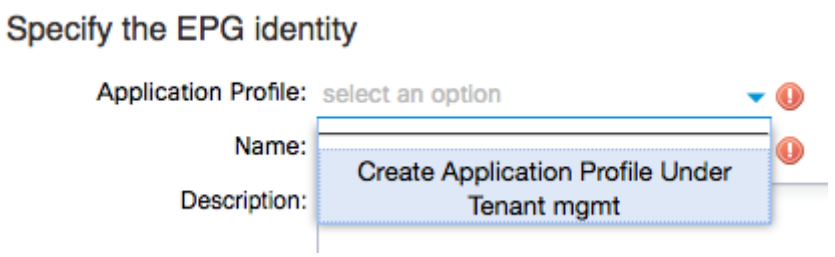


Figure 111.

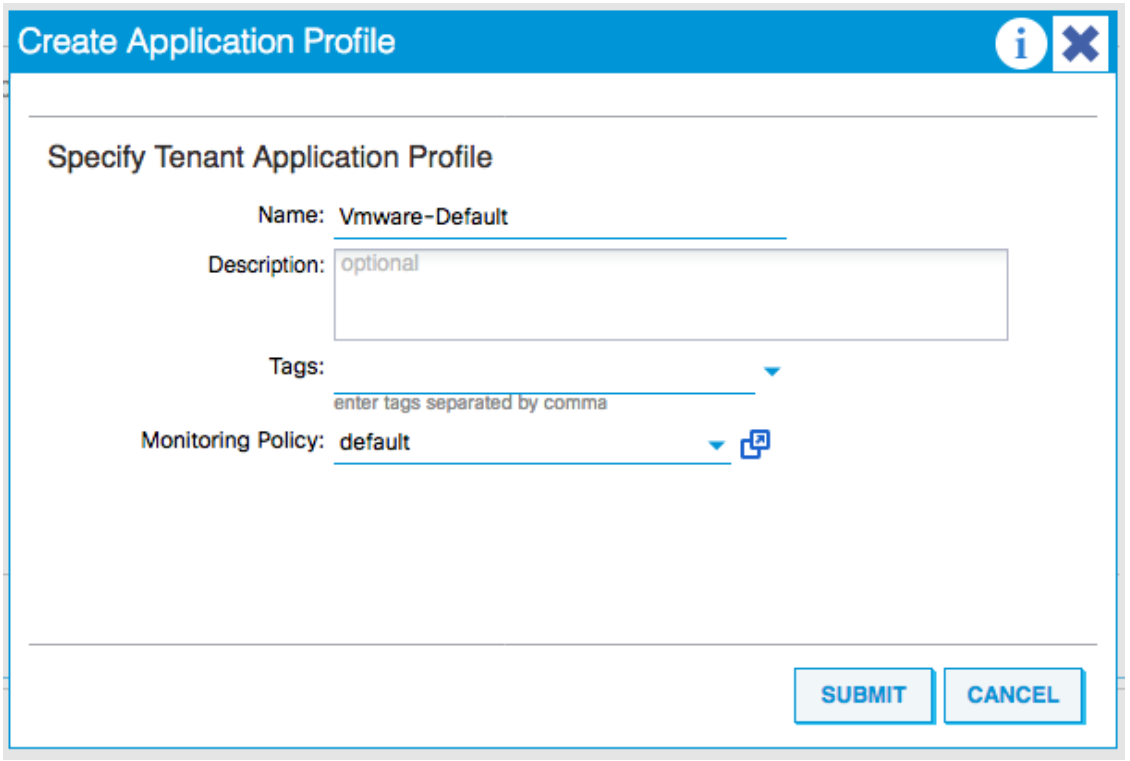
8. We also need to create the management EPG. Clicking on the drop down arrow next to this option brings up the option to “Create EPG Under Tenant mgmt.” Clicking on this option brings up another window.



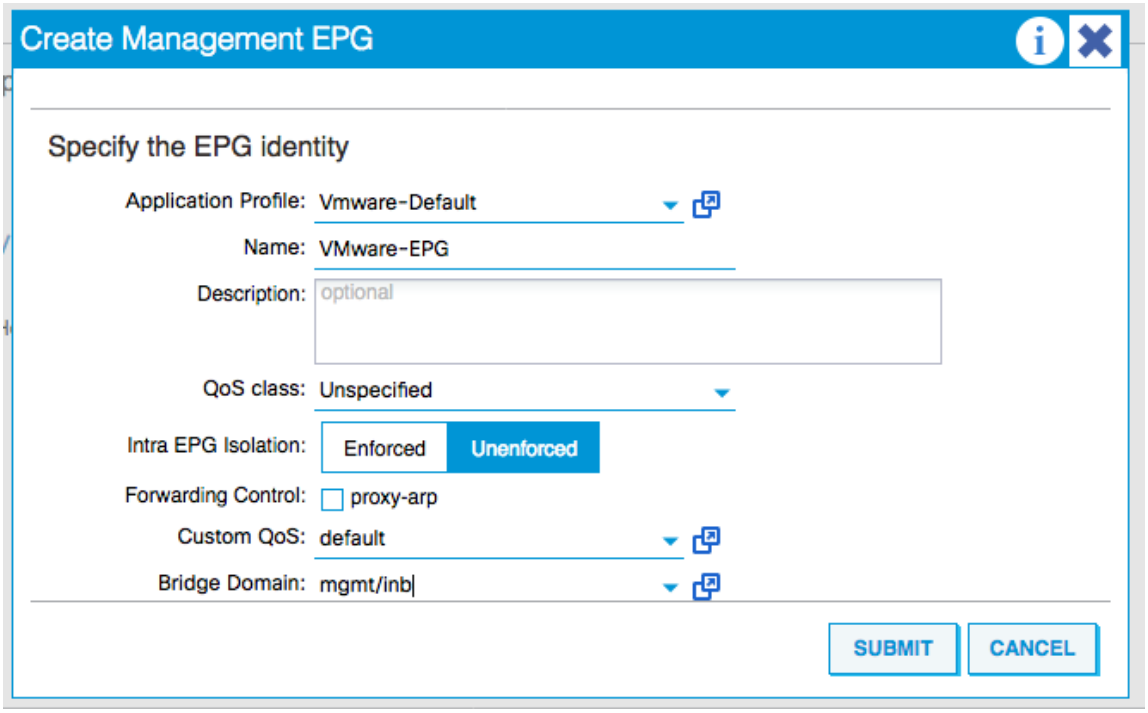
9. We create an Application Profile by clicking the drop down in the image above and selecting "Create Application Profile Under Tenant mgmt.".



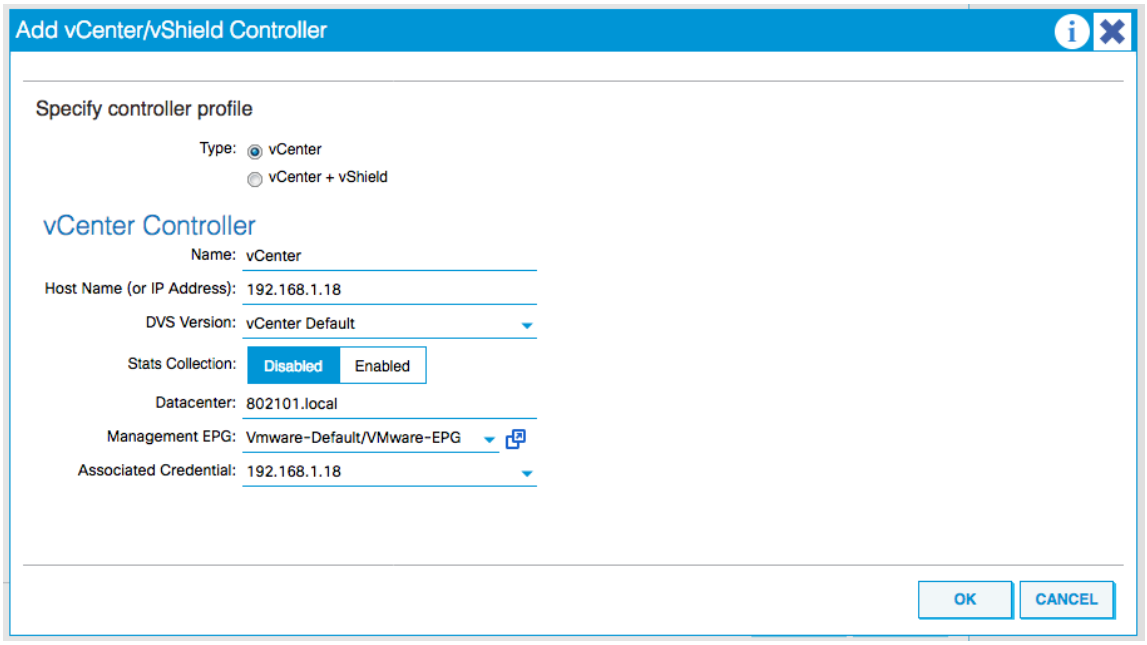
10. We name the application profile and select the monitoring policy (if we have one) and add any tags or description.



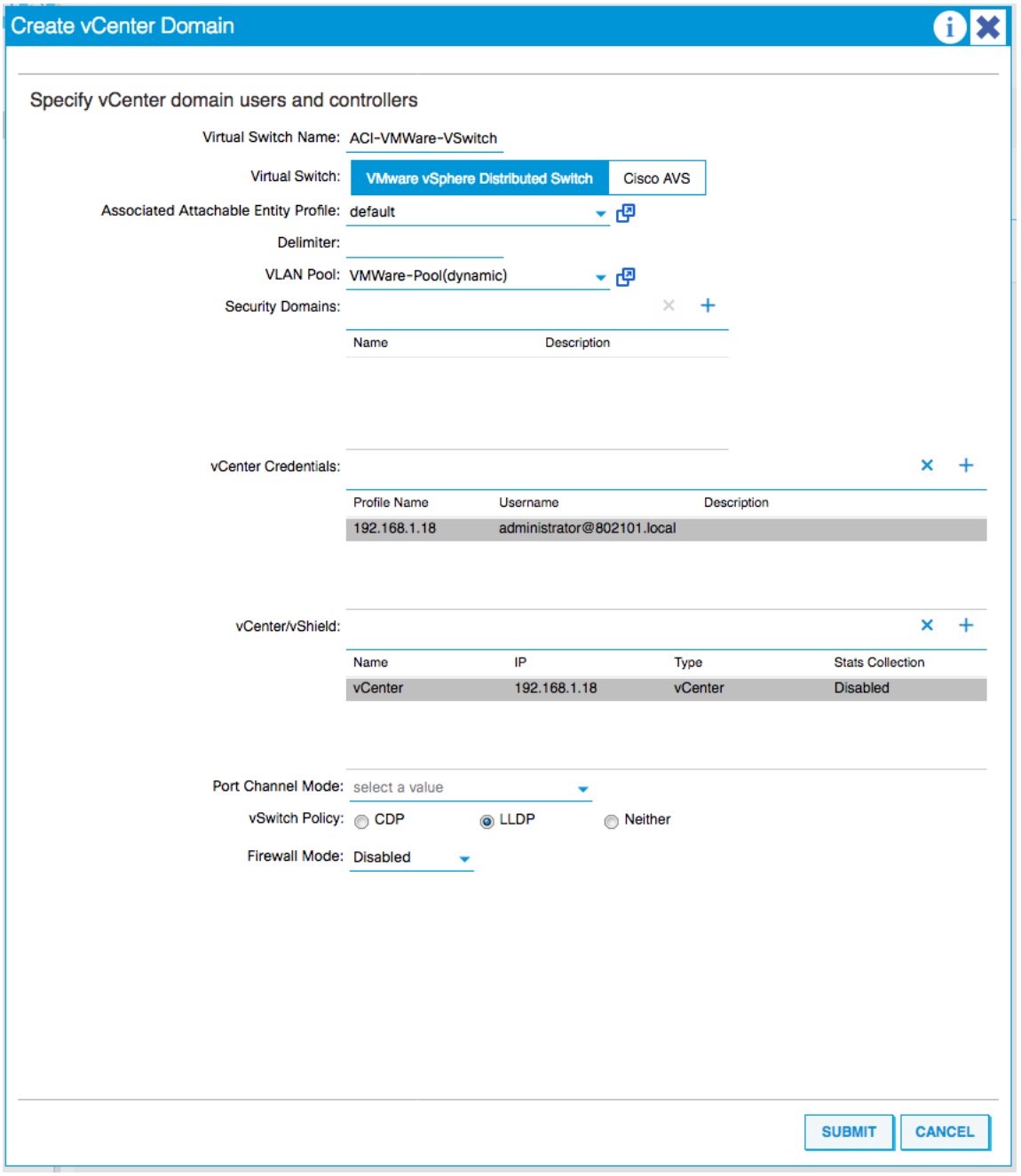
11. Click on **Submit**.
12. Returning to the previous screen, we can name the EPG and accept the default values for the other options.



13. Click Submit.
14. On the final screen, click on **OK**



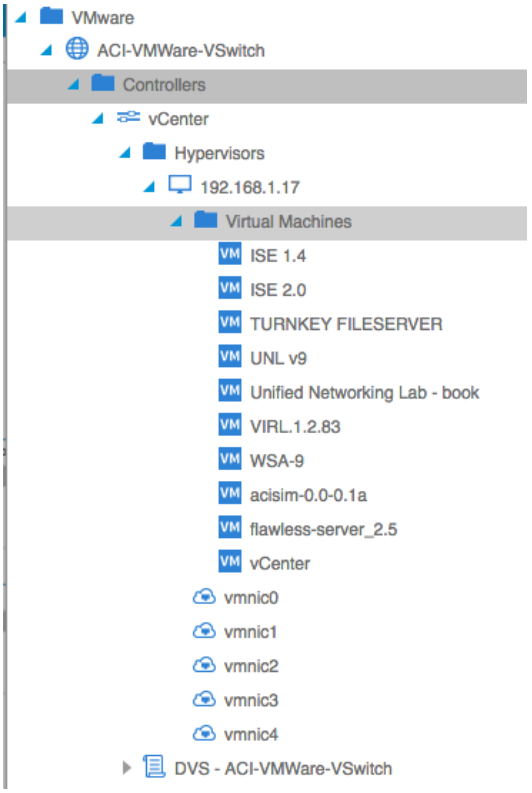
15. Click on **Submit**



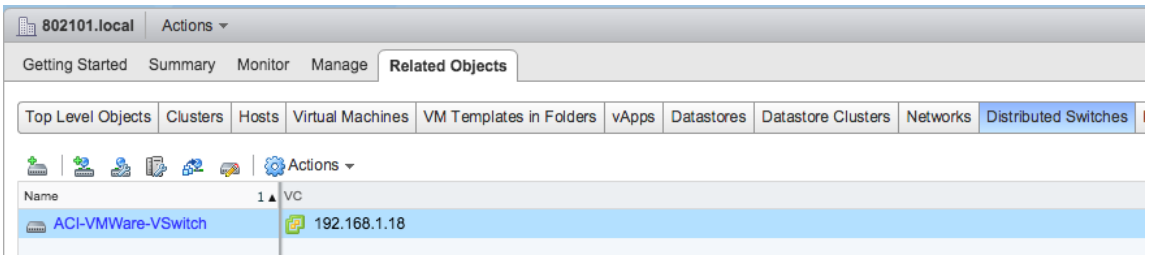
I have set the vSwitch policy to use LLDP, which will be significant later.

Note that although we have not specified any Security Domains or Port Channel Mode, the new configuration is accepted.

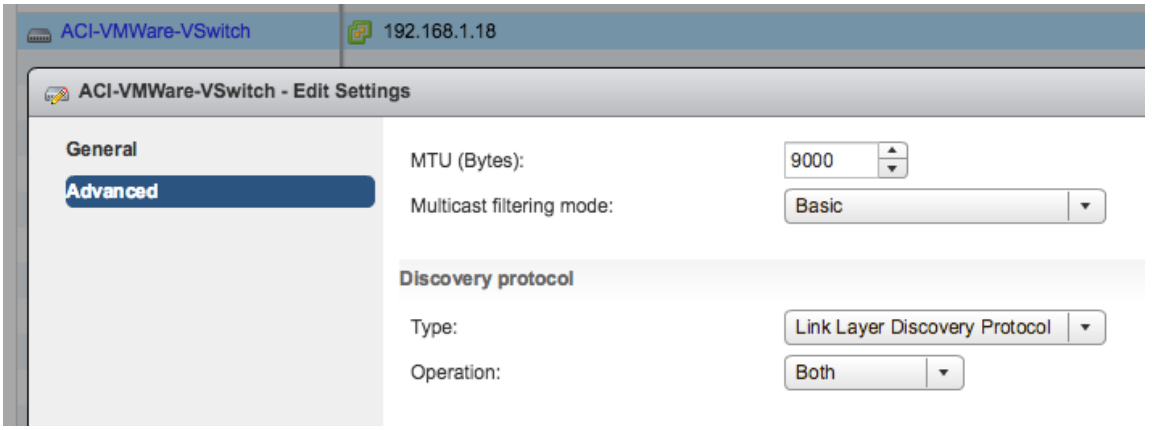
16. If we start drilling through the left-hand side menu, we should be able to see any virtual machines running on the ESXi hosts connected to the vCenter server:



17. We can check that the DVS is created by looking through the vCenter console:



18. If we look at the properties of the vSwitch, we can see that LLDP is enabled (as per the configuration made earlier.



There's more…

There is an ACI plugin for vCenter:

`http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/virtualization/b_ACI_Virtualization_Guide_2_0_1x/b_ACI_Virtualization_Guide_2_0_1x_chapter_01010.pdf`

The plugin allows you to manage the ACI fabric from within vSphere web client. You can create, modify and delete tenants, application profiles, EPGs, contracts, VRFs and bridge domains, instead of switching between the different applications. You can install the plugin by visiting https://<APIC IP>/vcplugin

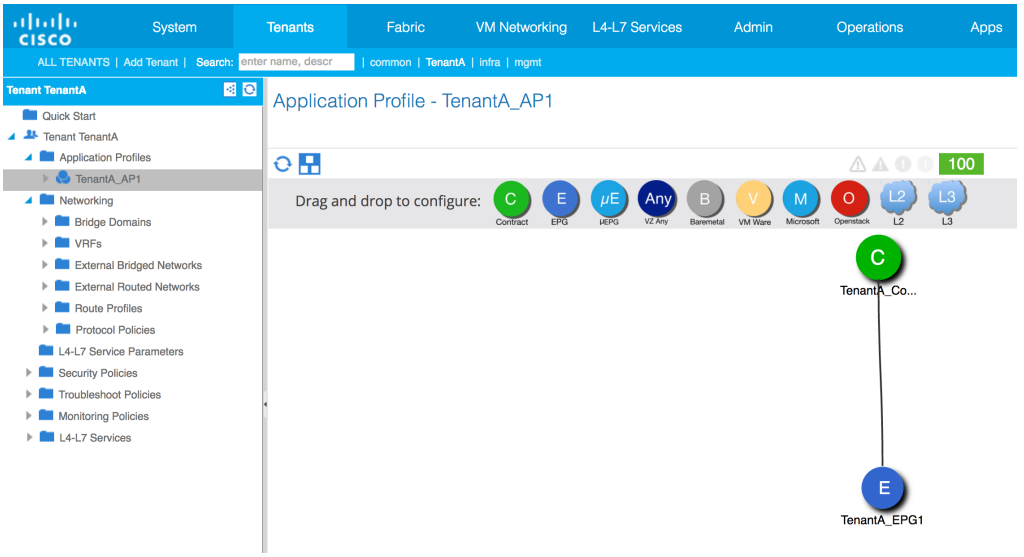
The next step is to associate the vCenter domain with our tenant.

Associating a vCenter domain with a tenant

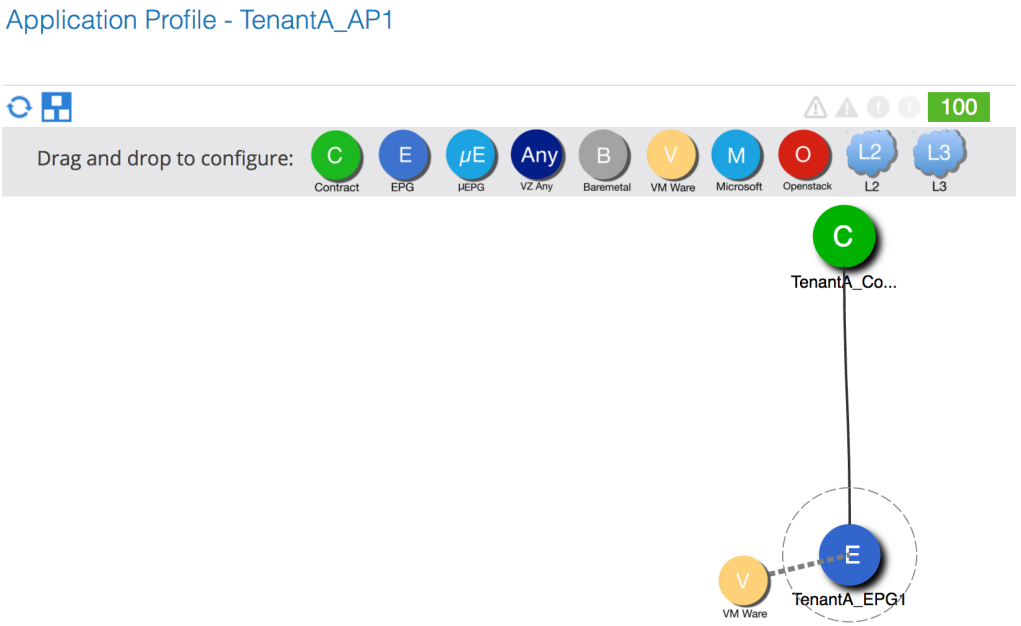
We can associate a vCenter domain with a tenant through the drag and drop interface.

How to do it…

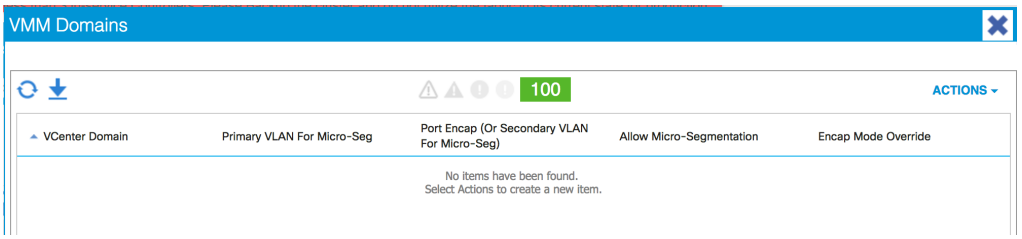
1. Navigate to the tenant's Application Profile.



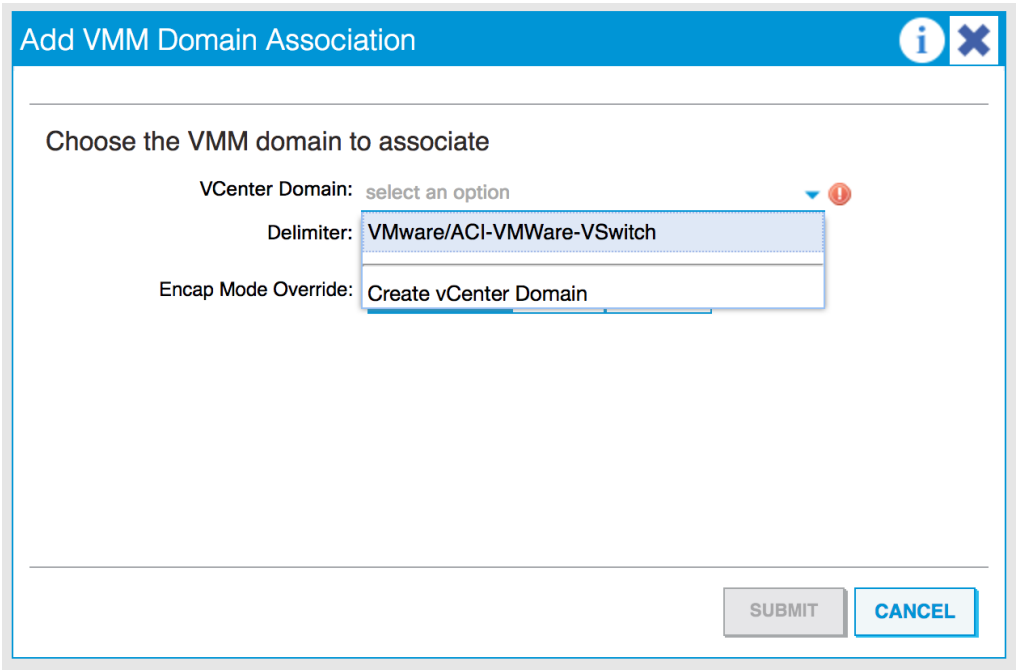
2. Drag a “VM Ware” object onto the Tenant EPG. You should see a dotted line appear.



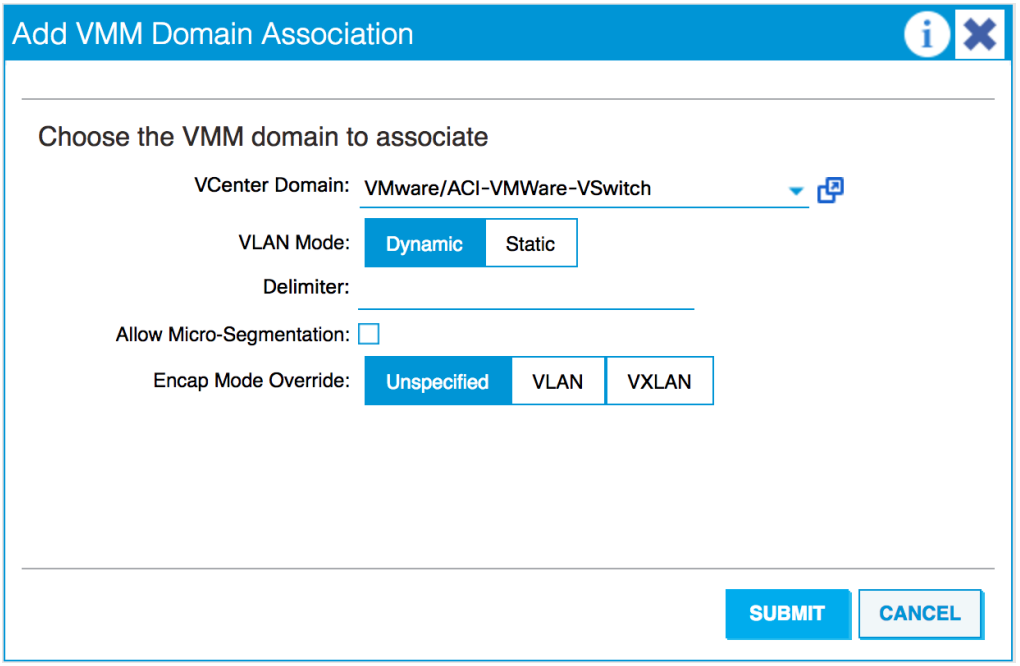
3. Once you release the mouse button, a new window will appear.



4. Click on the Actions menu and select "Add VMM Domain Association."
5. In the new window, select the vCenter domain added in the previous recipe.

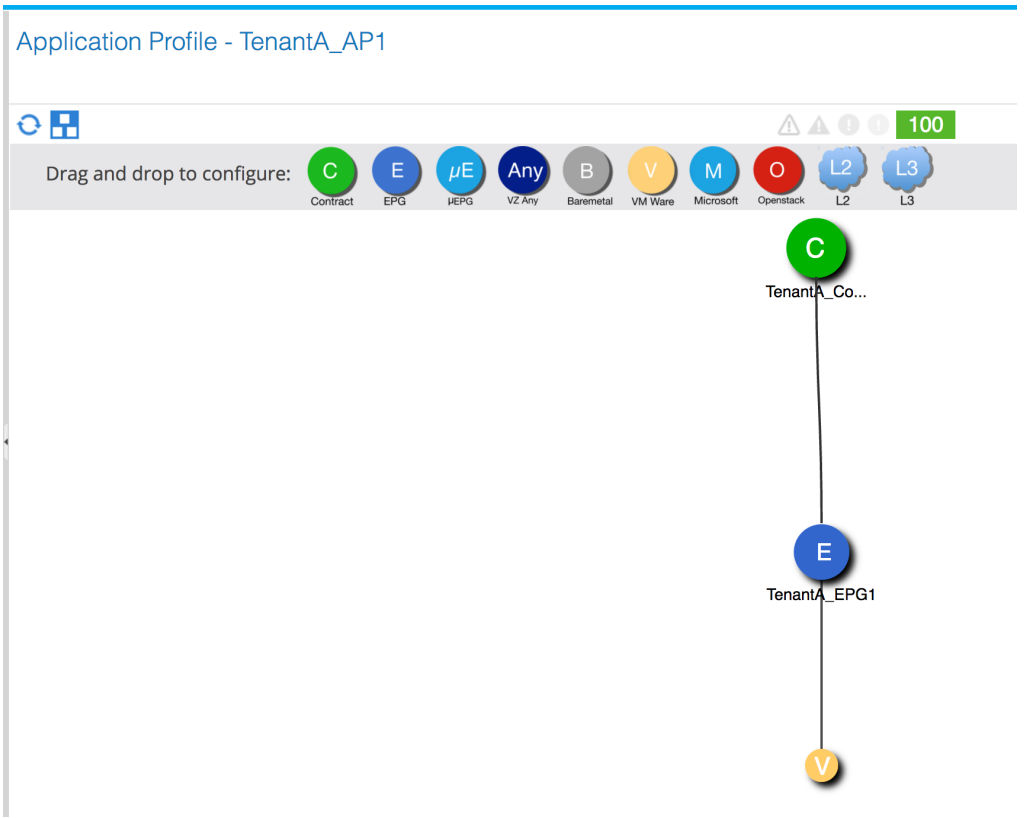


6. Choose the appropriate VLAN mode and encapsulation mode.



Click **Submit**.

7. The VMM domain will now be associated, and appear on the tenant's Application Profile.



How it works…

We have now created a **distributed virtual switch** (**DVS**) and, through this, have connected vCenter to the tenant.

There is another way to connect vCenter though; the Cisco way.

Deploying the AVS

The Cisco **Application Virtual Switch** (**AVS**) is an alternative to the **vSphere Distributed Switch** (**VDS**) we set up earlier.

The AVS is based around the Nexus 1000v switch but customized for ACI.

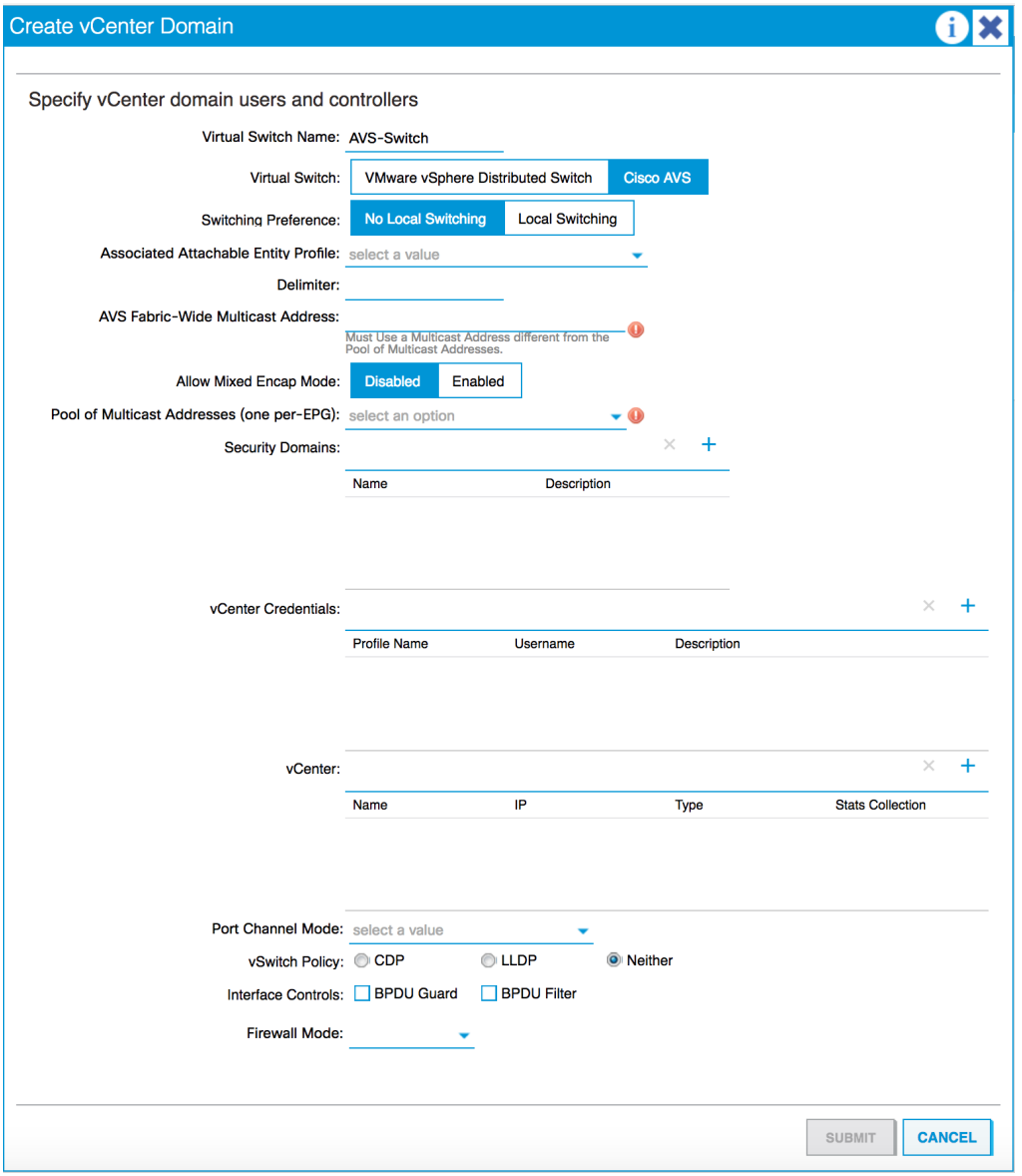
The benefits of the AVS are that it allows you to create a Virtual Tunnel End Point (VTEP) on the VMWare hosts. This enhances the scalability (over the VDS) as we are not bound by a one-hop limit. One of the differences between using the DVS to the AVS is that the DVS uses LLDP for VM discovery, whereas AVS uses OpFlex.

We need to install an additional plugin to vCenter to be able to run the AVS. You can download it from the following link: `https://software.cisco.com/download/release.html?mdfid=282646785&softwareid=286280428&release=1.1` You will require a CCO account and the specific entitlement.

How to do it…

Earlier in this chapter, we created a vCenter domain. As part of this, we created a vSphere Distributed Switch. The other option would be to use the Cisco AVS (refer to the figure in step 5 in the Creating VMM domains and integrating VMWare recipe above).

1. Click on VM Networking
2. Right click on VMware
3. Select Create vCenter Domain
4. Give it a name.
5. Select Cisco AVS as the Virtual Switch



6. Set the switching preference

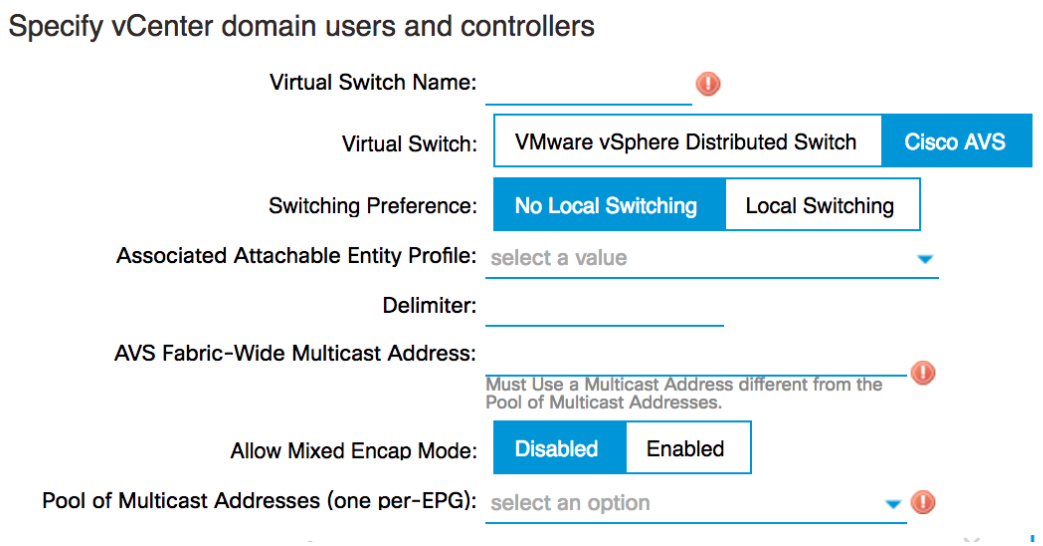
7. Set the "AVS Fabric-Wide Multicast Address."

8. Set the multicast address pool (creating one if required)

9. Set the vCenter credentials and vCenter details (which is the same as the other recipe)
10. Set the port-channel mode, vSwitch policy, interface controls and firewall mode.

How it works…

Many of the options can be left as the default values. The essential ones will have a red circle with an exclamation mark in it.



In the settings above, No Local Switching was selected (the default setting). With this setting, all traffic between VMs in the same EPG must flow through the leaf node and VXLAN is the only encapsulation method available. This is also known as "FEX Enable Mode." This is not the recommended method.

The preferred method is Local Switching Mode. Traffic between VMs in the same EPG is routed by the hypervisor, instead of having to flow through the leaf node. Traffic between VMs in different EPGs will still need to flow through the leaf node, but there can still be

considerable performance gains with local switching mode. Local switching mode can also use VLAN and VXLAN for encapsulation, and is referred to as "FEX Disable Mode."

There's more…

- For the AVS to work, it needs to be installed. Follow this guide for how to install the plugin and the AVS.

```
http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus1000/avs/vsum-getting-started/1-0/b_Cisco_Virtual_Switch_Update_Manager_Getting_Started_Guide_Release_1_0_For_Cisco_AVS/b_Cisco_Virtual_Switch_Update_Manager_Getting_Started_Guide_Release_1_0_For_Cisco_AVS_chapter_01.pdf
```

- There is also a useful PDF of the differences between the DVS and the AVS.

```
http://www.cisco.com/assets/global/DK/seminarer/pdfs/Cisco_Tech_Update-ACI_Hypervisor_integration-22_og_24_september_2015.pdf
```

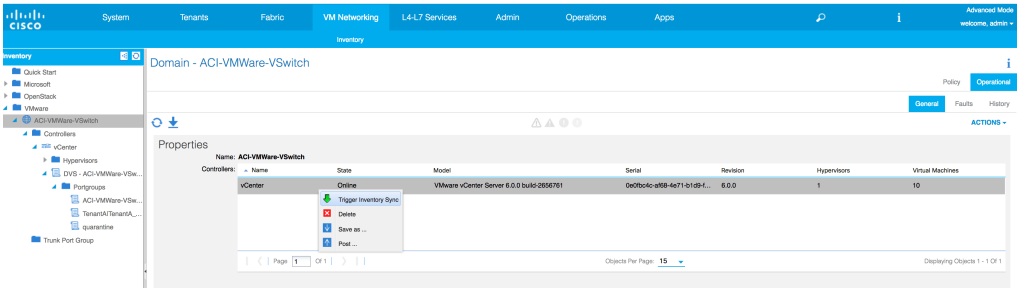
Discovering VMWare endpoints

Naturally, there will be additional VMs created within the VMWare environment. As such, we will need the new machines to be reflected in ACI.

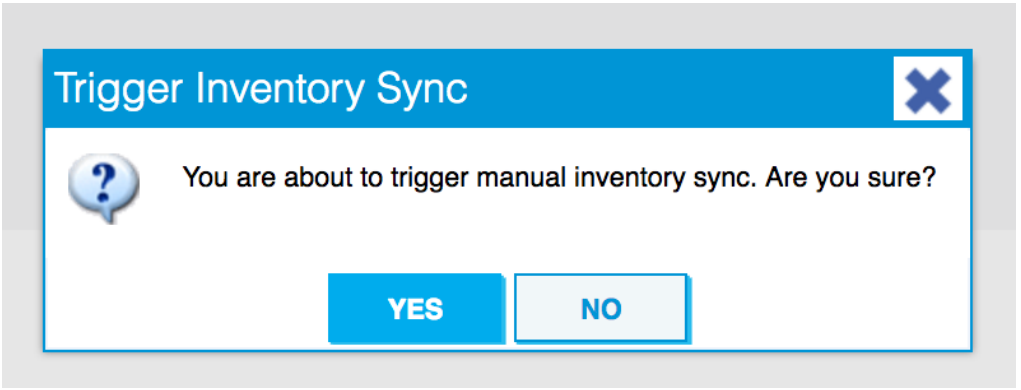
We can do this manually.

How to do it…

1. From the VM Networking tab, open up the VMWare menu, and click on the switch that was created earlier.
2. Right-click on the vCenter name in the properties window.
3. Select "Trigger Inventory Sync."



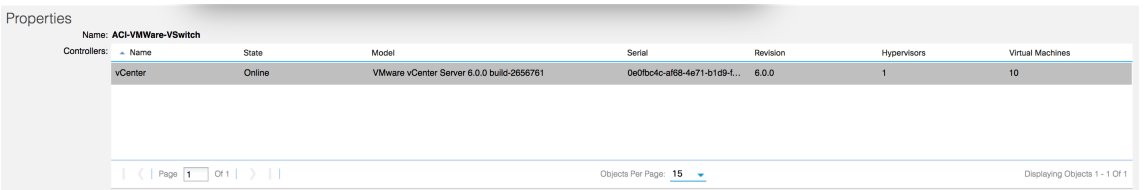
4. Click “Yes” to the message that pops up.



How it works…

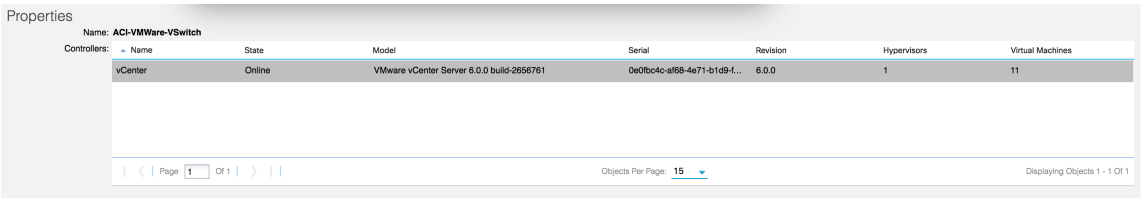
When we trigger the inventory sync, ACI polls the connected vCenter servers and pulls in a list of all the virtual machines.

Here are the statistics before the sync:



Notice that ACI sees ten virtual machines.

After the update, we can see 11 virtual machines.



We do not need to perform manual updates all the time, however as the APIC will do this automatically as well.

Adding Virtual Machines to a tenant

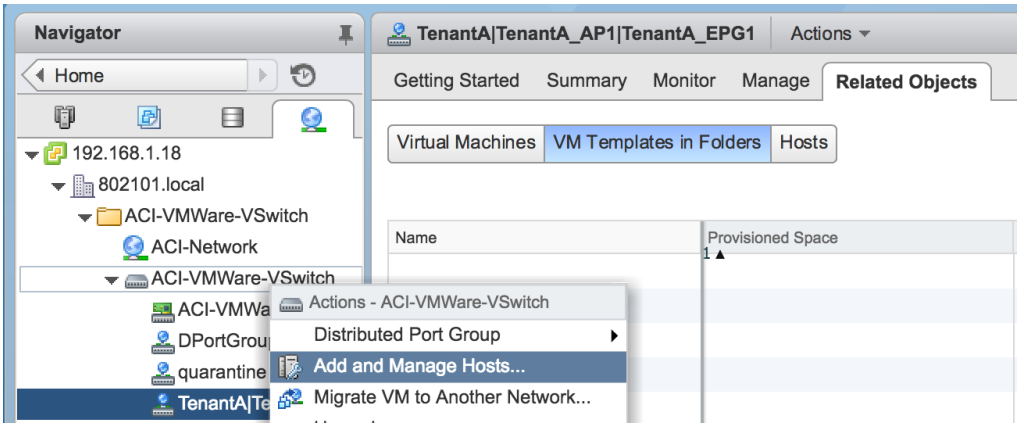
Now that we have associated our vCenter to the APIC, we need to start associating some virtual hosts to the tenant.

To do this, I set up a new ESXi host in VirtualBox (with the VM set to use 2CPUs, two 30GB hard disks, and 4GB memory). If you are short on available hardware and are running the ACI Simulator within one ESXi host, this avoids any issues that could be caused by adding the host controlling the ACI simulator to a tenant running within the simulator.

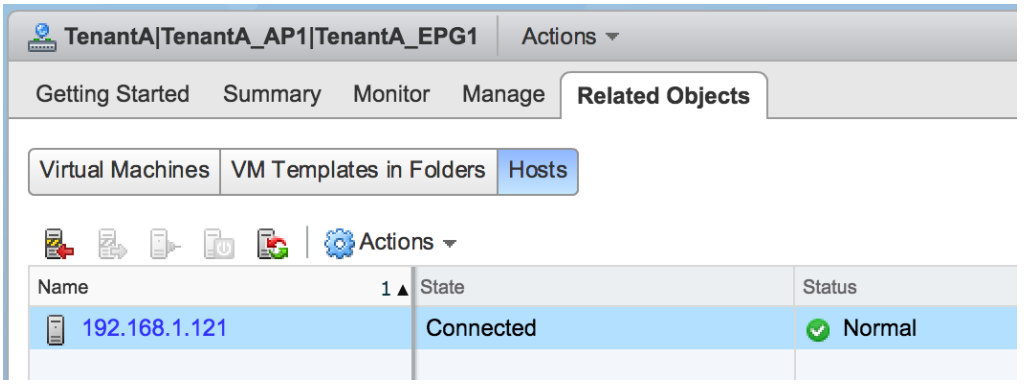
I am not sure if there would be any issues, but I did not want to encounter any “divide by zero” issues!

How to do this…

1. Add the ESXi host to vCenter and license it.
2. Go to the networking settings, and add the new host to the DVS by right clicking on the switch and selecting “Add and Manage Hosts.”

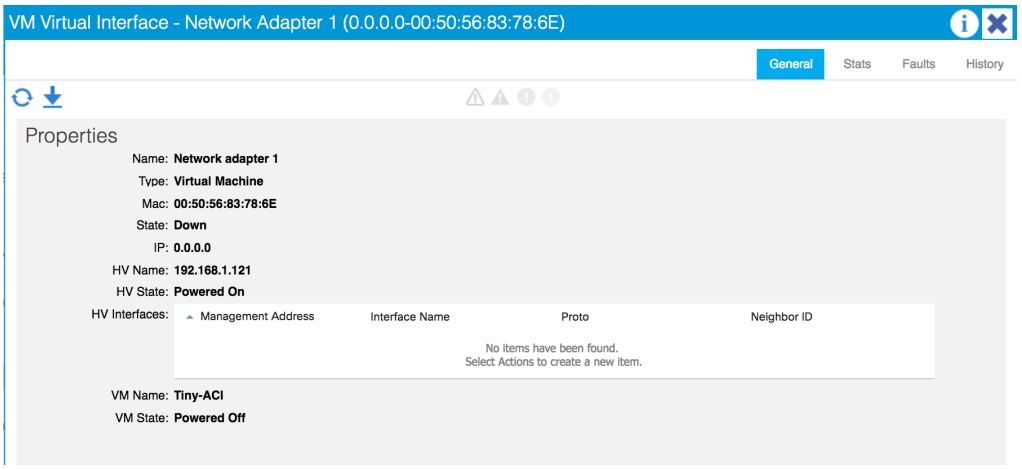


3. Follow the wizard to add the ESXi server. Once this has completed, you should see the host listed.



4. Import an OVA file (I am using a Tiny Linux OVA file due to small footprint), by selecting the import option under "Virtual Machines." I have called the VM "Tiny-ACI."

5. You should see the virtual machine from the ACI GUI.



How it works…

By adding an ESXi host to the tenant, we can run virtual machines within the tenant. But once we start adding virtual machines to hosts, how do we keep track of them? Let's find out!

Using Virtual Machine Tracking

Once the number of tenants increases, it is likely that the number of virtual machines will also increase. How, then, do you keep track of all the VMs and to which tenant they belong?

Thankfully, the ACI Toolkit can help you with this (and a host of other scenarios).

How to do it…

1. Download the ACI Toolkit virtual machine from `http://bit.ly/1IzliZY`.
2. Deploy the OVA file to create the VMWare guest virtual machine.
3. Update the packages by logging into the virtual machine using the username and password of “acitoolkit,” then run the command “sudo ~/install.”
4. Connect the APIC to the ACI Toolkit’s MySQL database by running the command

1. `python aci-endpoint-tracker.py -u https://192.168.1.205 -l admin -p apicpassword -i 127.0.0.1 -a root -s mysqlpassword`

5. Log into MySQL using the command

    5. `mysql –u root –p`

    You will be prompted for the password

6. Switch to the ACI Toolkit database using the command

    6. `use acitoolkit;`

7. We can then query the endpoint table by using some of the following commands:

    7. `select * from endpoints;`
    `select * from endpoints where tenant='Tenant_A"`

How it works…

The ACI Endpoint Tracker connects the APIC to a MySQL database, which you can easily query.

For updates to be inserted into the database, the Endpoint Tracker must be running.

There's more…

This recipe is just to wet your appetite. Please refer to the developer's documentation for the wealth of functions that the toolkit offers.

`https://acitoolkit.readthedocs.io/en/latest/tutorialpackages.html`

`https://acitoolkit.readthedocs.io/en/latest/endpointtracker.html`

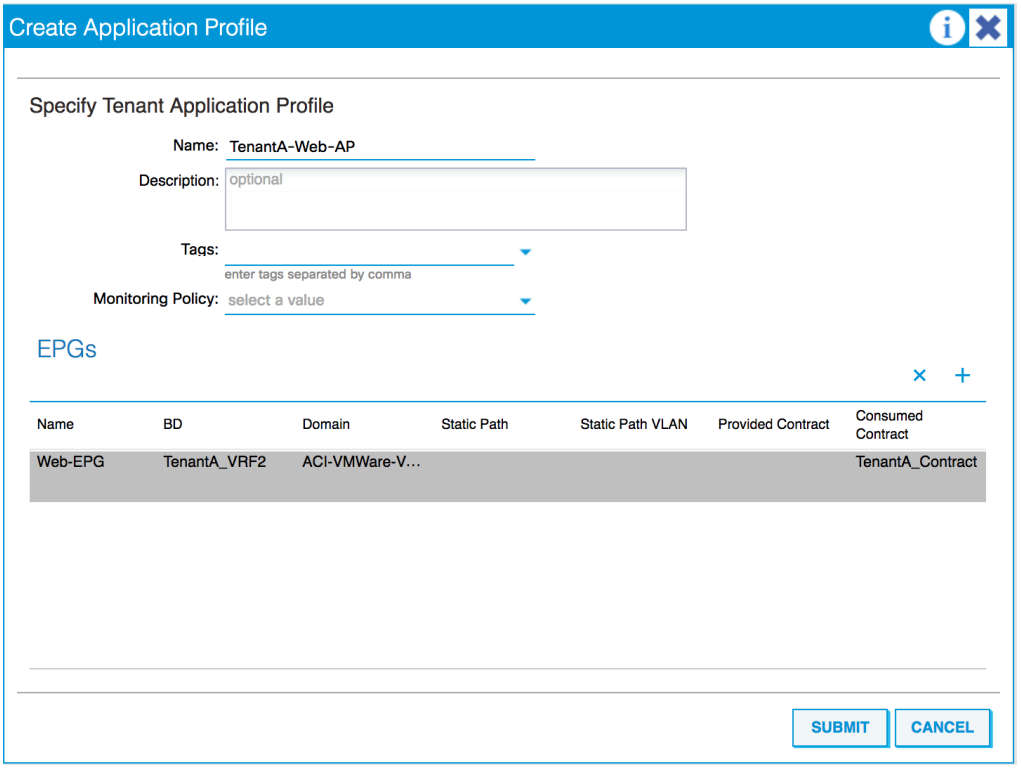
Integrating with A10

The A10 Thunder is the easiest way to get started with L4-L7 services. This is because you can download a trial of the A10 Thunder, and also download the device package from A10. The other services listed are a little more difficult to get started with unless you have a large wallet.

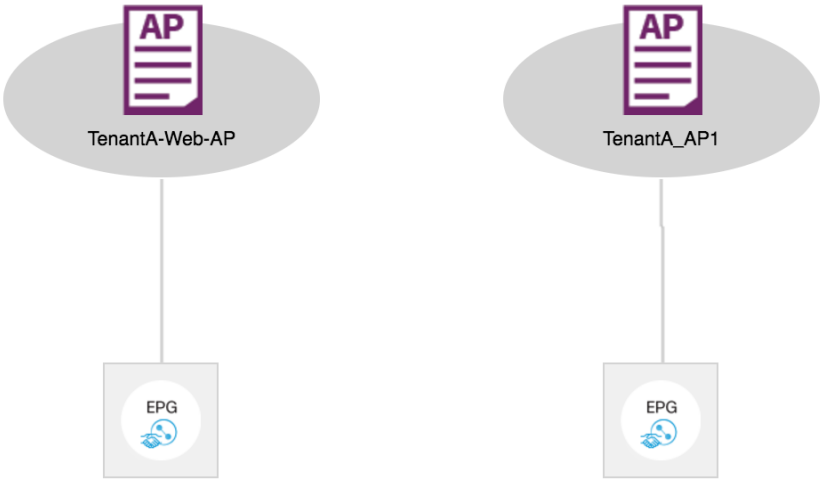
In this recipe, we will be adding a new application profile and EPG so that we can deploy the A10 between the existing TenantA_AP1 and the new Web application profile.

How to do it…

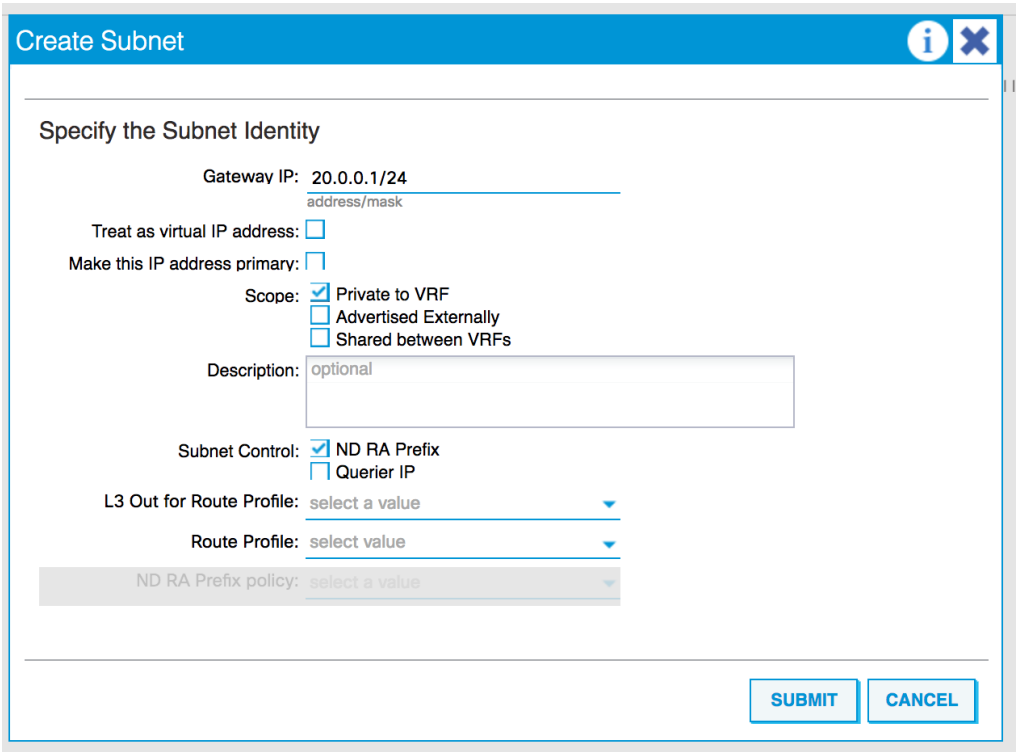
1. Create a second Application Profile, called TenantA-Web-AP, using TenantA_VRF2 bridge domain created in Chapter 2.



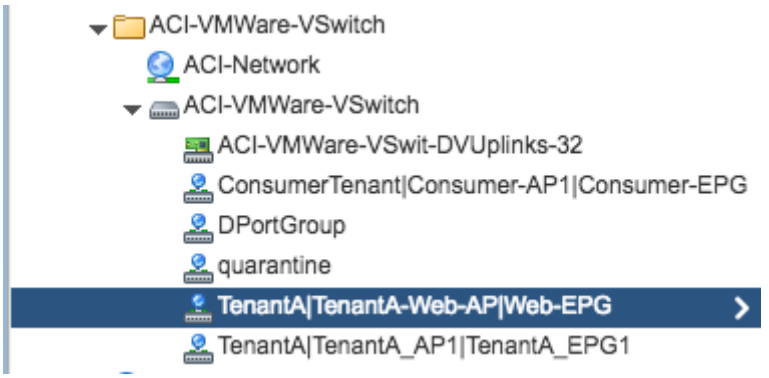
The Application Profiles window should look like this:



2. Create a subnet for TenantA_VRF2:



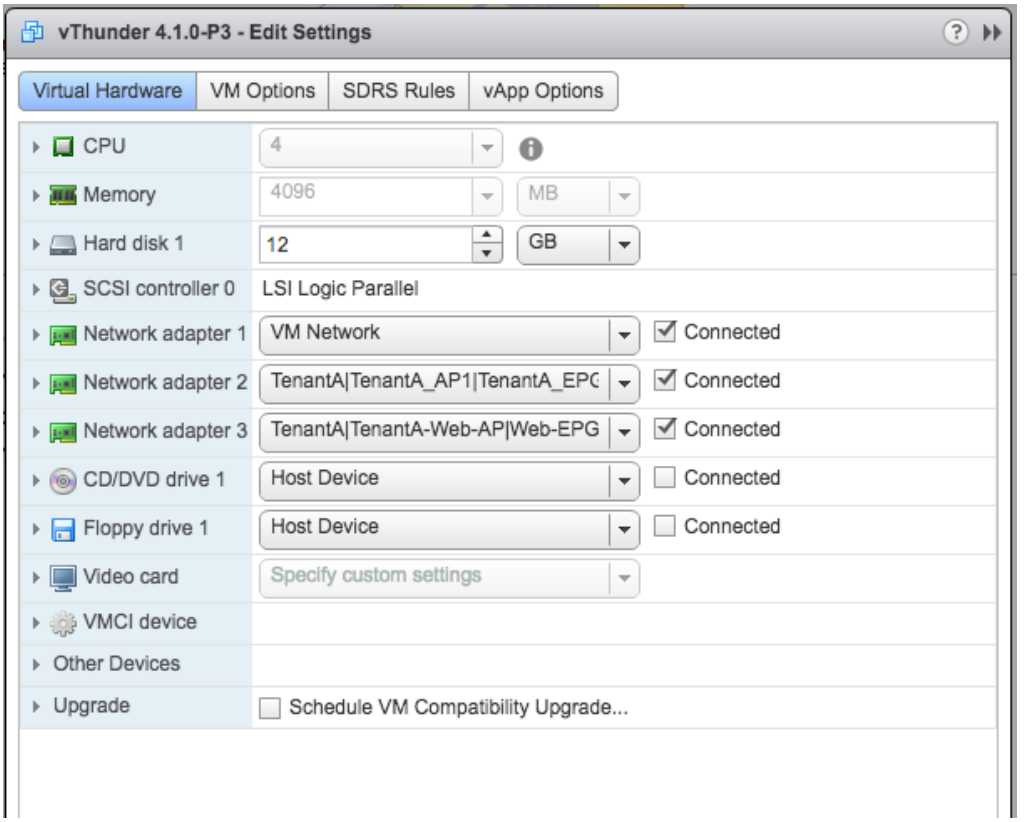
3. vCenter should see the new EPG settings



4. We have already covered installing the vThunder device package at the start of this chapter, but you can register to download it from here: `https://www.a10net`

`works.com/cisco-aci-integration`

5. Download the trial from `https://www.a10networks.com/products/virtual-ddos-mitigation-load-balancer-application-delivery`
6. Deploy the OVA template into vCenter. When you get to section 2d (Setup Networks), make sure that the first Ethernet adapter is set to use the management network (often shown as "VM Network") and that ethernet1 is set to the TenantA's AP1 EPG, and ethernet2 is set to use the web EPG.Once the VM has been deployed, the settings should look like this:



7. The VM will need licensing. There will be a licensing link on the download page; we need to get the current UID to generate the license. We can SSH to the VM (using the password of "a10" when prompted:

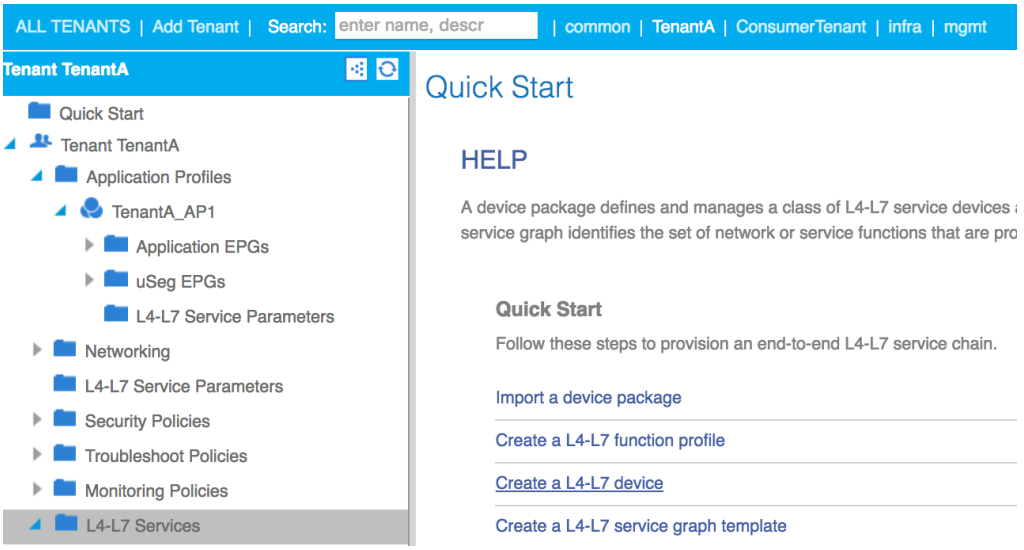
```
1. Stuarts-iMac:~ stu$ ssh admin@192.168.1.223
   Password: (type "a10" here)
```

```
ACOS system is ready now.
[type ? for help]
vThunder(NOLICENSE)>
vThunder(NOLICENSE)>en
Password: (press enter here)
vThunder(NOLICENSE)#show license uid
```

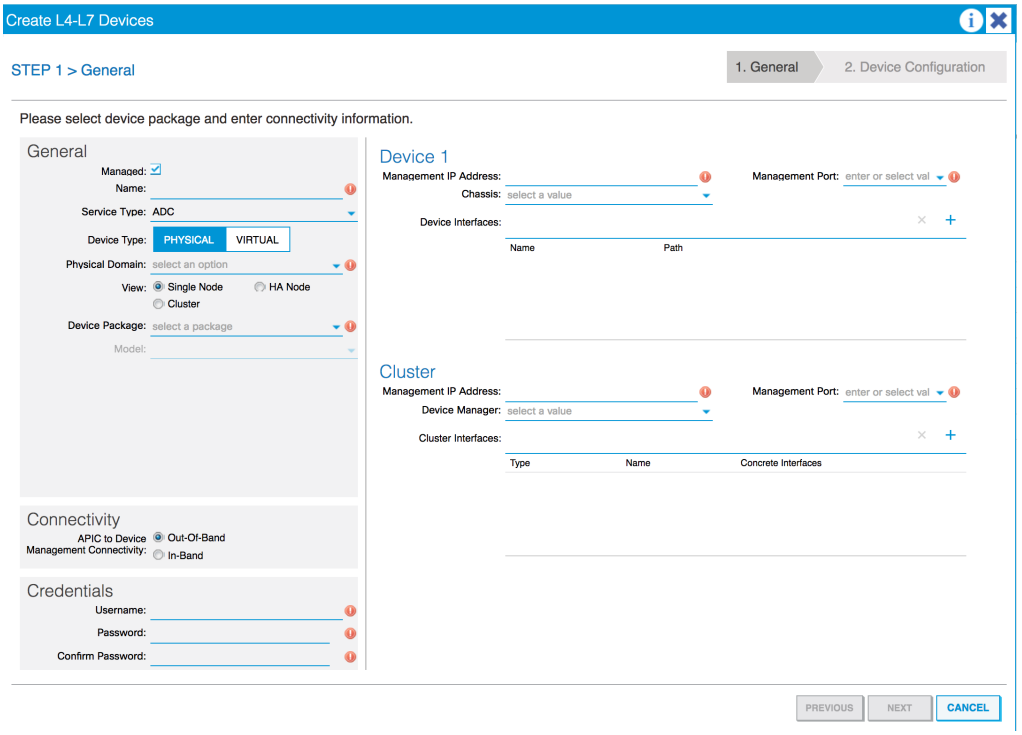
2. Copy this license into the licensing page and wait for them to email you the license file.
3. Once you have the file, start up a TFTP server and copy the trial_license.txt file to the TFTP server directory.
4. From the vThunder CLI, install the license

1. `vThunder(NOLICENSE)#import license trial_license.txt tftp://192.168.1.88/trial_license.txt (make sure that you replace any IP addresses with ones for your environment)`

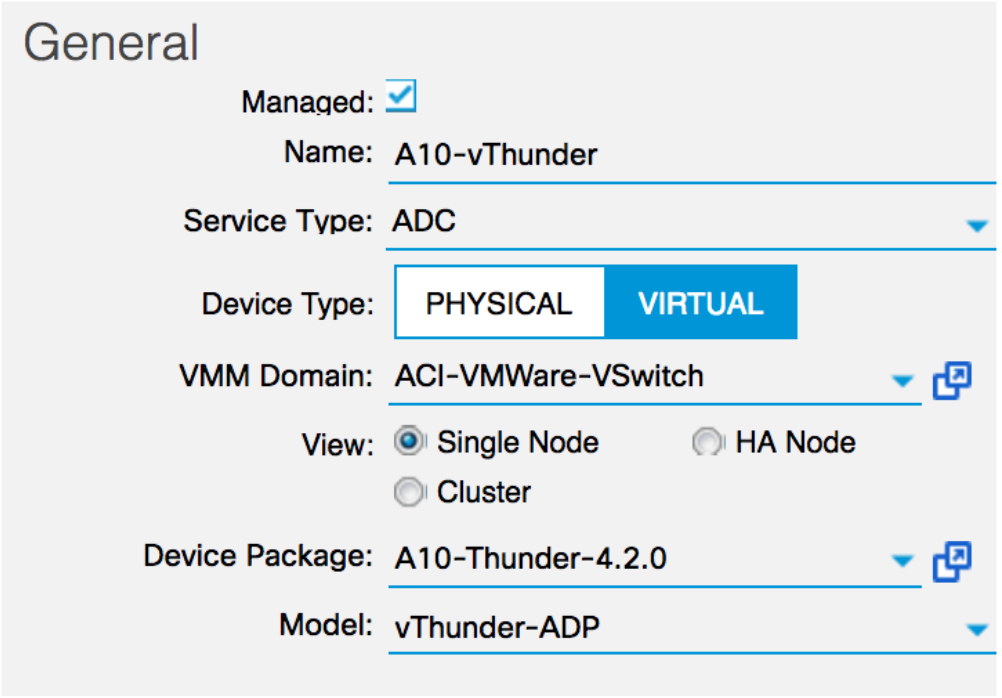
2. We now need to assign the device to the tenants. From TenantA, click on L4-L7 Services, and select “Create a L4-L7 device”:



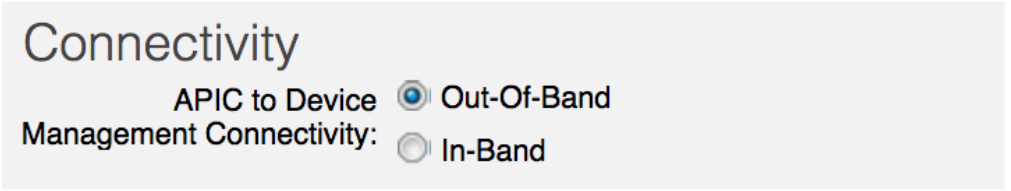
3. As mentioned earlier, the essential fields will have red exclamation circles next to them.



In the General field, fill in the name, device type, physical domain and device package:

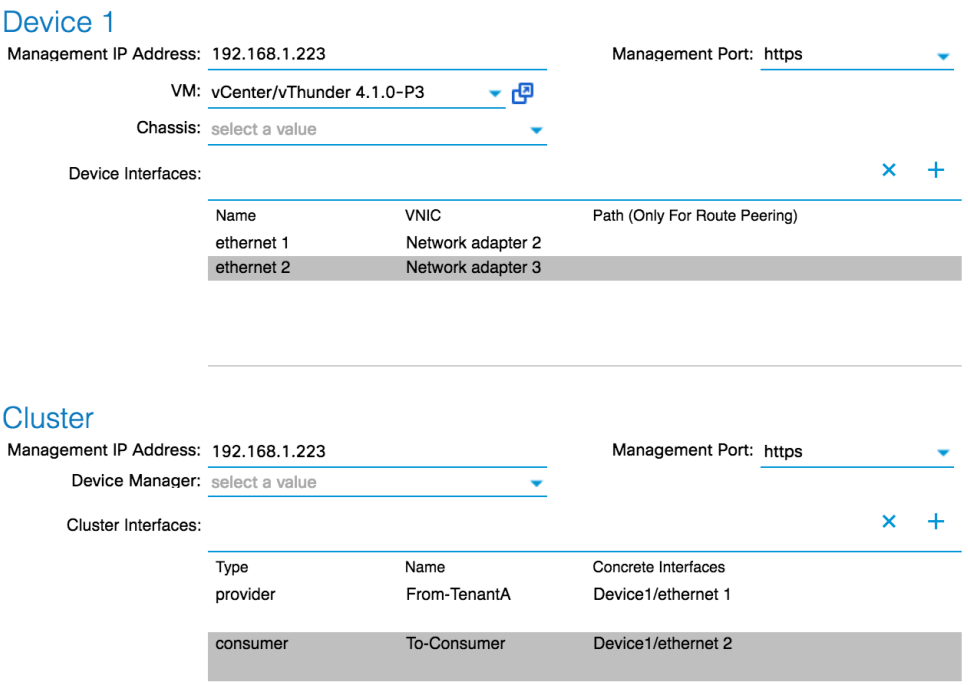


Leave the Connectivity setting to "Out-Of-Band."



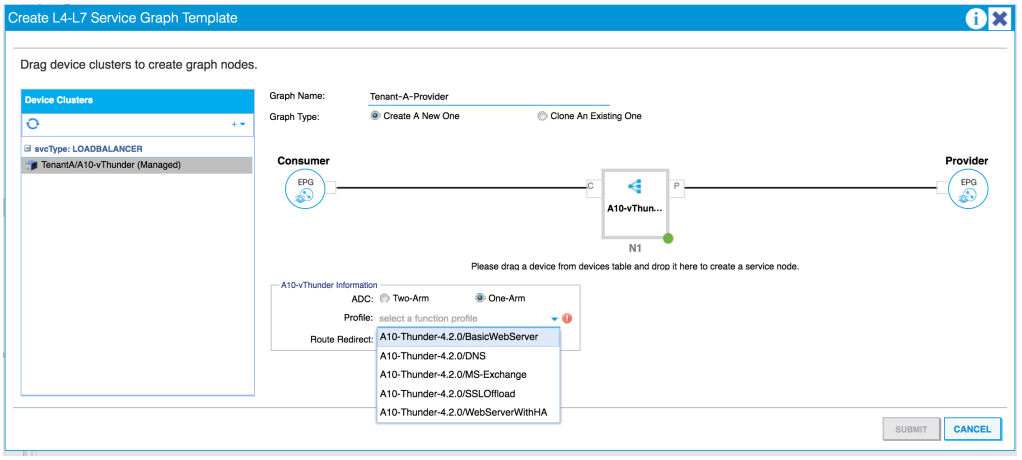
For the Credentials, use the username of admin and a password of "a10" (minus the quotes).

4. For the Device 1 field, set the IP address and port. Do not change the HTTPS port, otherwise; you will see errors in the APIC about an "unsupported port." Select the VM from the drop-down list and set the interfaces. Note that we need to set Ethernet 1 to the VM's second network adapter (which has been mapped to AP1), and set Ethernet 2 to the VM's third network adapter (assigned to the web AP).
5. In the Cluster settings, the management IP address and port will be copied from the device one settings. Click on the plus sign to create a new cluster interface, setting the type as "provider," give it a name (From-TenantA) and the concrete interface to Ethernet 1. Do the same again, setting the type as "consumer," give it a name (To-Consumer) and the interface to Ethernet 2.

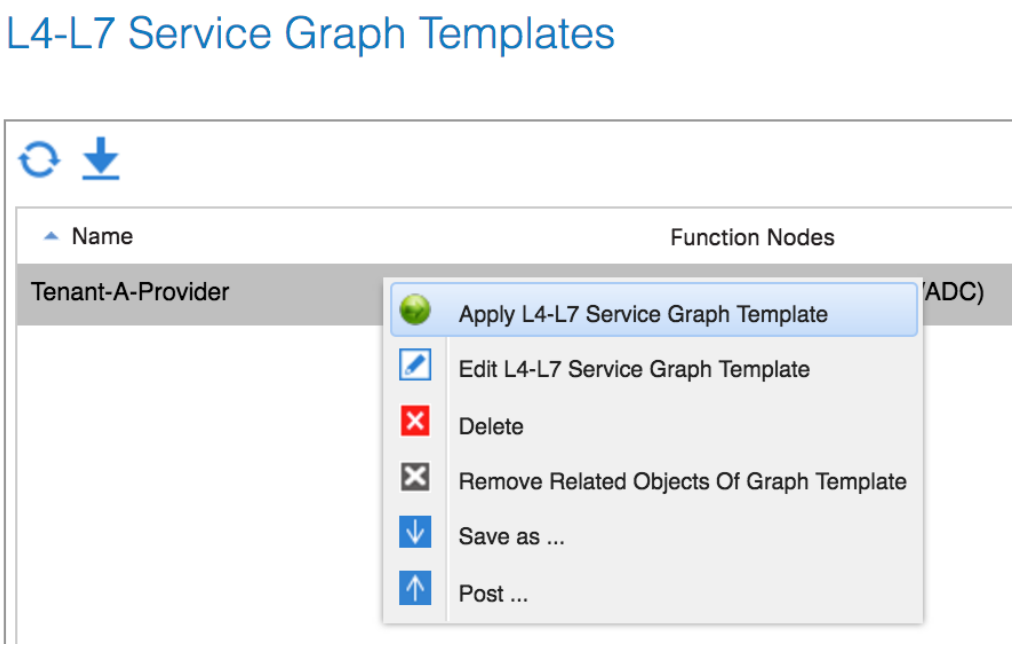


Click **Next**.

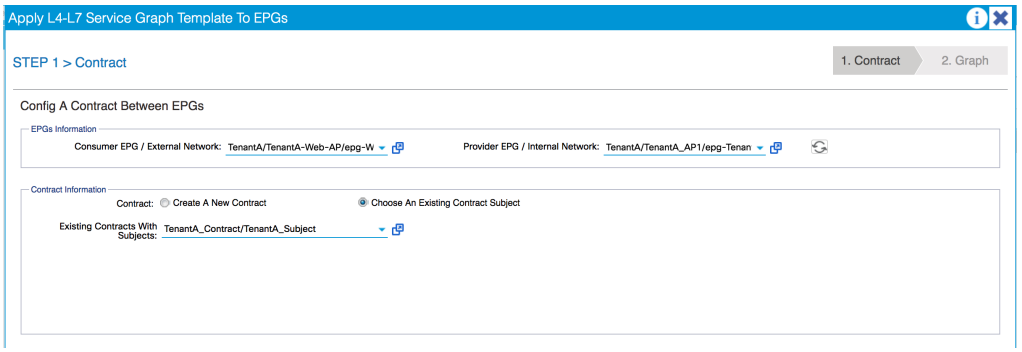
6. Click **Finish**.
7. The next step is to create a service graph template. From the L4-L7 Services (under the tenant), select “L4-L7 Service Graph Templates”. Click on the Actions menu, and select “Create L4-L7 Service Graph Template”.
8. Set the graph name (Tenant-A-Provider) and drag the TenantA/A10-vThunder object between the consumer EPG and the provider EPG. Select the profile in the window that pops up (Basic Web Server).



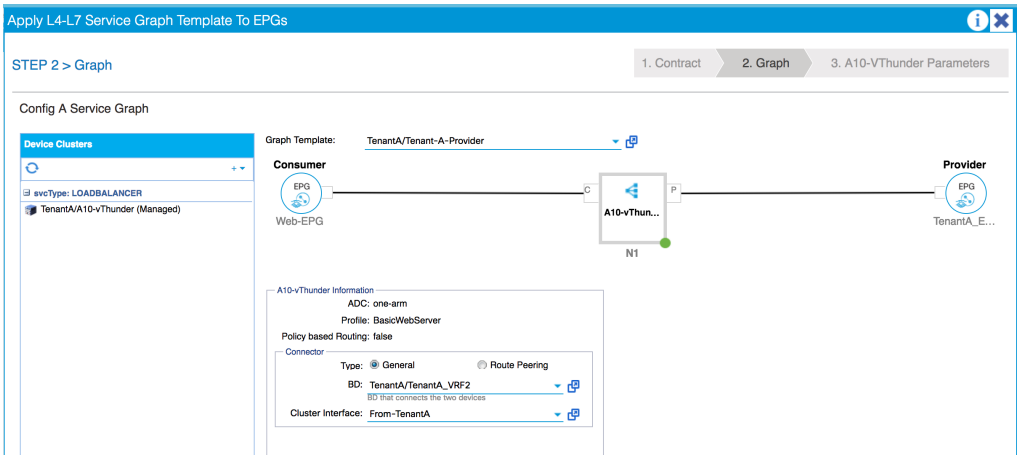
9. Click **Submit**.
10. Right click on the graph name, and select "Apply L4-L7 Service Graph Template."



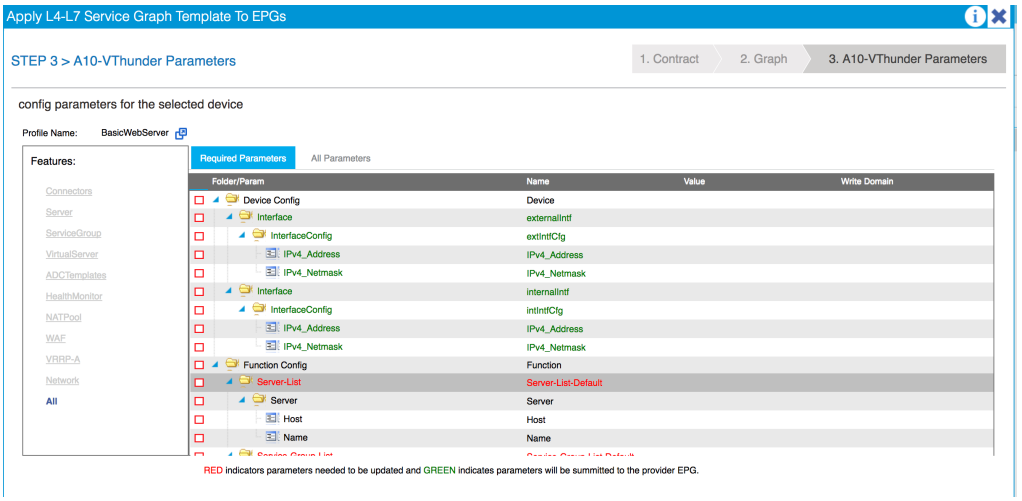
11. Set the consumer and provider EPGs. The provider EPG will be TenantA, and the consumer will be the Web AP. Set the contract to be the TenantA_Subject contract we created in chapter 2.



12. Click Next
13. Set the cluster interface.



14. Click Next
15. We need to set some values for the vThunder here. You can edit the fields by double clicking on them, typing in the relevant details and clicking “Apply.” If any required fields are not completed, you will receive a message when you click on Finish.



16. Click Finish.

How it works…

In this recipe, we created a second EPG within TenantA. We created a new subnet and inserted an A10 Thunder virtual appliance between the two EPGs, one of which is the provider, the other is the consumer.

There's more…

Refer to the following PDF for the full walkthrough:

`https://www.a10networks.com/sites/default/files/A10-DG-16143-EN.pdf`

Deploying the ASAv

Deploying the ASAv consists of two parts. Firstly, we need to upload the ASAv device package and secondly, we need to deploy the OVA file to create the ASAv guest on the ESXi hosts. This is the same method as the A10 deployment in the previous recipe.

How to do it…

We start by installing the device package, following the same steps as the "Installing Device Packages" recipe earlier.

The ASAv device package and OVA file do require specific entitlements from Cisco before you can download them.

1. From L4-L7 services, select packages and click on "Import Device Package."
2. Browse to the package location and upload it. The package name will be something like "asa-device-pkg-1.2.4.8".
3. The new device should be visible from the L4-L7 Service Device Types window in the GUI.
4. From vCenter, choose the option to deploy the OVA template (File > Deploy OVF Template).
5. Select the appropriate download option. If you are using a standalone ESXi server, use the file name "asav-esxi.ofv," or if you are using vCenter use the file named "asav-vi.ovf."
6. Follow the installation wizard, accepting the terms and conditions, and set the hostname, management IP address, firewall mode.

1. OOB (Out-of-Band) management must be enabled!

7. Click Finish to deploy the ASAv.
8. Power on the ASAv virtual machine.
9. Connect to the console of the virtual machine and configure a username and password.

    1. `username admin password admin123 encrypted privilege 15`

10. Enable HTTP access

    1. `http server enable`

```
http 0.0.0.0 0.0.0.0 management
```

2. From the ACI GUI, select the tenant which will be using the firewall.
3. Go to L4-L7 Services
4. Right click on L4-L7 Devices
5. Click "Create L4-L7 devices". This will open the device import wizard. Use the following settings: **General** Managed: Selected Service Type: Firewall Device Type: Virtual Mode: Single Function: Goto Select the appropriate VMM domain, device package, and model. **Connectivity** Out-Of-Band **Credentials** Username: admin (or username set in step 9) Password: admin123 (or password set in step 9) **Device 1** Management IP address: The IP address assigned to the ASAv during setup. Management Port: https Device Interfaces: Assign the appropriate interfaces to the desired path. **Cluster** Set the consumer and provider contract information and interfaces.
6. We need to create a Service Graph Template like we did with the A10 device in the previous recipe, but with a couple of differences. Start by right clicking on "Function Profiles", which is in the L4-L7 Services menu option within the tenant.
7. Select the option to "Create L4-L7 Services Function Profiles Group". Give the group a name.
8. The new group should appear underneath. Right-click on the newly created group, and select "Create L4-L7 Services Function Group".
9. As we will be setting the firewall up on router mode (GoTo mode in ACI-speak) we need to select "WebPolicyForRoutedMode" in the profile drop-down.
10. Set the IP addresses for the internal and external interfaces, along with the security level.
11. Because of the profile we chose, HTTP and HTTPS will be allowed through the firewall, but here we can also add any new rules we want to implement.
12. Click **Submit** when you have set the appropriate options.
13. Create the Service Graph Template, give it a name, drop the firewall cluster object between the two EPGs. Set the firewall to be Routed, and set the function profile to the one we created a moment ago.
14. Apply the **template**.
15. The steps here are, again, very similar to the A10 set up; we set the consumer and provider EPGs, create a new contract and set a filter if we want to. It is advisable not to set the filter here, instead, we select "No Filter (Allow All Traffic)", which means that all traffic filtering performed is decided upon by the contents of the access list set earlier (step 20).
16. Click **Next**

17. Select the appropriate BDs (Bridge Domains) and cluster interfaces.
18. Click **Next**
19. Set the config parameters.
20. Click **Finish**.

We will be setting up the ASAv in Chapter 10.

How it works…

Configuring the ASA is very similar to the A10, with a few additional steps, in fact, once you have added one device package and set up the corresponding device, each additional device, no matter the vendor, is all very similar, the hardest part is getting access to the package file and the virtual machine! The steps are very different than adding a hypervisor, though, so let's run through that another time, by adding OpenStack to our environment.

There's more…

`http://www.cisco.com/c/en/us/support/docs/cloud-systems-management/application-policy-infrastructure-controller-apic/200428-ASAv-in-GoTo-L3-Mode-with-the-Use-of-A.html`

Integrating with OpenStack

The easiest way to get started with OpenStack is, by far, the Mirantis OpenStack bundle. It is easy to set up and comes with everything you need to get started with OpenStack if this is the first time you have put your toe in this particular pool. You will still require the correct Cisco subscription to get the package file.

How to do it…

1. Install the Mirantis OpenStack virtual machines, by following the guide at `https://www.mirantis.com/how-to-install-openstack/`. This takes about 20 minutes or so to complete and once completed will give you one “FUEL” controller and three OpenStack nodes to play with.
2. Download the FUEL plugin from `http://plugins.mirantis.com/repository/a/c/aci_opflex/` Copy the plugin to the OpenStack controller:

1. **scp aci_opflex-9.0-9.0.12-1.noarch.rpm root@10.20.0.2:/tmp**
   root@10.20.0.2's password: **r00tme**
   aci_opflex-9.0-9.0.12-1.noarch.rpm
   100% 615KB 615.1KB/s   00:00   Stuarts-iMac-2:Downloads stu$

2. Login and install the plugin:

   1. **ssh root@10.20.0.2**
      root@10.20.0.2's password: **r00tme**
      [root@fuel ~]#
      [root@fuel /]# **fuel plugins --install /tmp/aci_opflex-9.0-9.0.12-1.noarch.rpm**

2. Upload the .rpm file, downloaded from Cisco.

   1. **scp aci_opflex-9.0-9.0.12-1.noarch.rpm 10.20.0.2:/var/www/nailgun/plugins/aci_opflex-7.0/repositories/ubuntu/**

2. Update the packages file in the above directory

   1. **dpkg-scanpackages -m . /dev/null | gzip -9c > Packages.gz**

   (you may need to install dpkg-dev for the above command to work).

2. From the FUEL GUI, create a new OpenStack environment.
3. Name the new environment and select the relevant OpenStack release. Click **Next**.
4. Select either KVM or Qemu as the hypervisor. Click Next.
5. On the Networking Setup page, select Neutron with VLAN or VLXLAN segmentation (depending on your requirements). Click Next.
6. Assign a minimum of 1 Controller node and 1 Compute node.
7. Click on the Settings icon for the nodes and enable the ACI OpFlex plugin. Complete all the fields, making sure that the driver mode is set to ML2.
8. Click Next through the remaining fields until you finish the environment creation.

How it works…

The new environment should appear as a new tenant within the APIC GUI. The plugin links the Neutron component of OpenStack to ACI, nodes added to OpenStack will appear as nodes within the APIC.

There's more…

If you would like to read more on getting started with OpenStack, read this link: `https://www.mirantis.com/how-to-install-openstack/`

For a great video on integrating ACI and OpenStack, watch this video: `https://www.youtube.com/watch?v=pWMXTb237Vk`

Integrating with F5

F5 stopped support of the standard ACI package back in December 2016. This is not to say, though, that they have shunned ACI, quite the opposite. They continue to embrace ACI, but getting the two technologies to co-exist works slightly differently.

In this recipe, we will create a basic virtual server on the F5 BIGIP and set this up on our APIC.

Getting ready…

Before we can configure the APIC, we need to make sure that the F5 components are running and configured.

For this, we need iWorkflow (version 2.0.0 or above) and a BIG-IP appliance. I am using **iWorkflow-2.1.0.0.0.10285-scsi.ova** and **BIGIP-13.0.0.0.0.1645.ALL-scsi.ova**. These are available fo download from the F5 website (`https://downloads.f5.com`) and you can download the free trials.

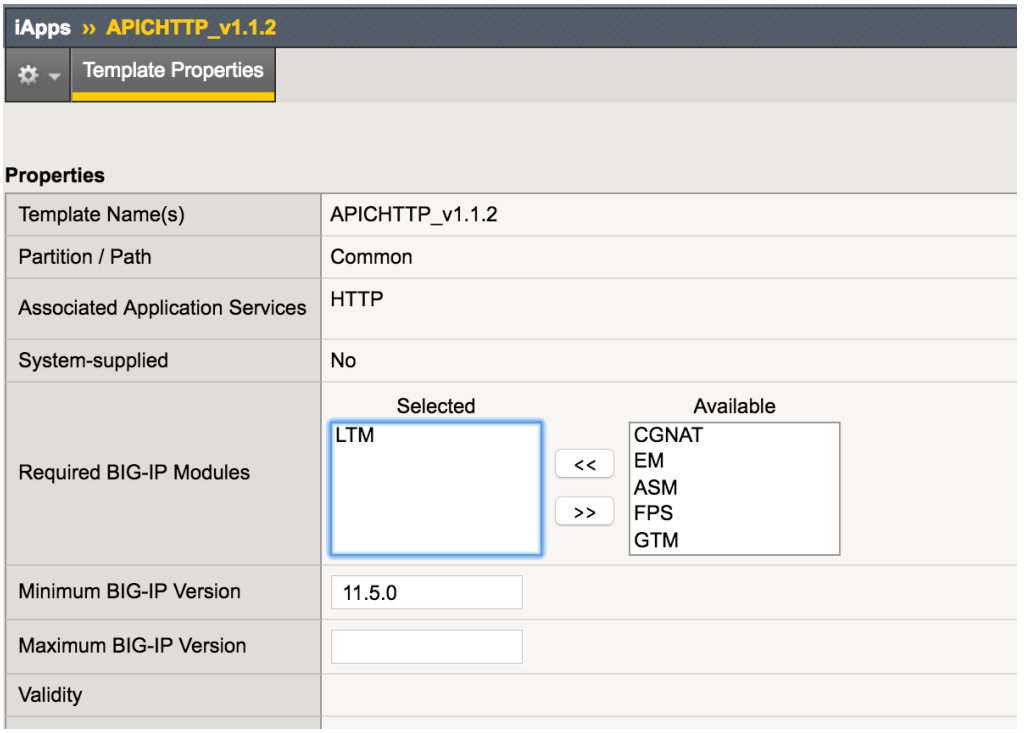
These need to be imported into vCenter and the VMs started. Once started, run through the configuration wizards to set up the basic IP addressing, NTP and DNS configuration. The BIGIP device should be added into iWorkflow.

How to do it…

1. From the F5 Big-IP GUI, navigate to **iApps > Templates**. Select the inbuilt "f5.http" template and scroll to the bottom of the screen. Click on "Copy" and give the template a name.

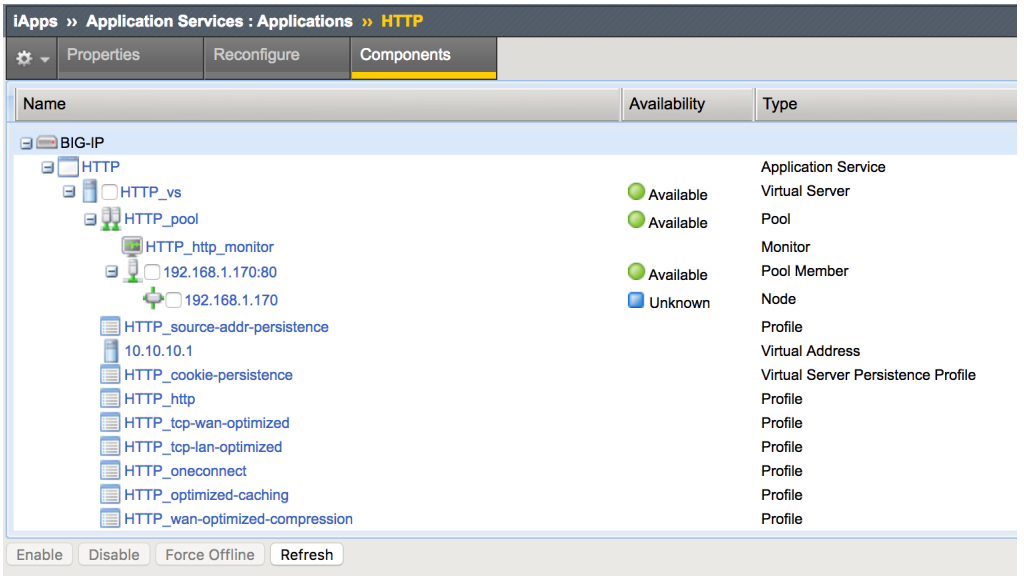
1. Make sure you name it correctly as it must be in the format "name_v1.1.1" or "name.v1.1.1", so "ACIHTTP_v1.1.2" will work fine, whereas "ACIHTTP-v1.1.2" will not work as this does not follow the iWorkflow naming convention.

2. Click on Finish. The resulting template will look like this:

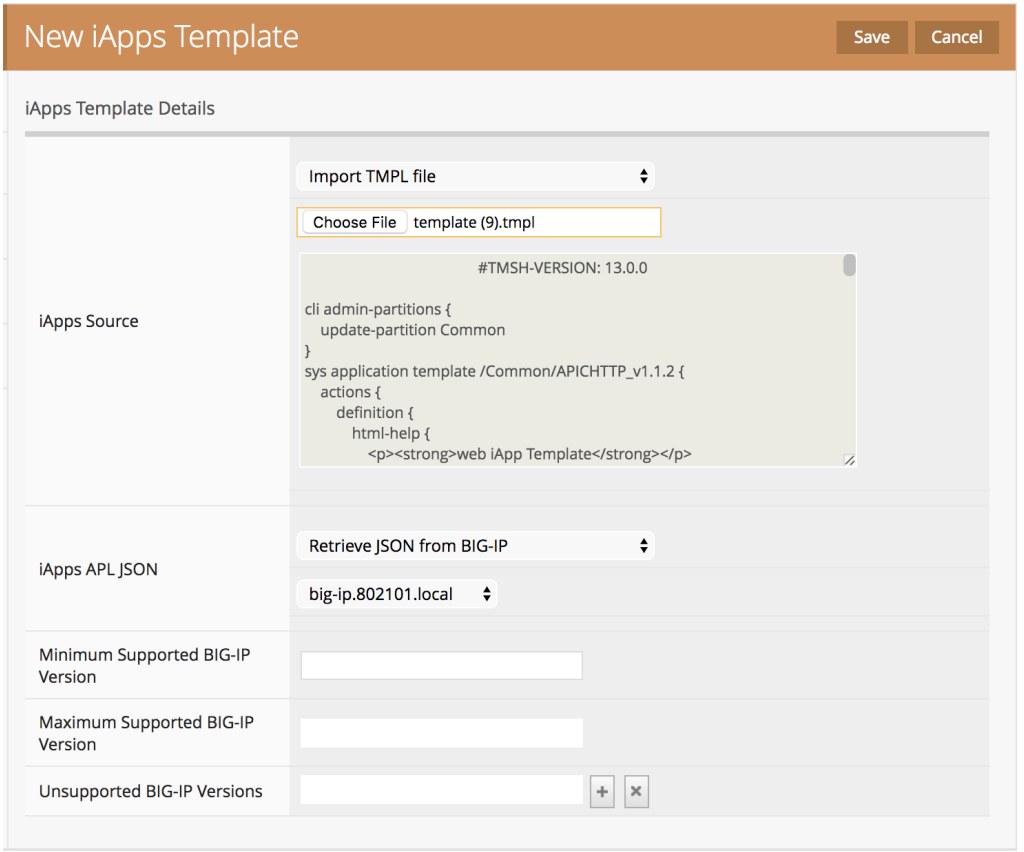


3. Navigate to **iApps > Application Services** and create a new application. I created one called "HTTP" and used the template created in step 1. There are some required fields, such as the virtual server IP address, port, and the DNS address.

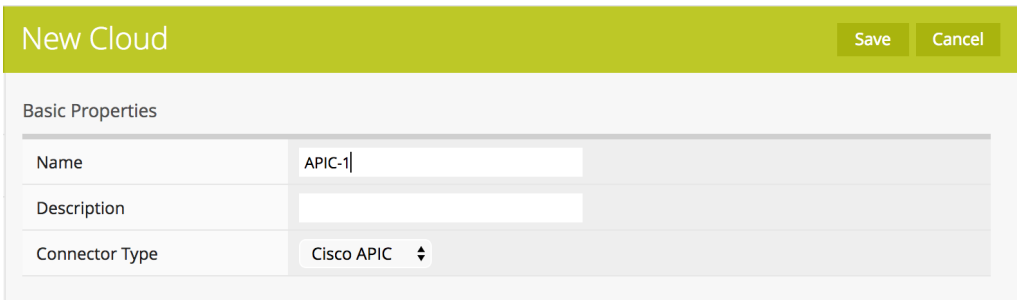
4. Click on Finished. The resulting service should look a little like this:



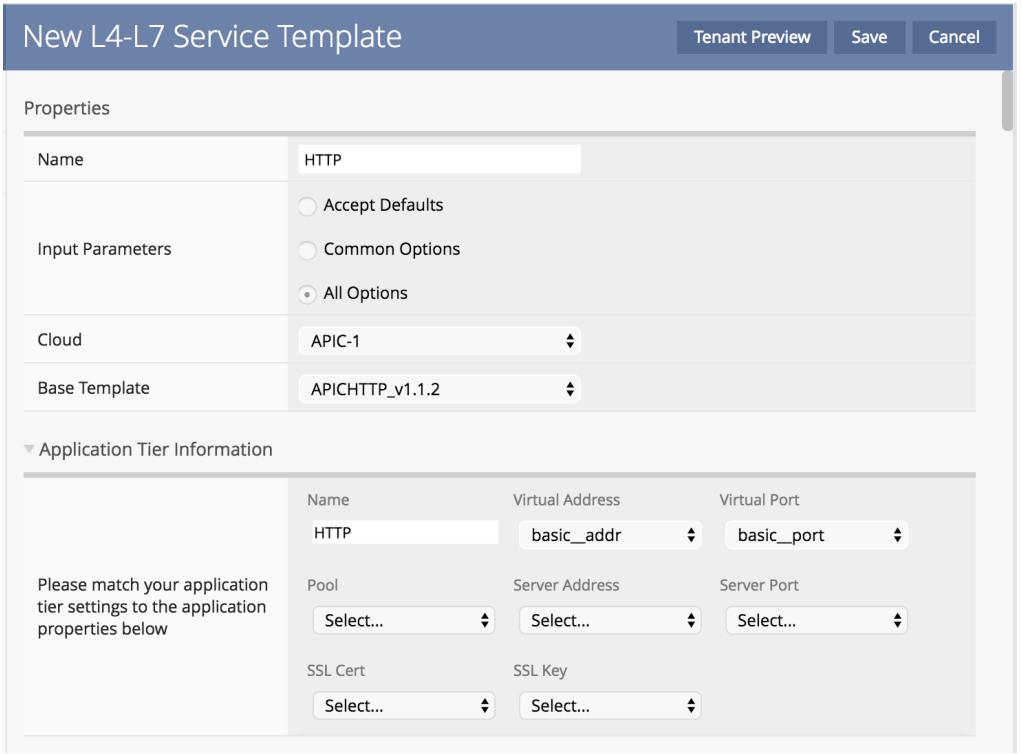
5. Return to **iApps > Templates**. Select the template created in step 1 and click on Export. Download the template file.
6. In iWorkflow, navigate to Clouds and Services > iApps Templates.
7. Create a new template, selecting the template downloaded in step 5. Use the JSON file from BIGIP device, selecting it from the drop-down menu.



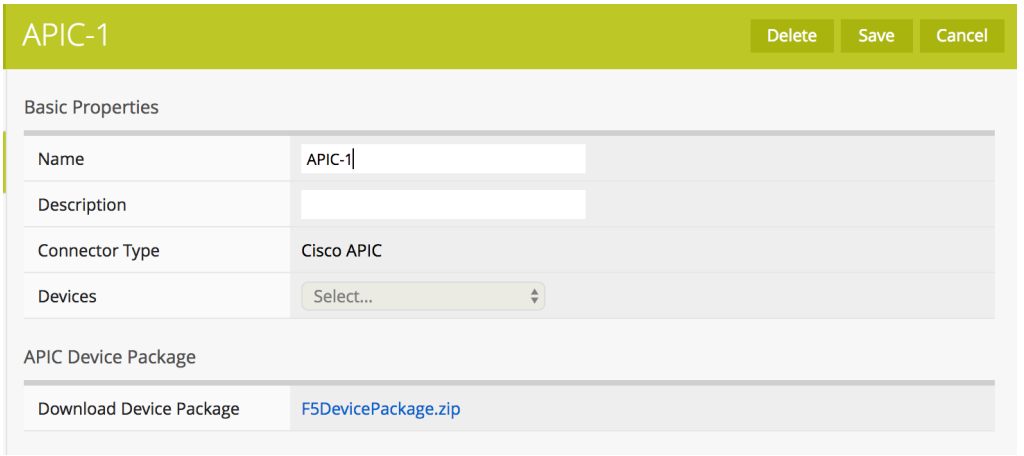
8. Click Save.
9. Navigate to Clouds and create a new cloud, setting the connector type to “Cisco APIC”.



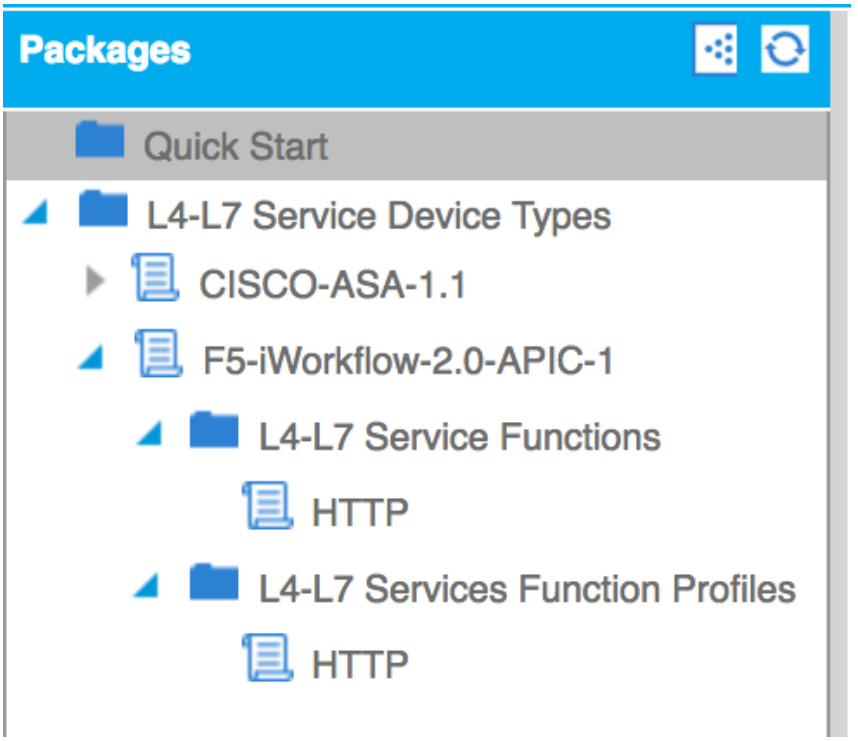
10. Create a Service template, setting the Cloud to the one created in the previous step, and the template to that created in step 7. Depending on the template you may need to enter some more information, such as the name, Virtual Address and Virtual Port.



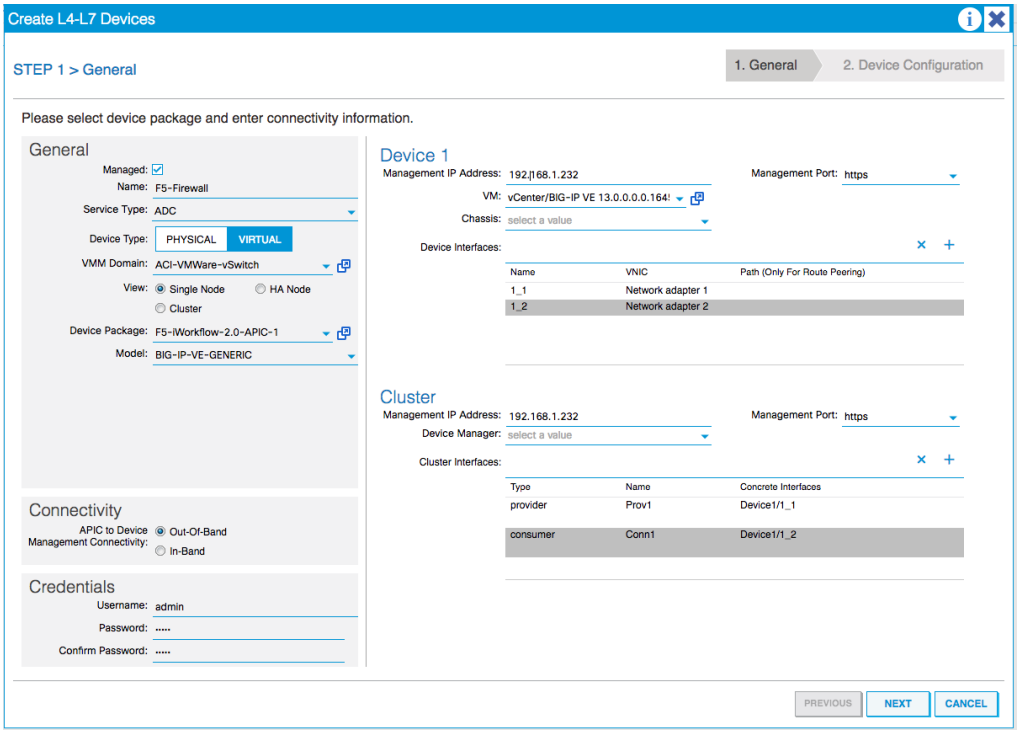
11. Click Save.
12. Go back to Clouds, and select the APIC cloud created in step 9.



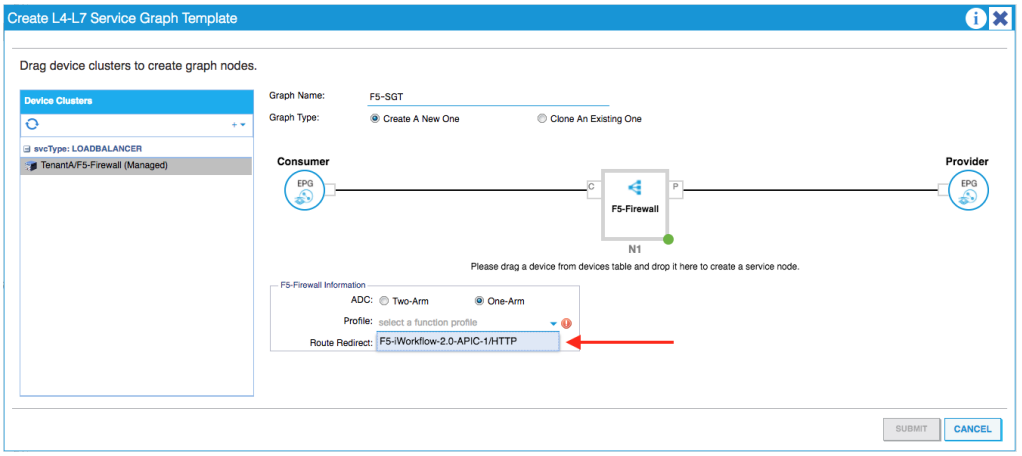
13. Click on "F5DevicePackage.zip" to download it.
14. Import the device package into the APIC (**L4L7 Services > Packages > Import a Device Package**).



15. Navigate to the tenant and import the device (**L4-L7 Services > L4-L7 Devices**, right-click on it and select "**Create L4-L7 Device**").
16. Fill in the details.

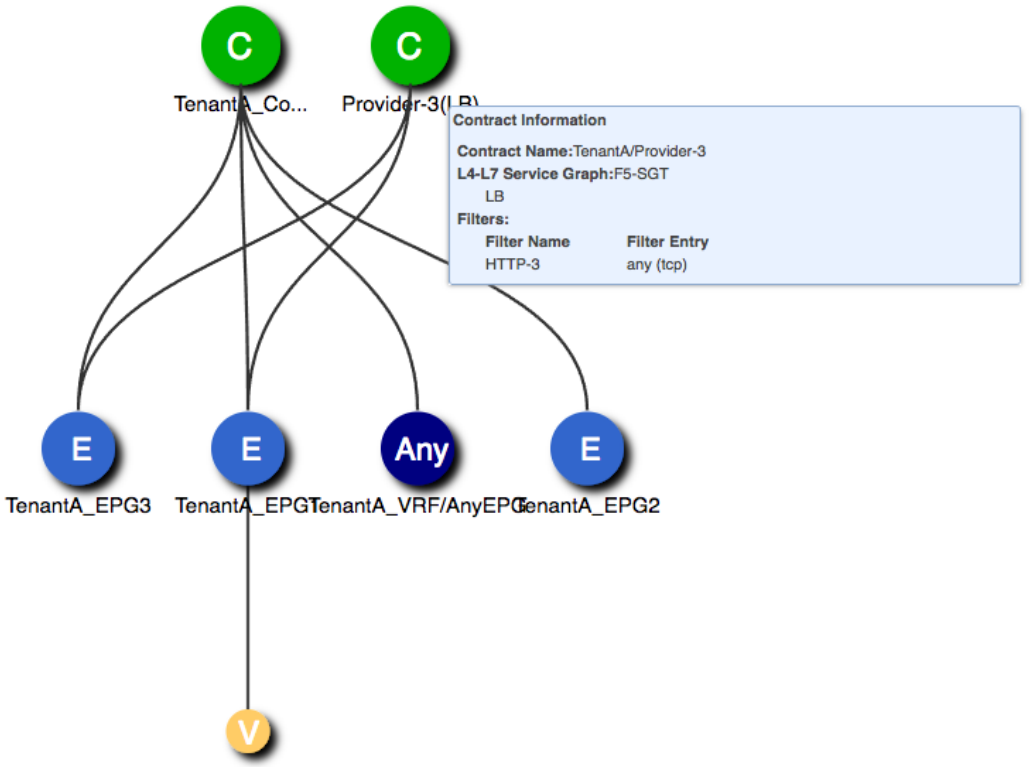


17. Click Next.
18. Click Finish.
19. From the L4-L7 Service Graph Templates menu, create a new service graph template.
20. Name the graph and then drag the F5 object between the Consumer and Provider EPGs.
21. Select the HTTP function profile.



22. Click Submit.

After that, you would create the contract between the provider and consumer, setting the service template to be the one created in the steps above.



Above, we can see a contract between TenantA_EPG3 (Provider) and TenantA_EPG1 (Consumer).

There's more…

If this recipe has whetted your appetite, you can read the full set up document:

`https://support.f5.com/kb/en-us/products/iworkflow/manuals/product/iworkflow-cisco-apic-administration-2-0-0/6.html`

There are a couple of good videos on YouTube showing how to integrate F5 with ACI:

`https://www.youtube.com/watch?v=VTE7Ei4Nj6c`

```
https://www.youtube.com/watch?v=gBAQeMUwgJE
```

Integrating with Citrix NetScaler

ACI works with Citrix NetScaler MPX, SDX, VPX and the 1000V series. Integrating the two is very straight forward, and if you have run through either the A10 or ASA recipes, then integrating NetScaler will not be any different.

Getting ready…

You will need to download the package from Citrix. This is restricted content, so you may need to speak to your account manager.

```
https://www.citrix.com/downloads/netscaler-adc/components/netscaler-device-package-for-cisco-aci.html
```

How to do it…

The documents from Citrix highlight how easy it is to integrate NetScaler and ACI. Their document lists the seven steps.

1. Install the NetScaler devices.
2. Configure the management interfaces and credentials.
3. Install the NetScaler device package onto the APIC.
4. Create a device cluster within APIC to manage the device.
5. Define the logical interfaces and define a VLAN pool.
6. Define the service graph.
7. Associate the service graph and the device cluster with a logical device and add to the application profile contract.

We have covered all these steps in the previous recipes.

There's more…

For the official Citrix documentation, refer to these links:

```
https://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/implementing-cisco-application-centric-infrastructure-with-citrix-netscaler
```

-application-delivery-controllers.pdf

https://www.citrix.com/blogs/2015/06/04/citrix-netscaler-and-cisco-aci-how-it-all-works/

4
Routing in ACI

In this chapter, we will look at:

- Creating a DHCP relay
- Utilizing DNS
- Routing with BGP
- Configuring a layer 3 outside interface for tenant networks
- Associating Bridge Domain with External Network
- Using Route Reflectors
- Routing with OSPF
- Routing with EIGRP
- Using IPv6 within ACI
- Setting up Multicast for ACI tenants
- ACI transit routing and route peering

Introduction

ACI works extremely well by itself, however, as the John Donne once said; “no man is an island”. Here we will look at how to extend ACI through routing. Before we delve into the world of routing, we will look at helping our nodes get IP addresses by creating a DHCP relay and help them to resolve names through DNS. Once we have covered these, we will look at configuring BGP (Border Gateway Protocol), OSPF (Open Shortest Path First), and EIGRP (Enhanced Interior Gateway Routing Protocol). We will then look at how easy IPv6 and Multicast support is within ACI, before finishing the chapter looking at transit routing.

Creating a DHCP relay

By default, ACI-wide flooding is disabled. Because flooding is disabled, connecting to a DHCP server is only possible if it is in the same EPG as the client (as flooding within a bridge domain is enabled).

The options, therefore, are to have one DHCP server per-EPG, which would be wasteful on "compute" resources and on administrative time, or to use a DHCP relay and have one, central, server.

In this recipe, we will be setting up a DHCP relay.

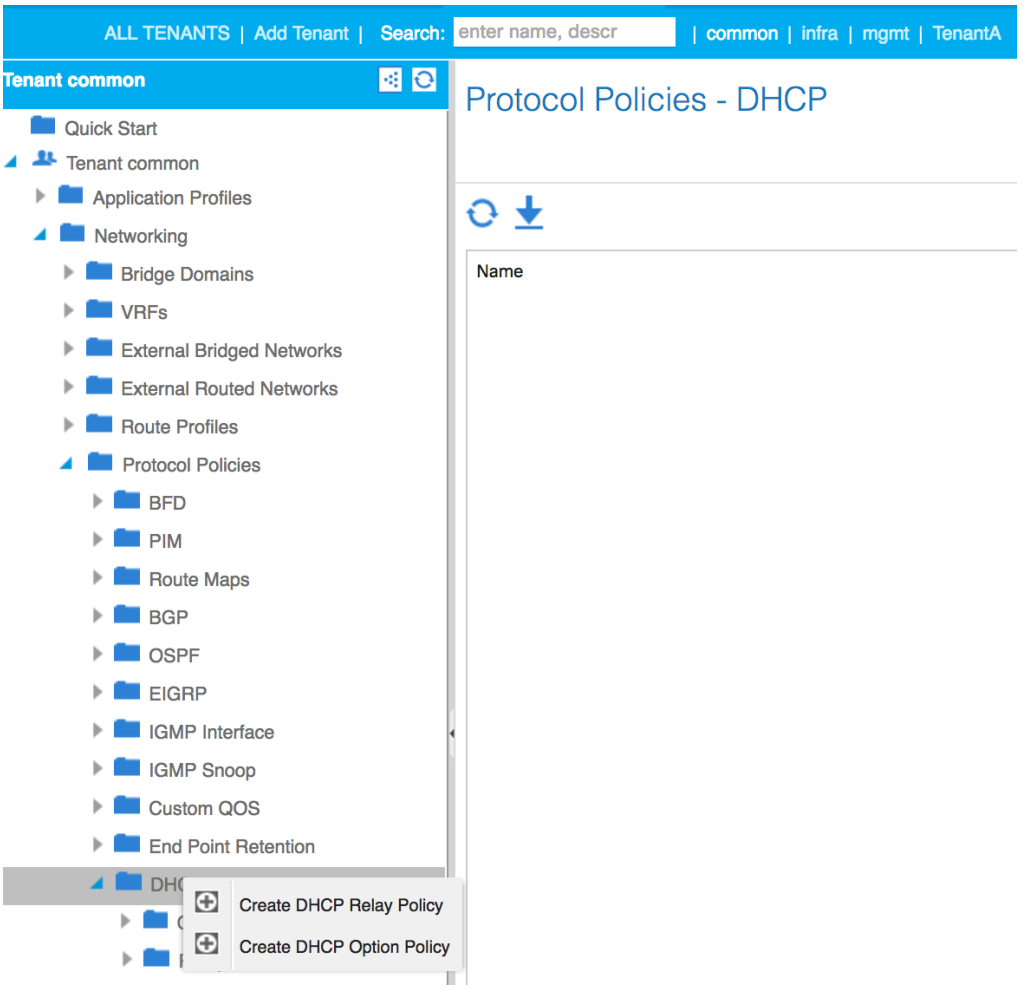
How to do it…

We will be using the Common tenant for this (first), as it is best practice to place resources within this tenant if they are to be shared across multiple tenants. Using the Common tenant is not the only way, though, we can also use the Fabric Access policies to achieve the same end-goal or the Infrastructure tenant.

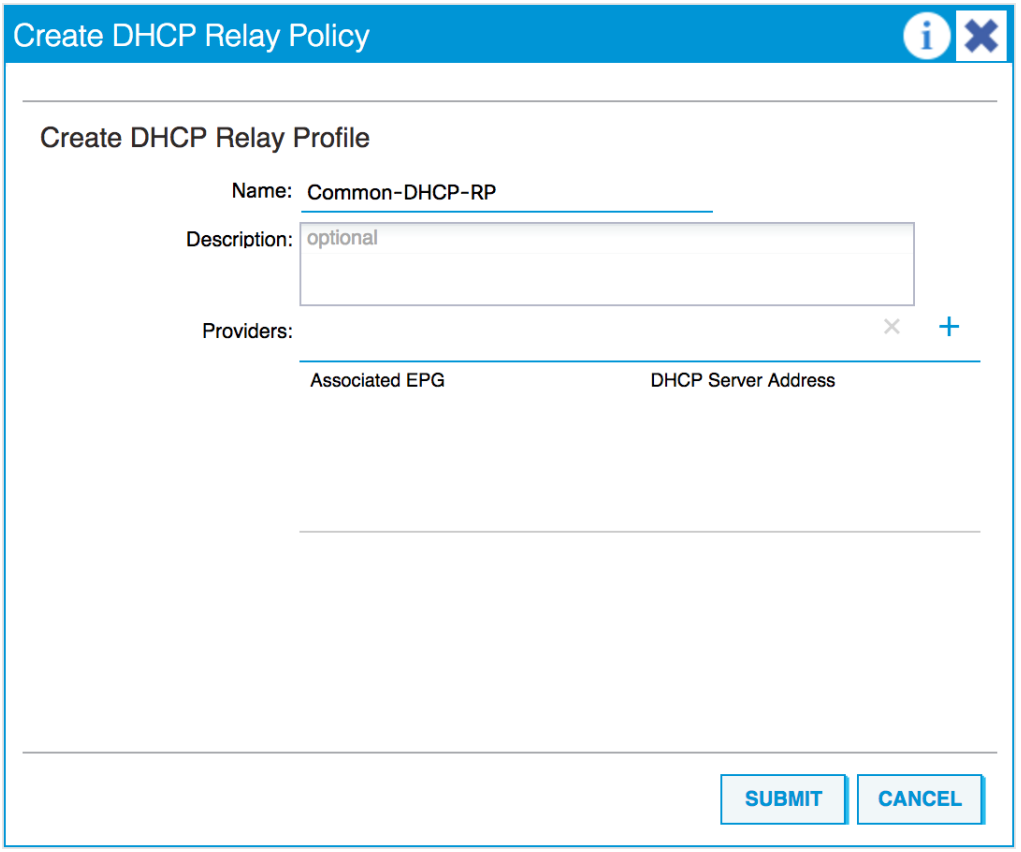
Why would you use one tenant over the other? Using the Common tenant means that the DHCP relays can be used by *any* tenant. If we use the Infrastructure tenant, the DHCP relay policies are *selectively* exposed to the other tenants. Configuring a Global DHCP relay under the Fabric Access means that any tenant can use the DHCP relays and we have a higher level of control.

Creating a DHCP Relay using the Common tenant

1. From **Tenants** > **Common** > **Networking** > **Protocol Policies** > **DHCP**, right click on Relay Policies and select **Create DHCP Relay Policy**.

1. 

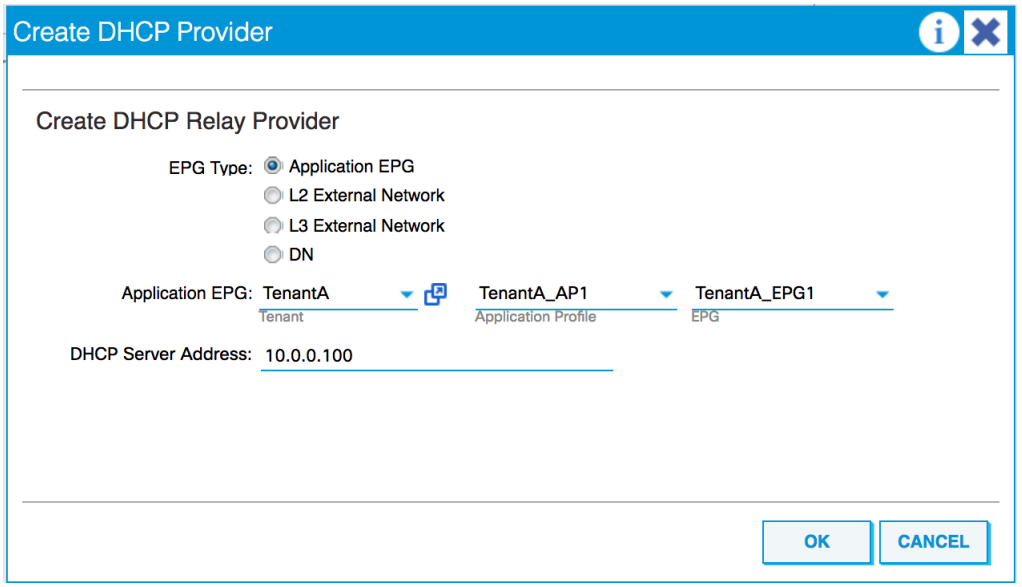
2. Name the policy, and click the plus sign next to Providers.



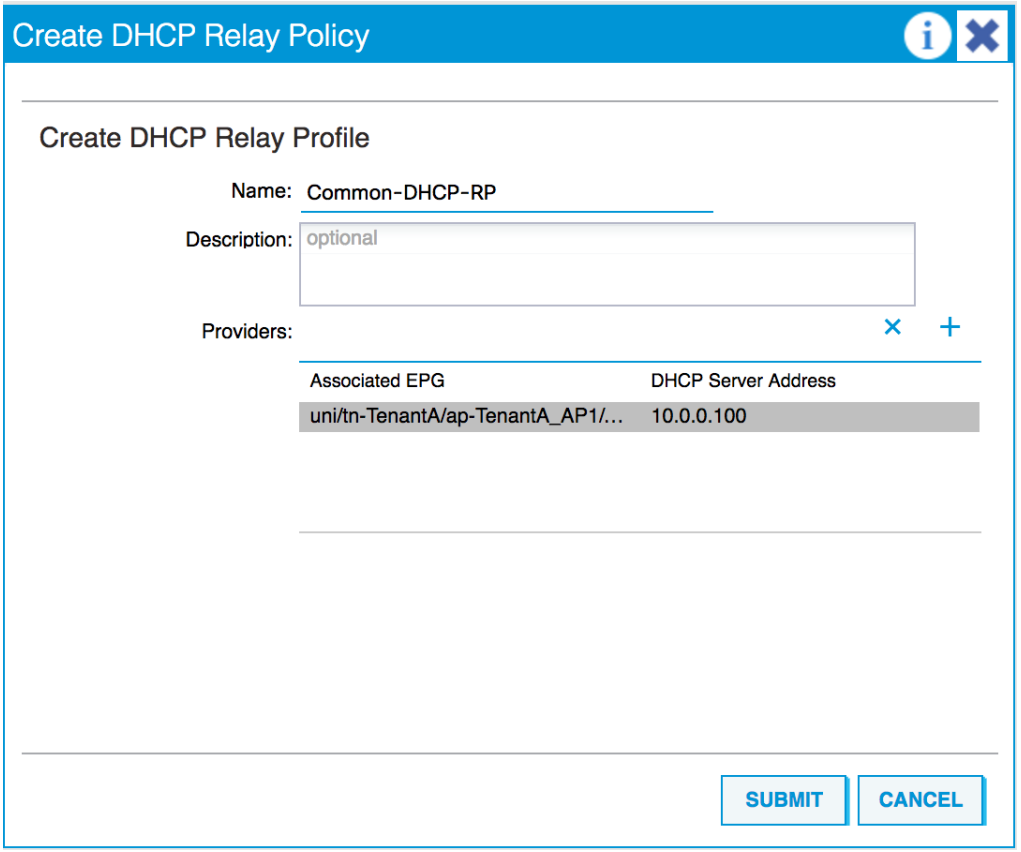
3. Select the EPG-type (which should be “Application EPG”).
4. Select Application EPG where the DHCP server is located (for this we are going to use one that would be situated within TenantA).

1. Notice that the Common tenant has access to all of the tenants, their application profiles, and their EPGs.

2. Enter the DHCP Server address.

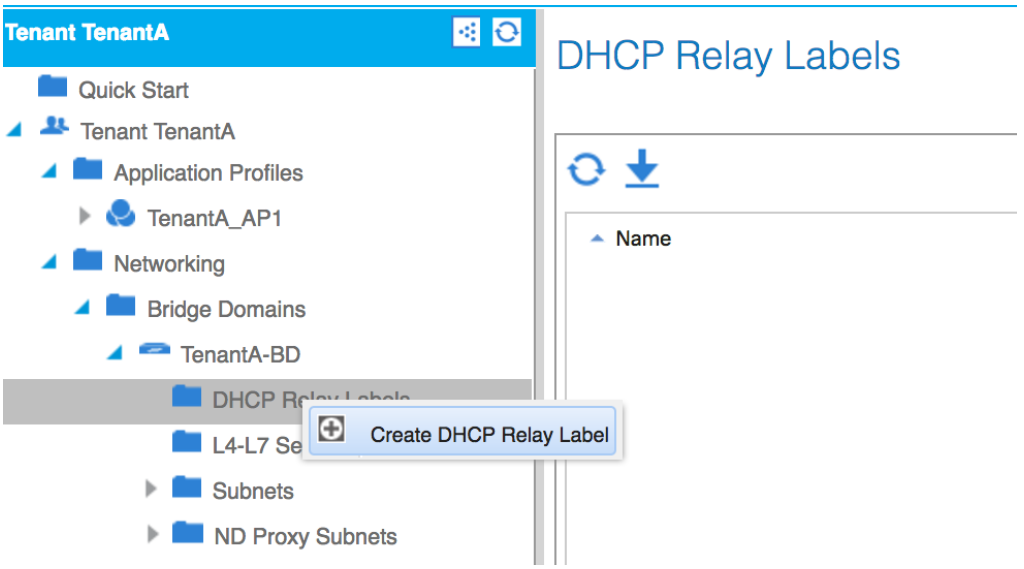


3. Click **OK**.
4. Click **Submit**.

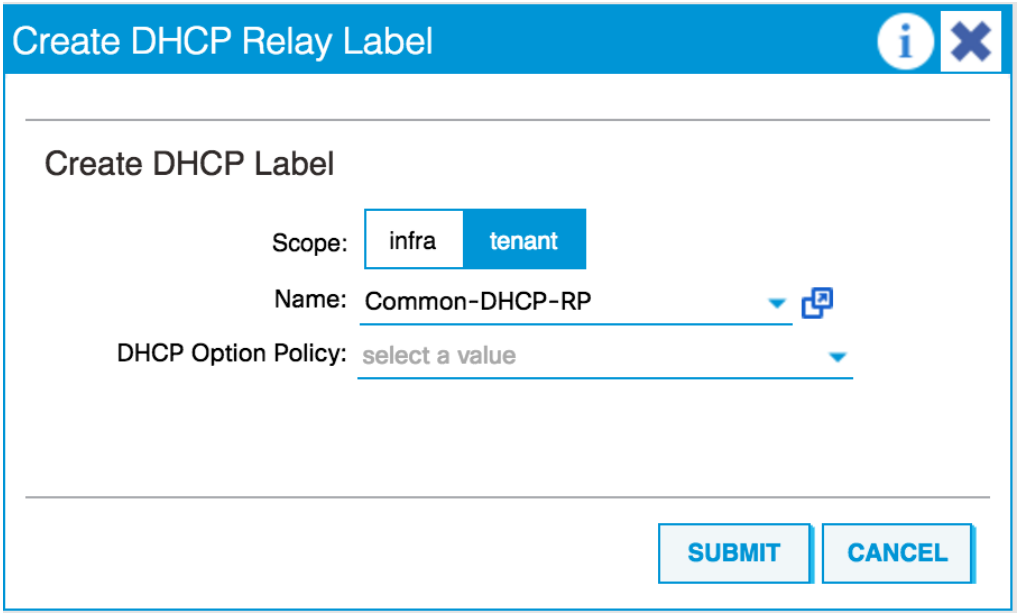


Next, we need to create the labels.

5. From TenantA, go to **Networking** > **Bridge Domains** > **TenantA-BD**. Right click on **DHCP Relay Label**s and select **Create DHCP Relay label**.



6. Set the scope to "**tenant"**.
7. Select the Tenant DHCP Relay policy we created earlier.

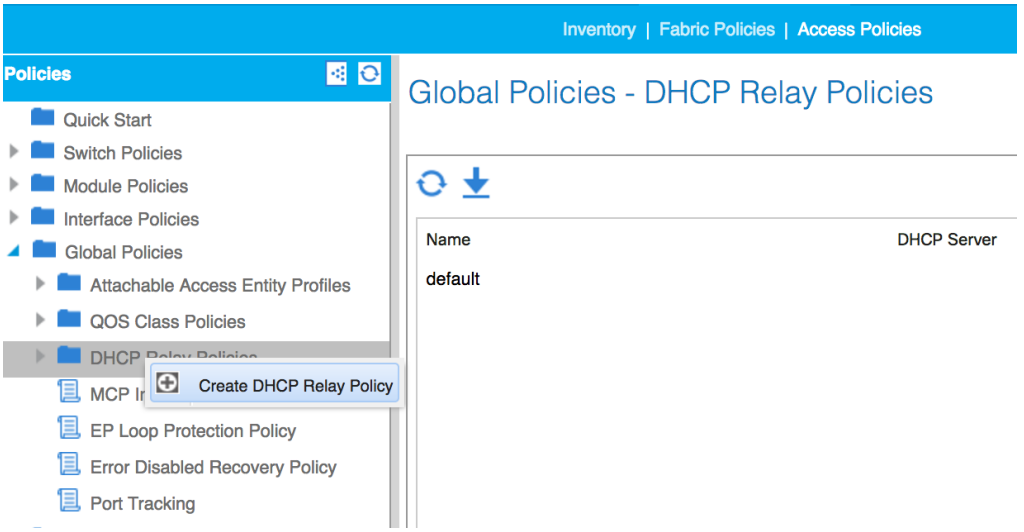
1. 

2. Click Submit.

The second method we will look at is creating an access policy for a global DHCP relay.

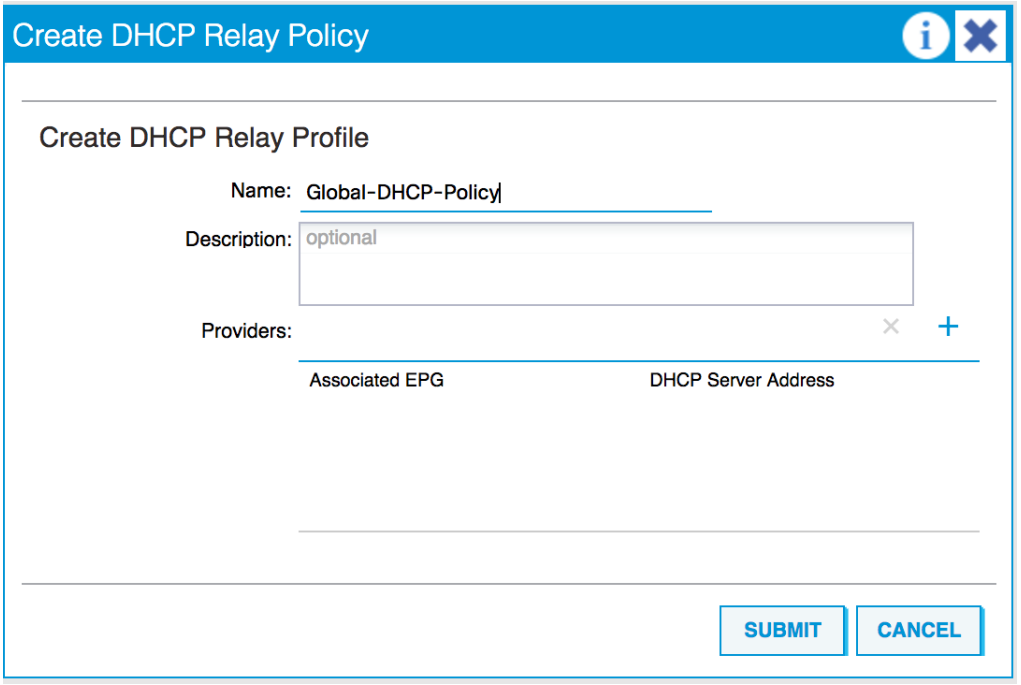
Creating a Global DHCP Relay

1. From Fabric > Access Policies > Global Policies > DHCP Relay Policies, right-click and select **Create DHCP Relay Policy**.

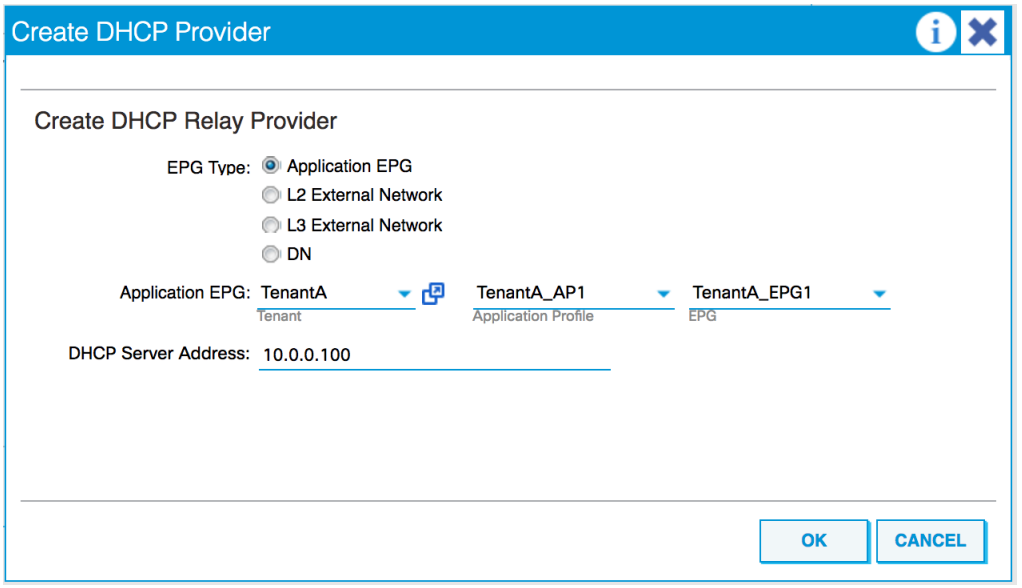


1.

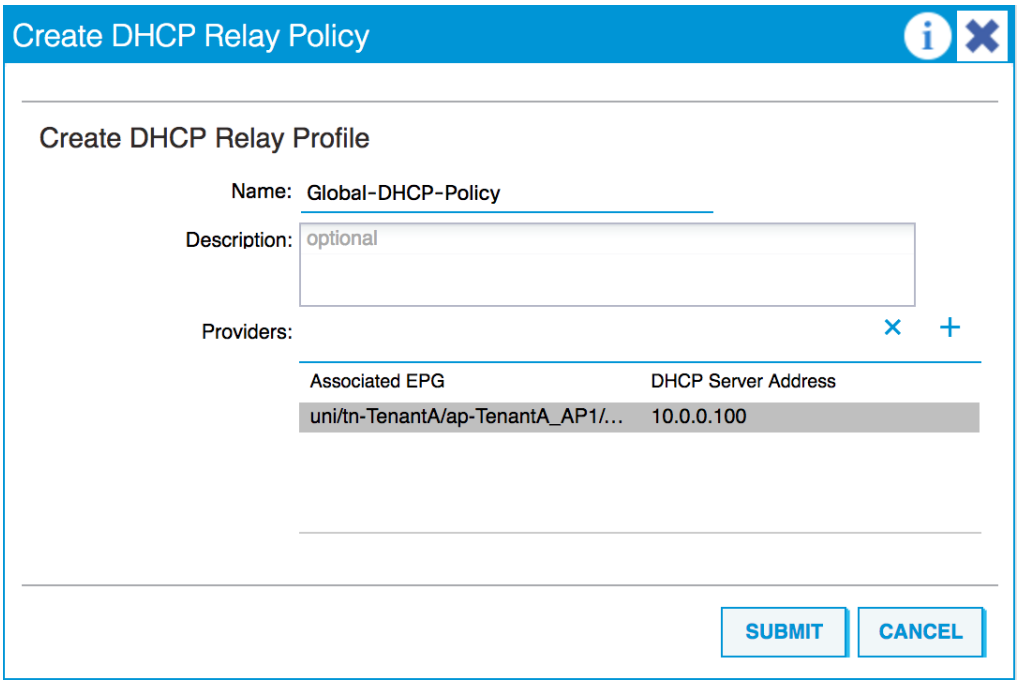
2. Name the policy and click the plus sign next to Providers.

1. 

2. Select the EPG-type (Application EPG).
3. Choose the application EPG where the DHCP server is.
4. Enter the DHCP server address.

1. 

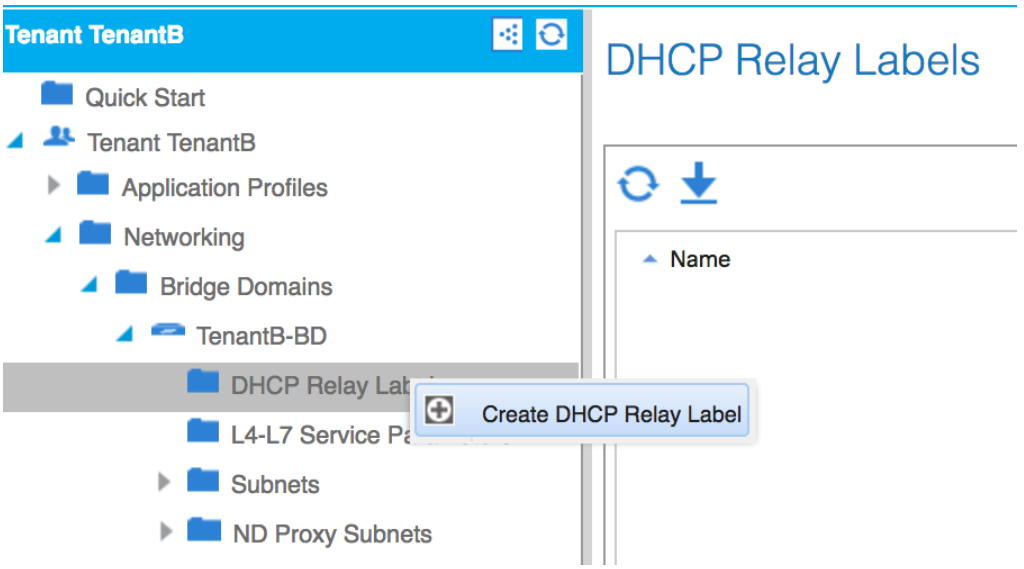
2. Click **OK**.

1. 

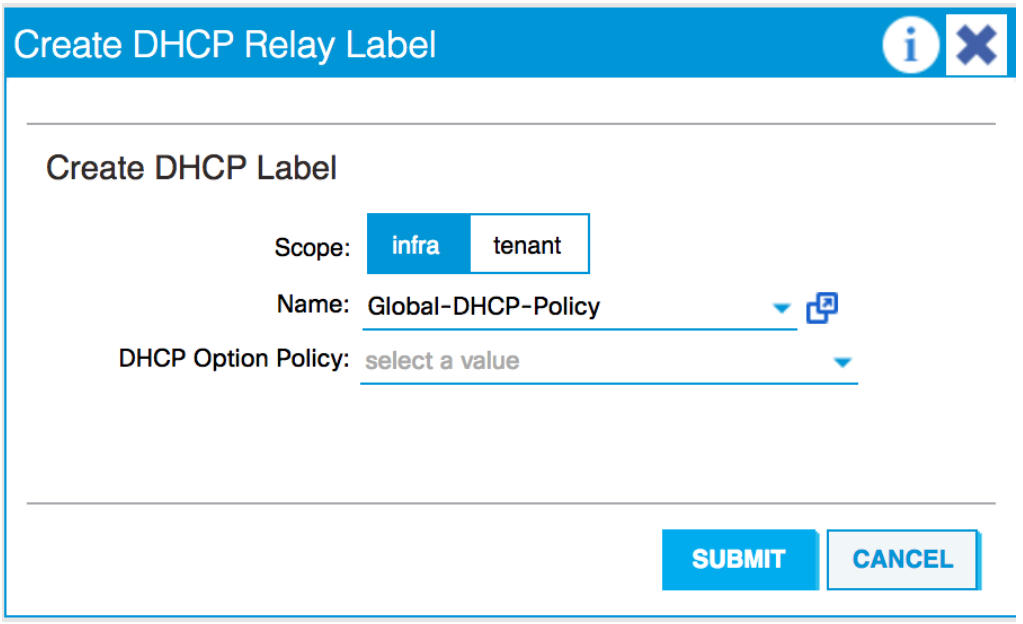
2. Click **Submit**. We need to create another set of labels here, as we did in the first method.

1. I have created a second tenant for this, called TenantB. Jump ahead to chapter 7 if you want to create it yourself using the REST client, or drop back to chapter 2 if you want to set it up by hand (just replace "TenantA" with "TenantB" and use the subnet 20.0.0.1/24).

2. From **TenantB**, select **Networking** > **Bridge Domains** > **TenantB-BD**.
3. Right-click **DHCP Relay Labels** and select **Create DHCP Relay Label**.

1. 

2. Set the scope as **infra**.
3. Select the Global DHCP Relay Policy created earlier.

1. 

2. Click **Submit**.

How it works…

For this to work between EPGs, we would need to have routing in place, as well as a contract to permit the DHCP traffic. Refer to *Chapter 2* for how to create contracts.

There's more…

In a multitenancy environment, tenants will require separate DHCP servers, otherwise, they could receive incorrect IP addresses, gateway addresses, DNS server addresses and much more.

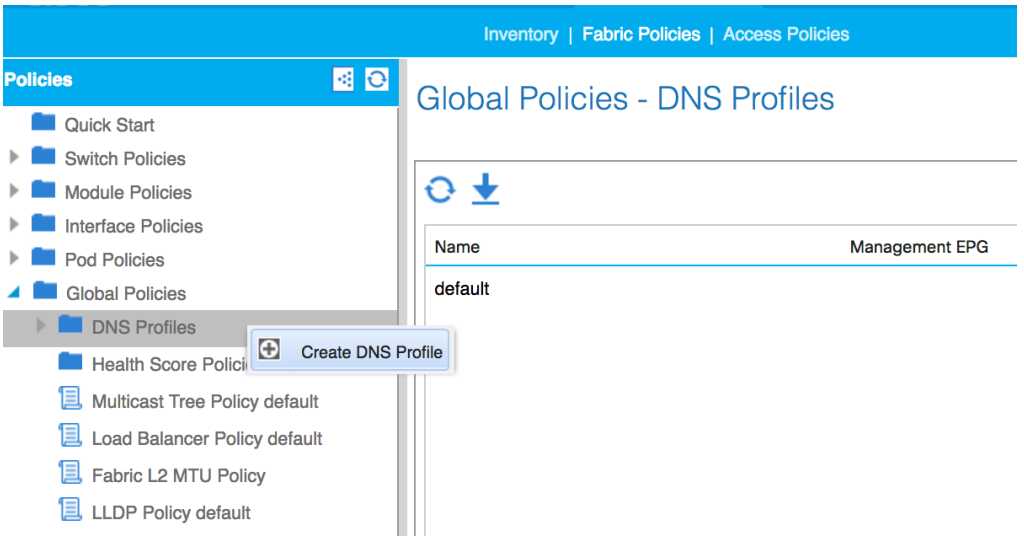
Utilizing DNS

A DNS policy will be required to connect to external servers by their name, rather than their IP address. Such services could be AAA, Radius, or vCenter.

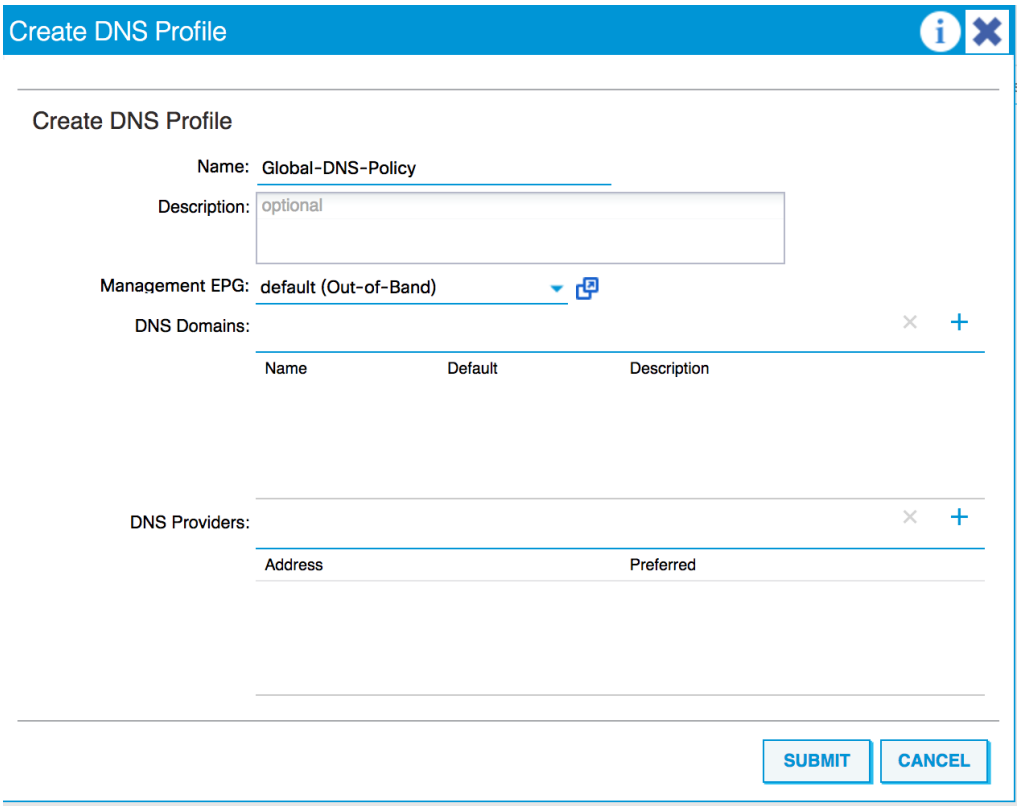
The DNS service policy is a shared policy, in that any tenant and VRF that uses this service must be configured with the particular DNS profile label.

How to do it…

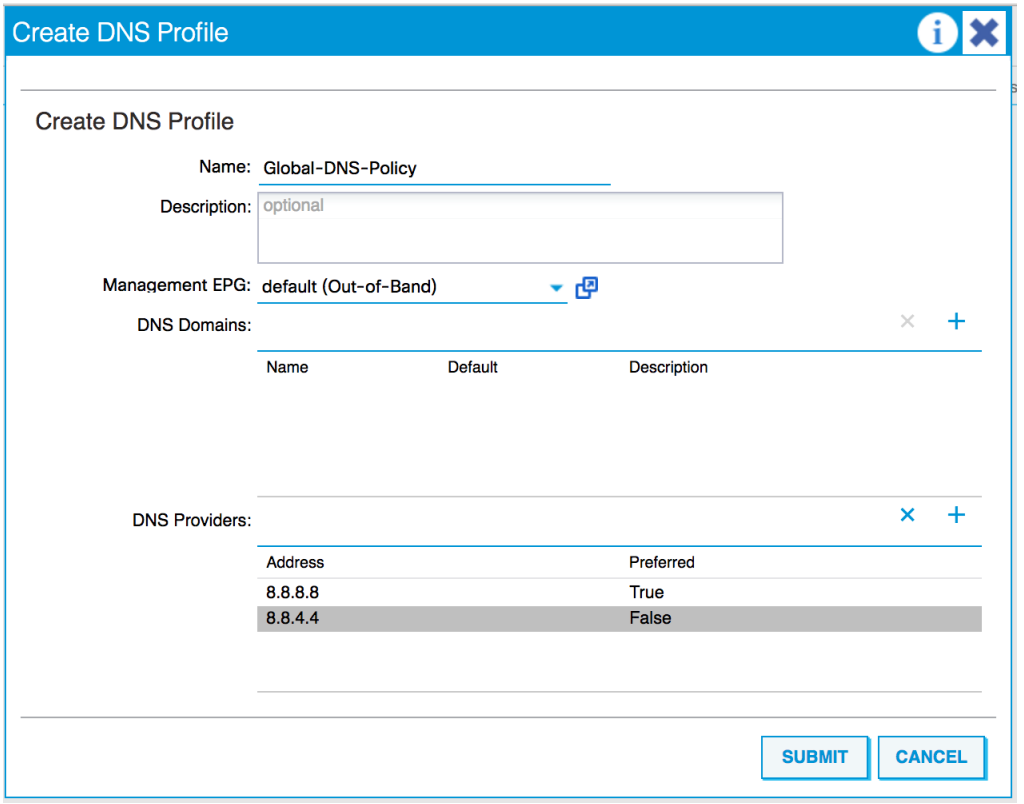
1. From Fabric, select **Fabric Policies** > **Global Policies** > **DNS profiles**. Right-click on DNS Profiles and select **Create DNS Profil**e.

1. 

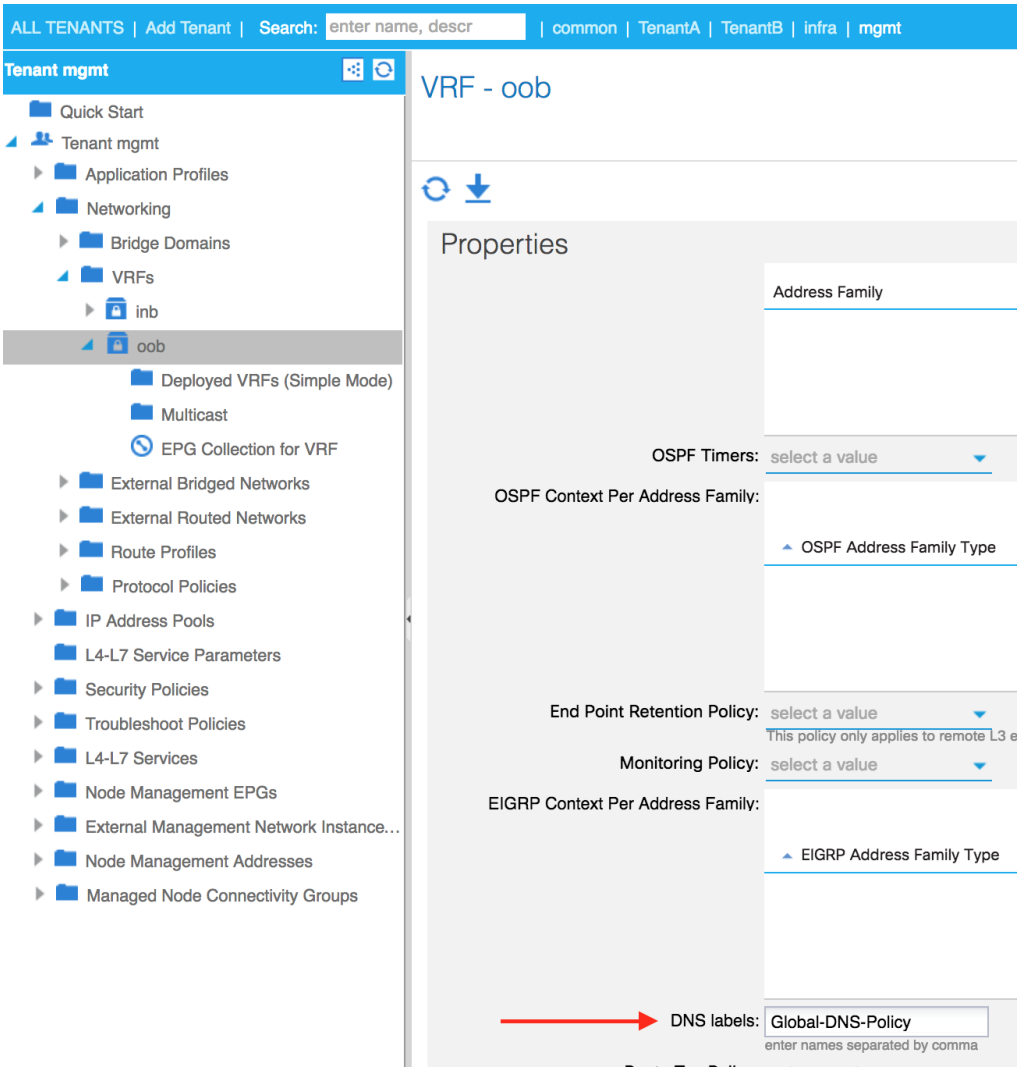
2. Name the profile and select the **default Out-of-band** option next to Management EPG.

1. 

2. Click the plus sign next to DNS Providers and add the IP addresses of the DNS servers. Select the **Preferred** tick box if you want to.

1. 

2. Set the DNS domains in the same manner.
3. Click **Submit**. We need to set the labels, again.
4. From Tenants, select the **mgmt**. tenant.
5. Go to Networking, then to VRFs, and select **oob**.
6. Scroll down the page in the working pane until you see the **DNS labels** box.
7. Enter the DNS label (Global-DNS-Policy).



1. Figure XXX.

2. Click **Submit**.

Perform the steps 6 to 10 with TenantA to provide the tenant with access to the same global DNS profile.

How it works…

Because the APIC uses a Linux-based operating system, the above recipe adds entries to the /etc/resolv.conf file. We can check the contents of this by running the command "**cat /etc/resolv.conf**".

```
apic1# cat /etc/resolv.conf
# Generated by IFC
nameserver 8.8.8.8

nameserver 8.8.4.4

apic1#
```

There's more…

In the recipe above there is the assumption that TenantA will require the same DNS resolution as the APIC management. In an environment where there are multiple tenants, who may have overlapping address spaces (as is often the case with RFC1918 private addresses), then they will need to deploy their own DNS servers to overcome this issue. To facilitate upstream DNS resolution (i.e. names outside of their own network), they would need to employ some sort of external routing within their own VRFs. We will look at external routing next.

Routing with BGP

ACI supports three routing protocols, BGP, OSPF, and EIGRP. We will start by looking at BGP.

As we go forward, we will see that the steps taken to implement OSPF and EIGRP are very similar. The steps, from a 10,000-foot view, are to create an "external routed network," configure an interface, and associate this interface to a bridge domain. To get a visualization of the type of deployment we would be looking at, refer to figure XXX.

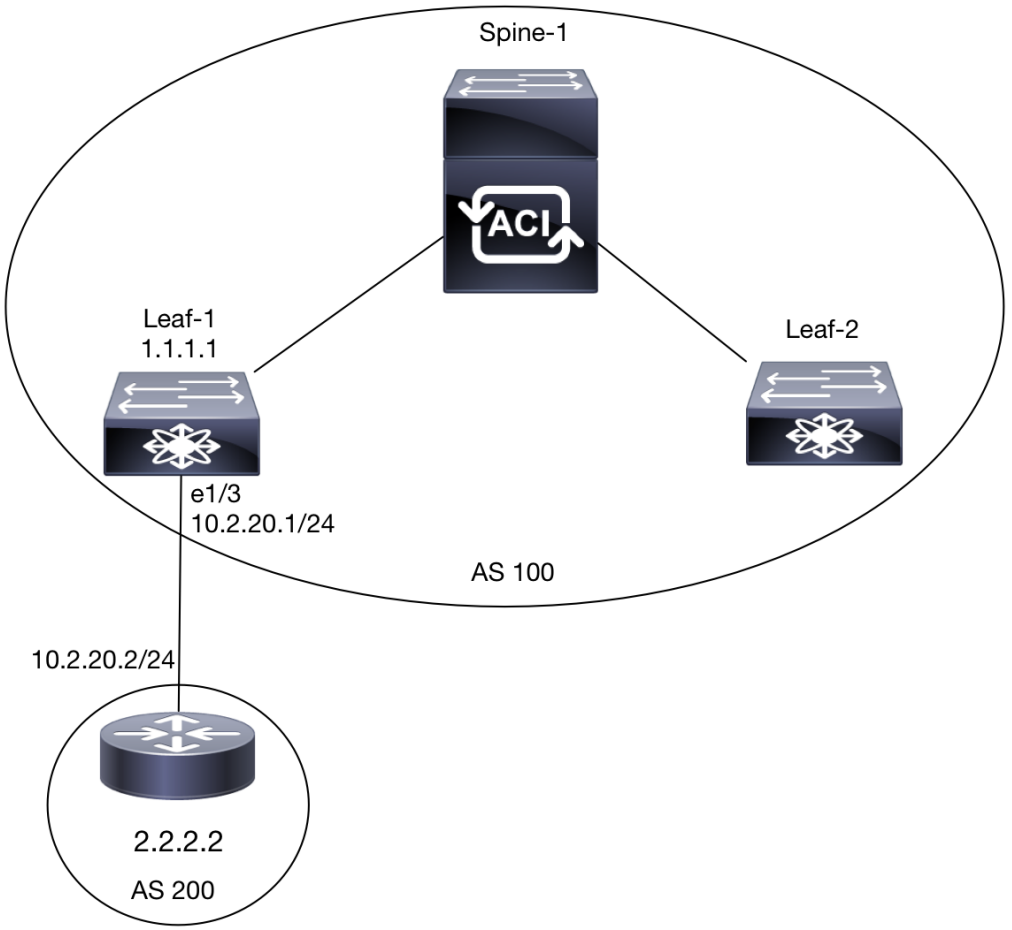
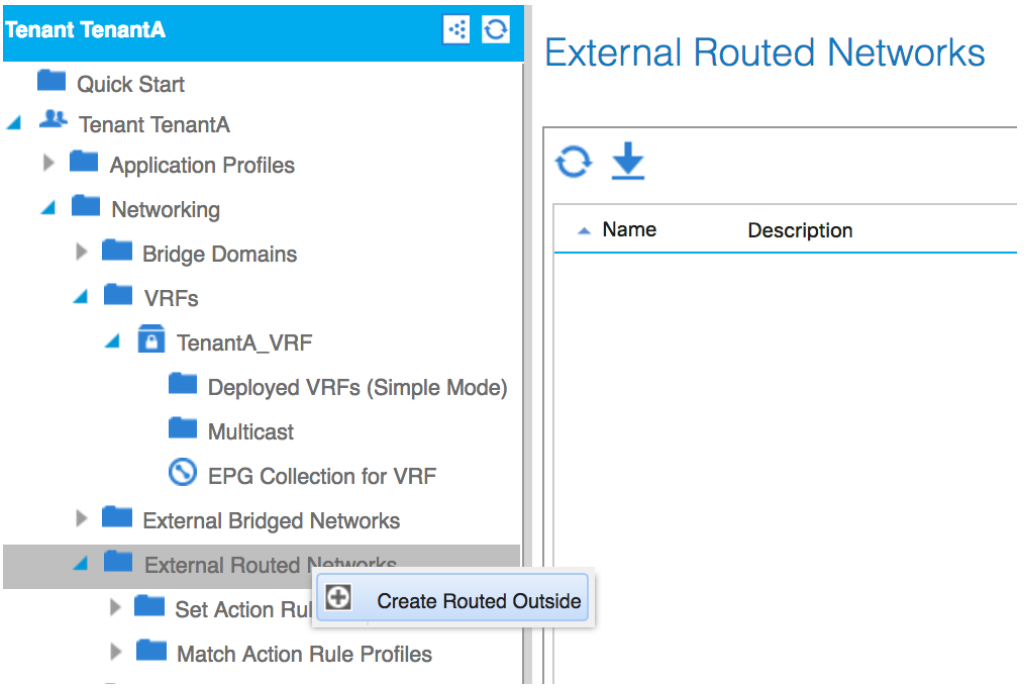


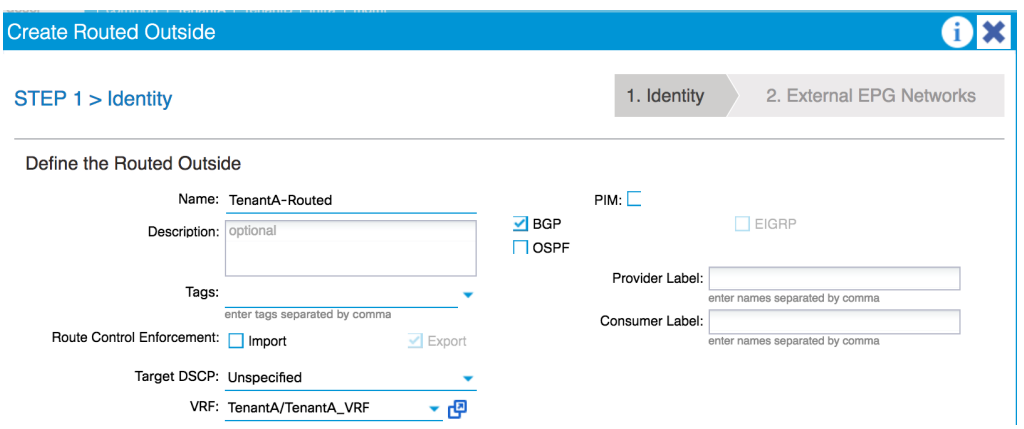
Figure XXX.

How to do it…

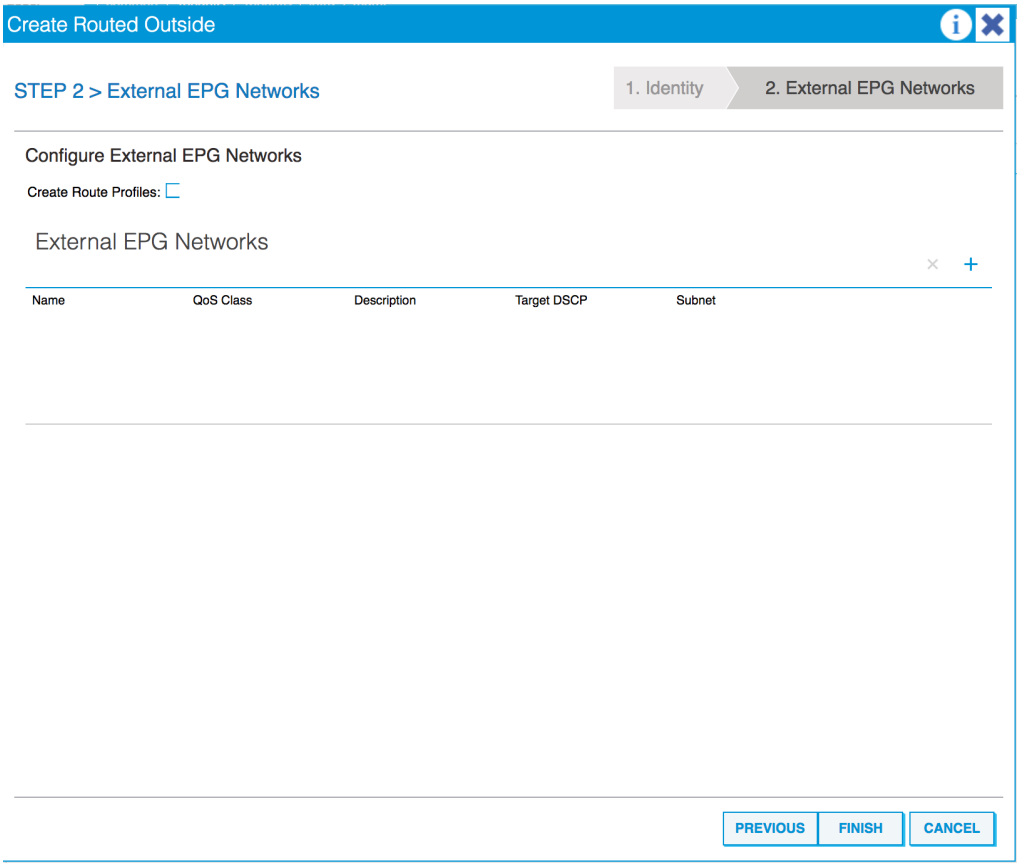
1. Navigate to TenantA > Networking > External Routed Networks.
2. Right-click on this and select “Create Routed Outside.”



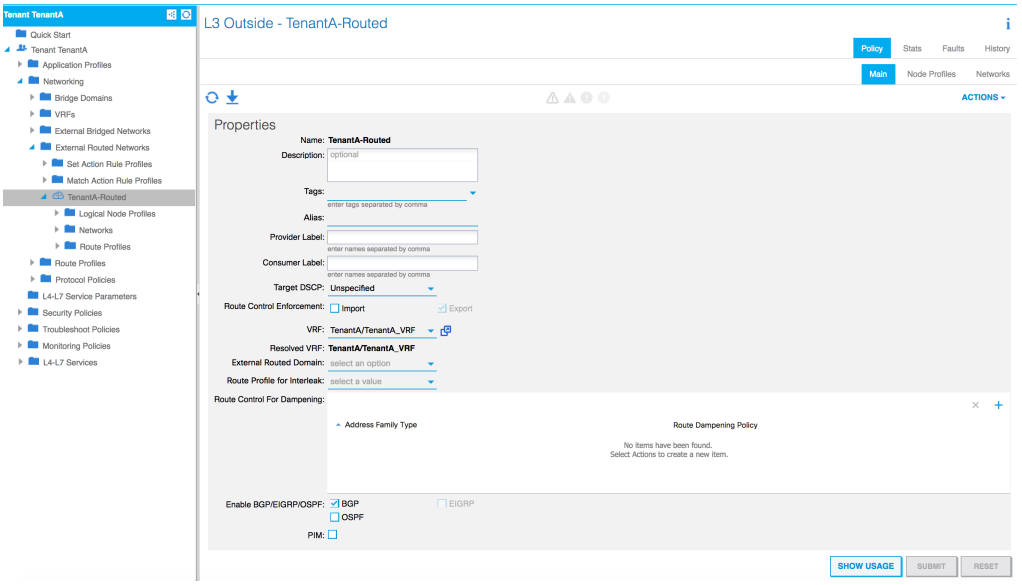
3. Give the new Routed Outside a name.
4. Click BGP checkbox.
5. Select the desired VRF.



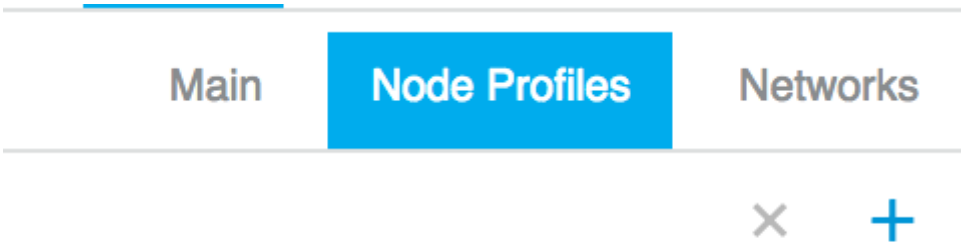
6. Click **Next**.
7. Click **Finish**.



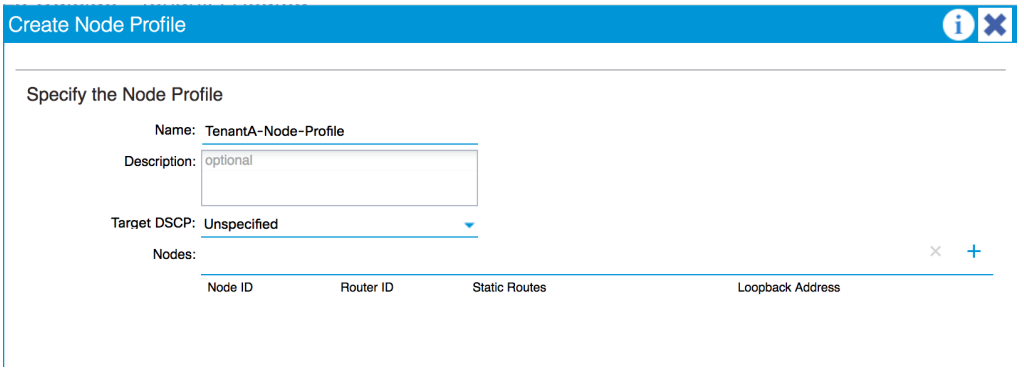
8. The new profile will appear in the work pane. Selecting it will show us the options we have configured so far.



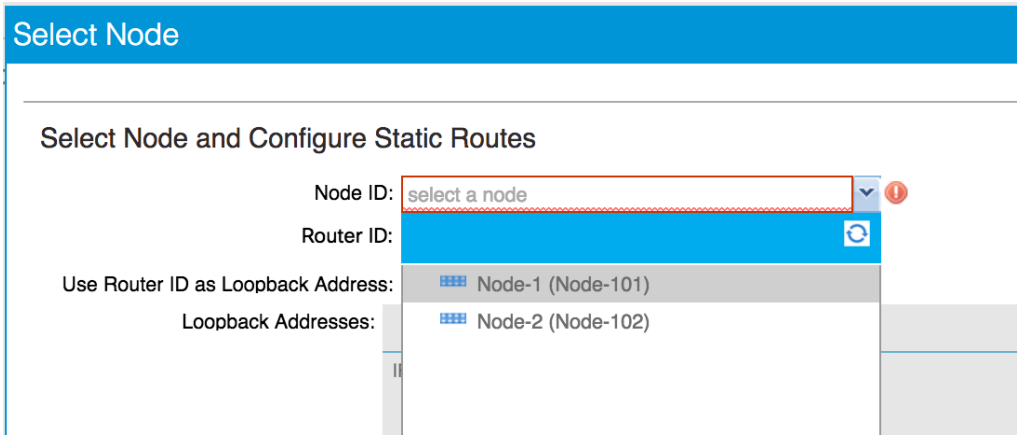
9. Select **Node Profiles** on the right-hand side and click the plus sign.



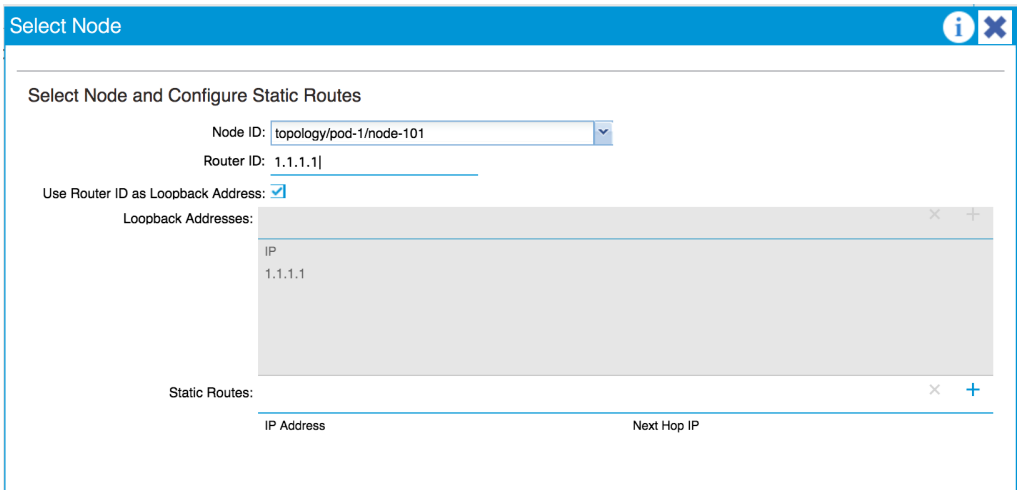
10. Name the profile, and click the plus sign next to Nodes.



11. Select a node from the drop-down menu.

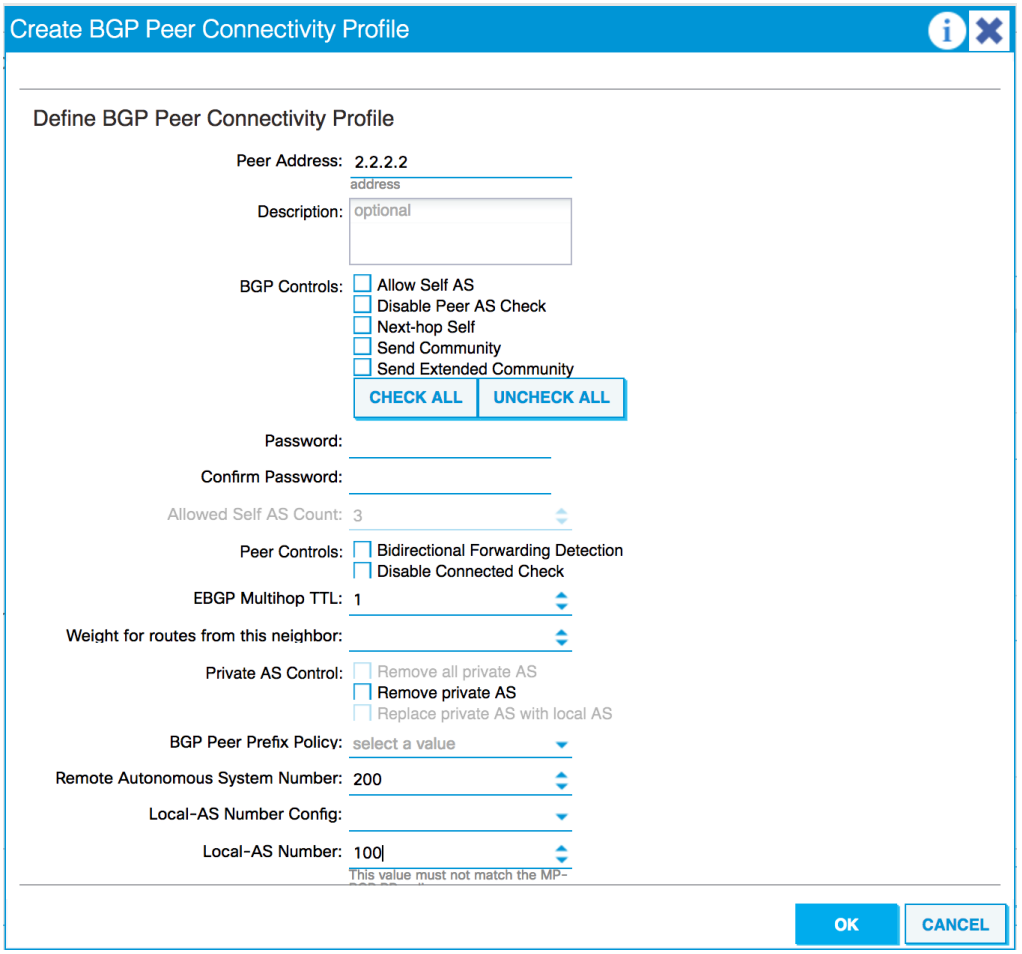


12. Enter a Router ID.



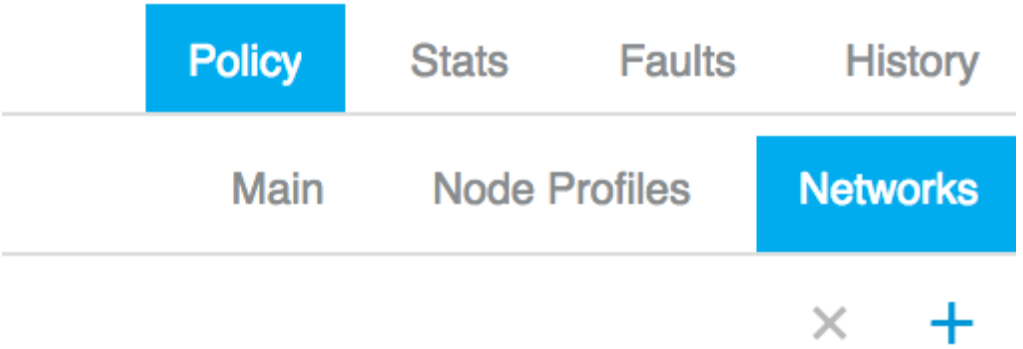
1. Whenever you create a router ID on a leaf switch; it creates an internal loopback interface with that IP address.

2. Click the plus sign next to "BGP Peer Connectivity Profiles."
3. Enter the peer address, the remote AS number and the local AS number.

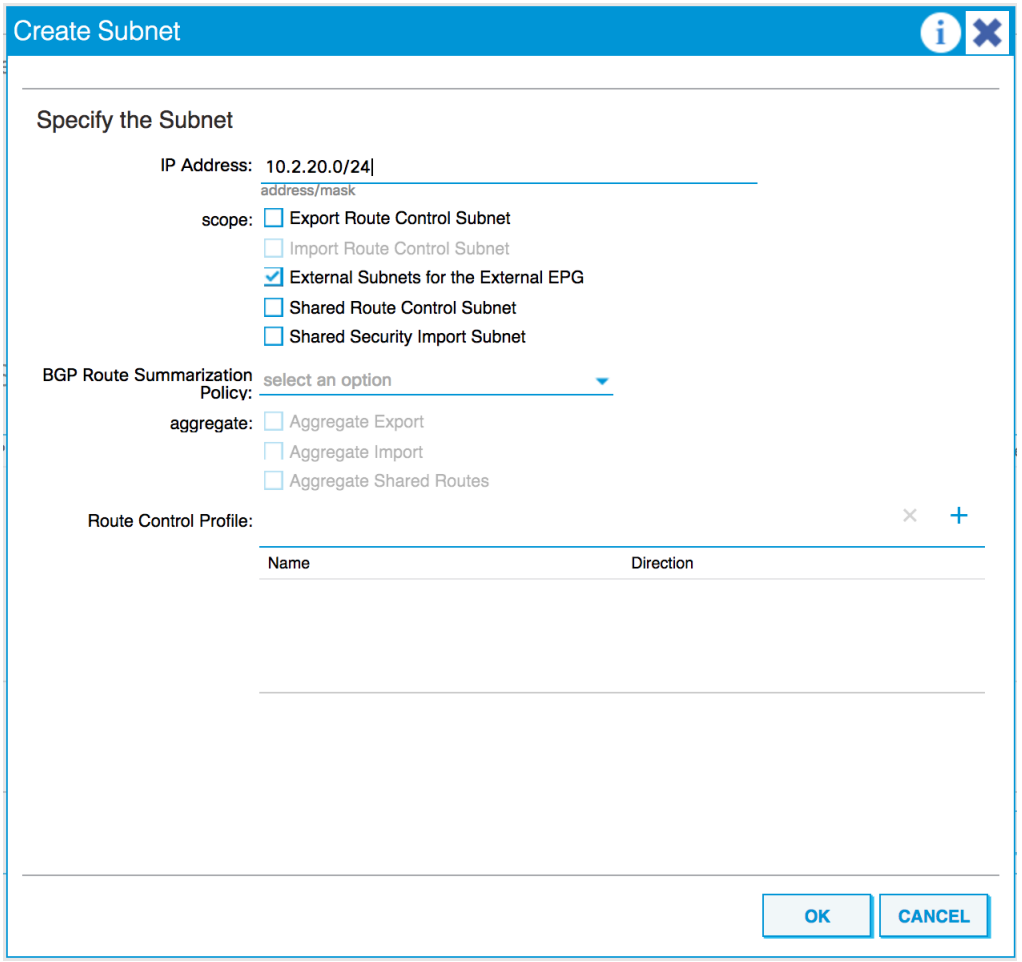


4. Click **OK**.
5. Click **Submit**. We can now create an external network

6. Click "**Networks**" on the top right-hand side, and click the plus sign.



7. Name the network, and click on the plus sign next to **Subnet.**
8. Set the IP address and Subnet.



9. Click **OK**.
10. Click **Submit**.

You can learn more about BGP from my book, **BGP For Cisco Networks**.

We now need to create an interface to connect through.

Configuring a layer 3 outside interface for tenant networks

We can create three types of interfaces for routing; these are:

- Routed Interfaces
- SVIs
- Routed Sub-interfaces

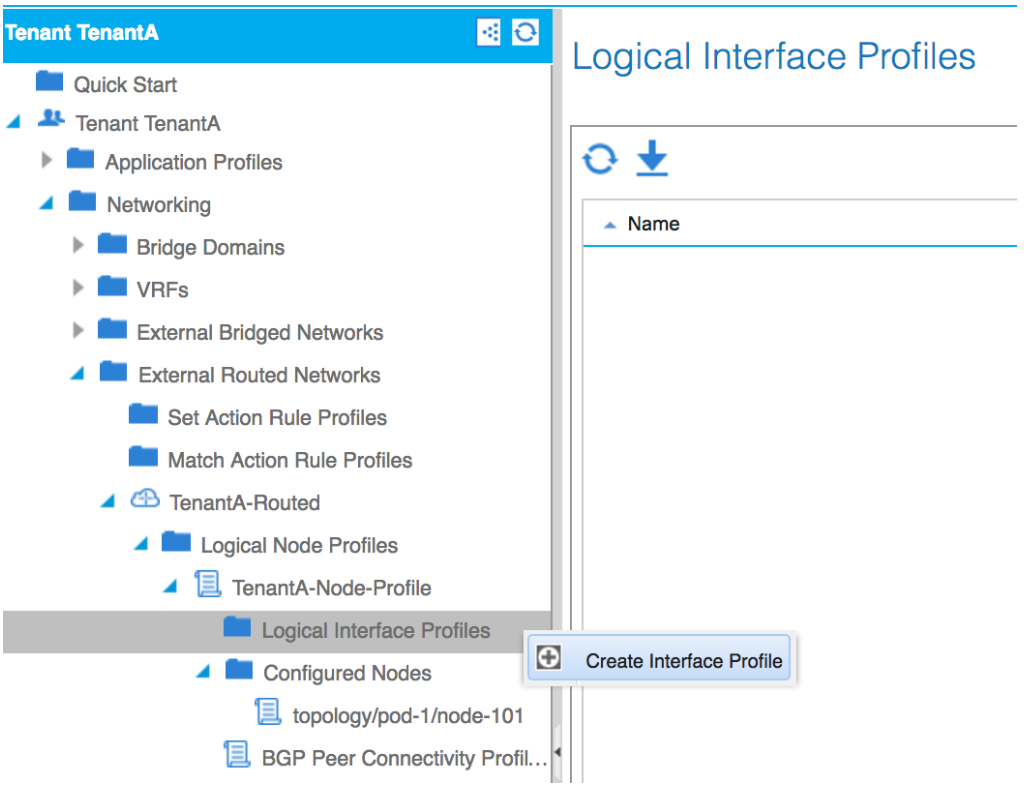
We will create one of each!

How to do it…

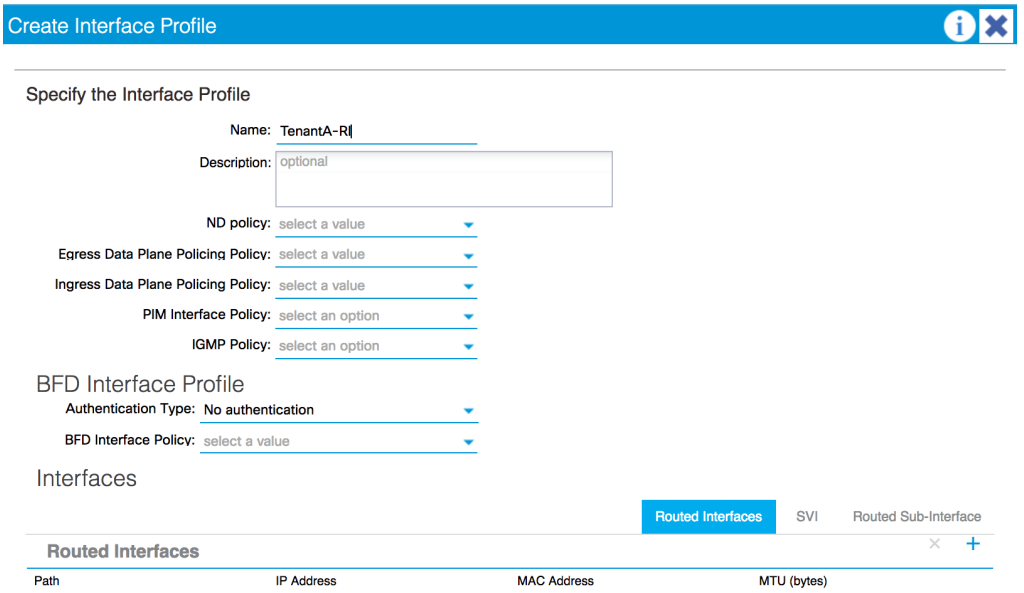
First, we will create Routed interfaces which are physical interfaces that have an IP address.

Creating Routed interfaces

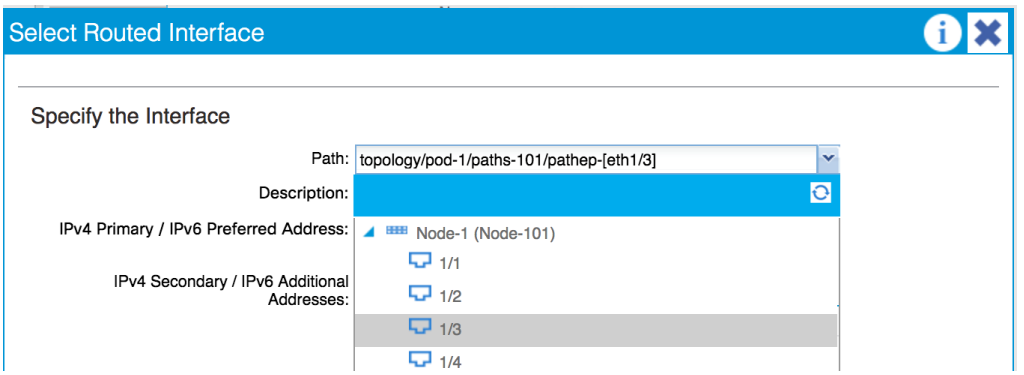
1. Navigate to TenantA > Networking > External Routed Networks > TenantA-Routed.
2. Expand Logical Node Profiles.
3. Expand TenantA-Node-Profile.
4. Right-click “Logical Interface Profiles.”
5. Select “Create Interface Profile.”



6. Name the interface profile and select the interface type (Routed Interface).



7. Click the plus sign to open the "Select Routed Interface."
8. Select the desired leaf node and interface from the drop down. This interface would connect to another device, such as a switch or router with which it would form the BGP peering.

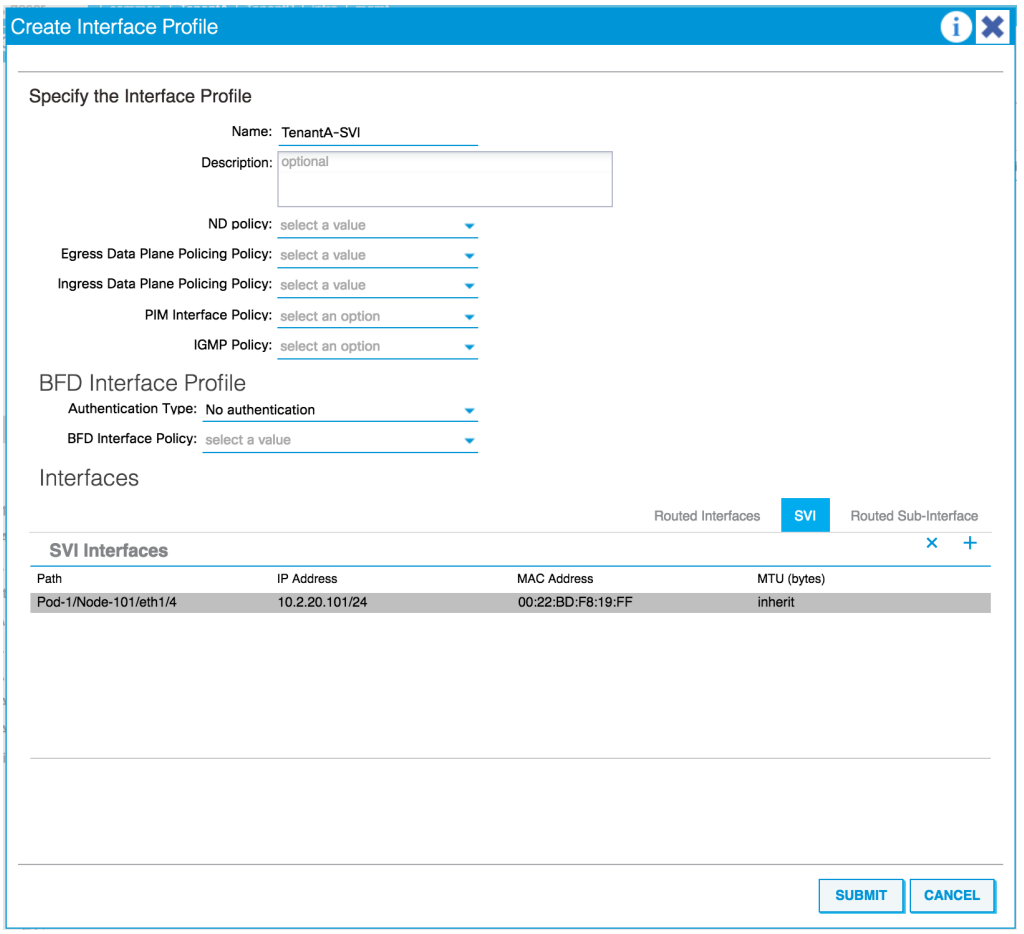


9. Set the IP address (I have chosen 10.2.20.1/24).
10. Set the peer-specific "BGP Peer Connectivity Profiles."
11. Click Submit.

Configuring an External SVI Interface

SVIs, or Switch Virtual Interface, are a virtual interface like the ones we would use in VLAN routing.

1. Follow steps 1-6 from the recipe above, selecting SVI as the interface type.
2. Choose the interface from the drop-down menu.
3. Set the IP address and subnet mask (10.20.20.101/24 here).
4. Click OK.



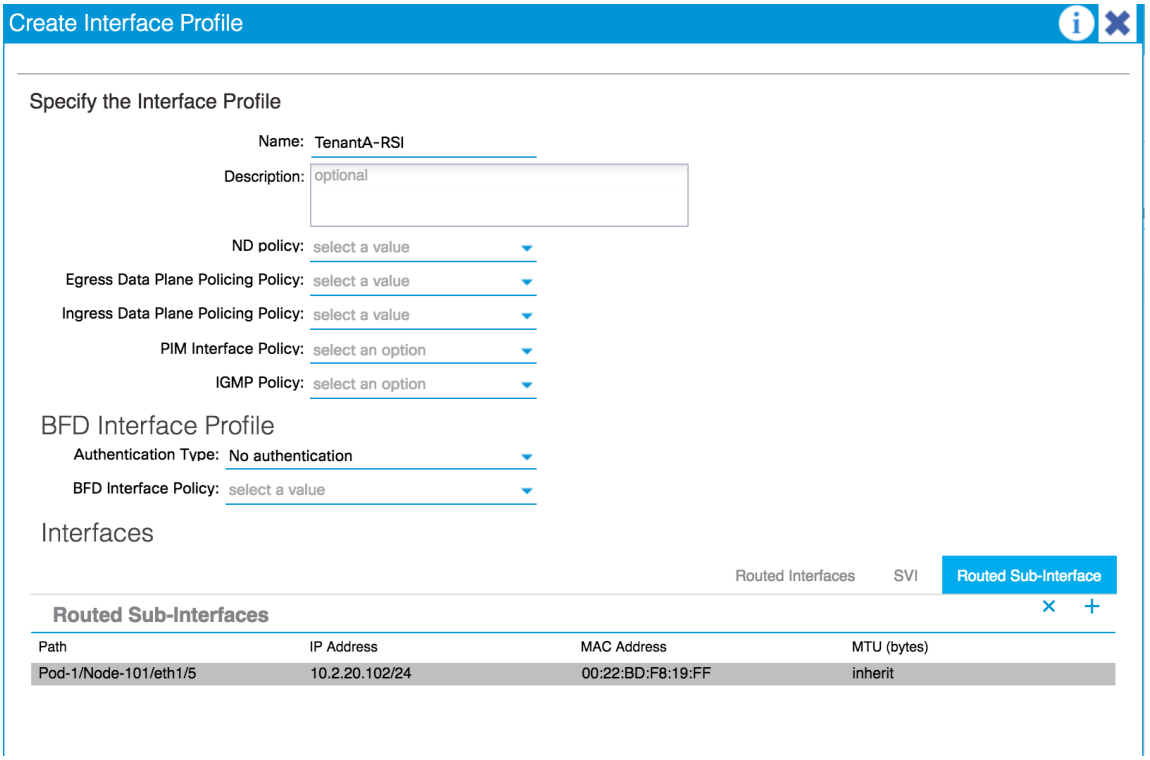
5. Click Submit.

Configuring Routed Sub-Interfaces

A Sub-interface is a virtual interface created “within” a physical interface. It uses dot1q encapsulation.

Follow the steps above for the SVI recipe, but select the Routed Sub-Interface as the interface type.

The result should look like this:

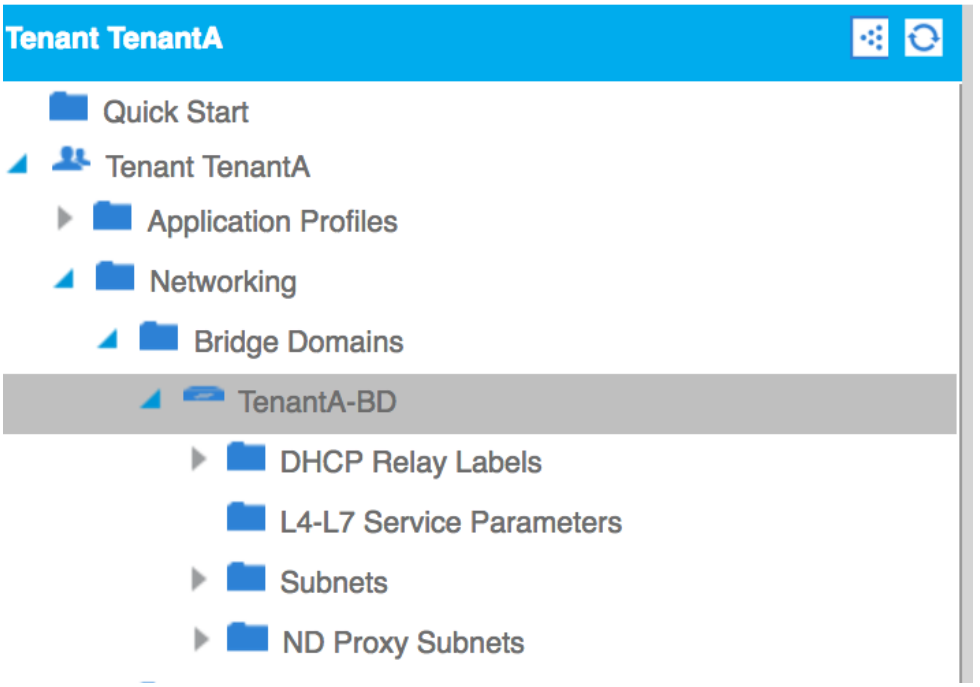


Associating bridge domain with External Network

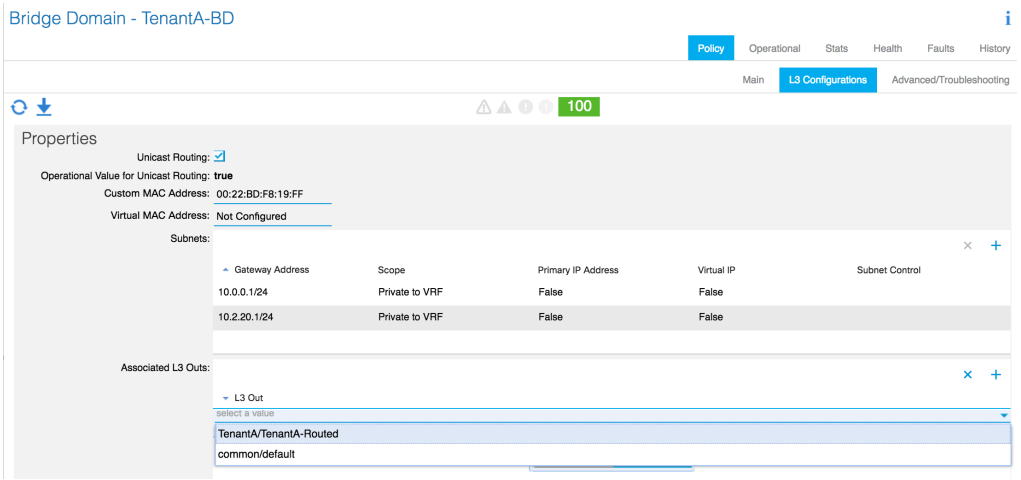
Now that we have a routed interface, we must associate it with the bridge domain.

How to do it…

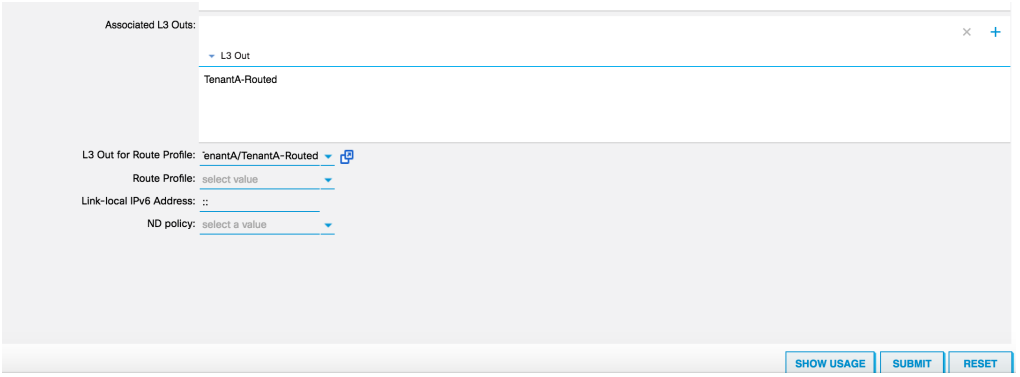
1. Navigate to TenantA > Networking > Bridge Domains > TenantA-BD.



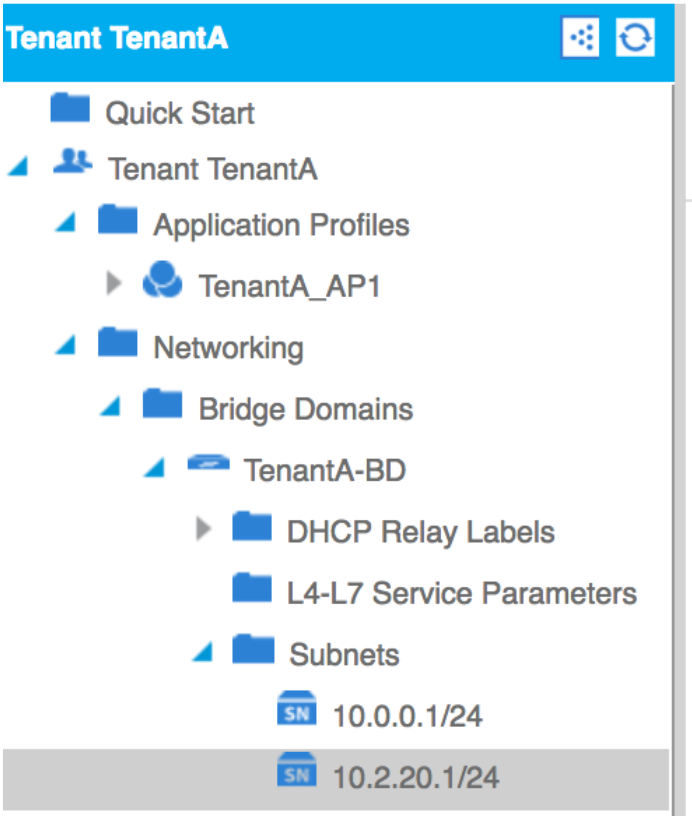
2. Click on L3 Configurations on the right-hand side.
3. If required, you can add additional subnets here as well. In the figure below, I have added the 10.2.20.0/24 subnet.
4. Select the TenantA-Routed L3 Out under “Associated L3 Outs”.



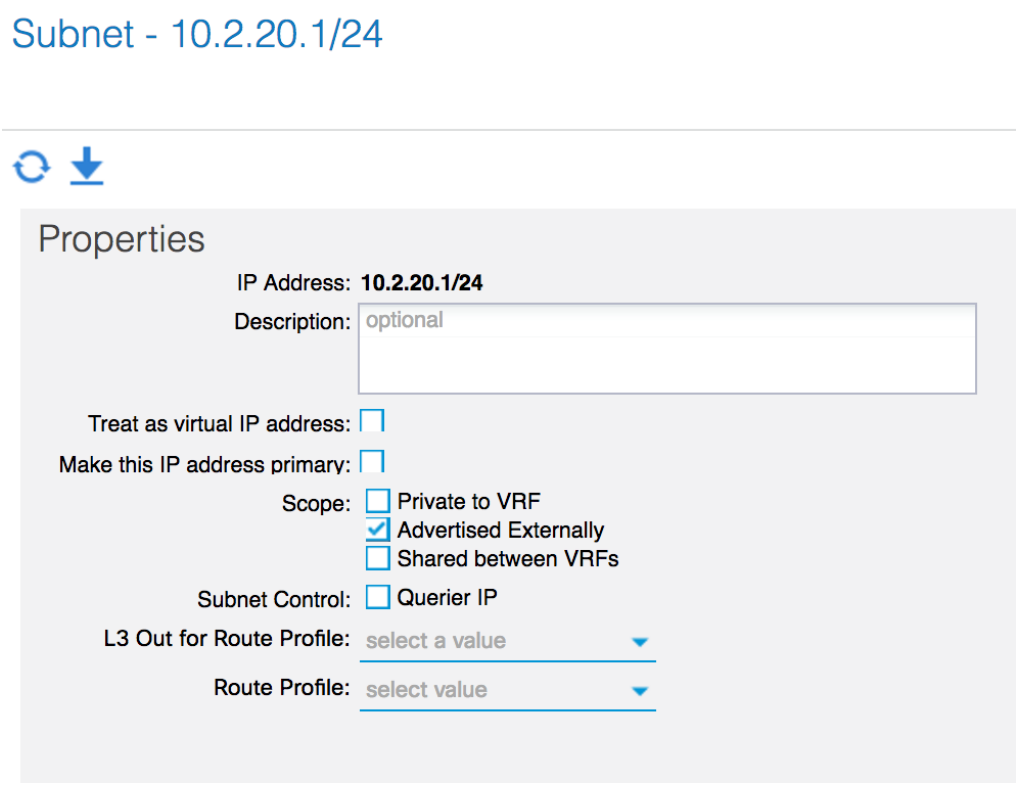
5. Select the same L3 Out for the “L3 Out for Route Profile”.



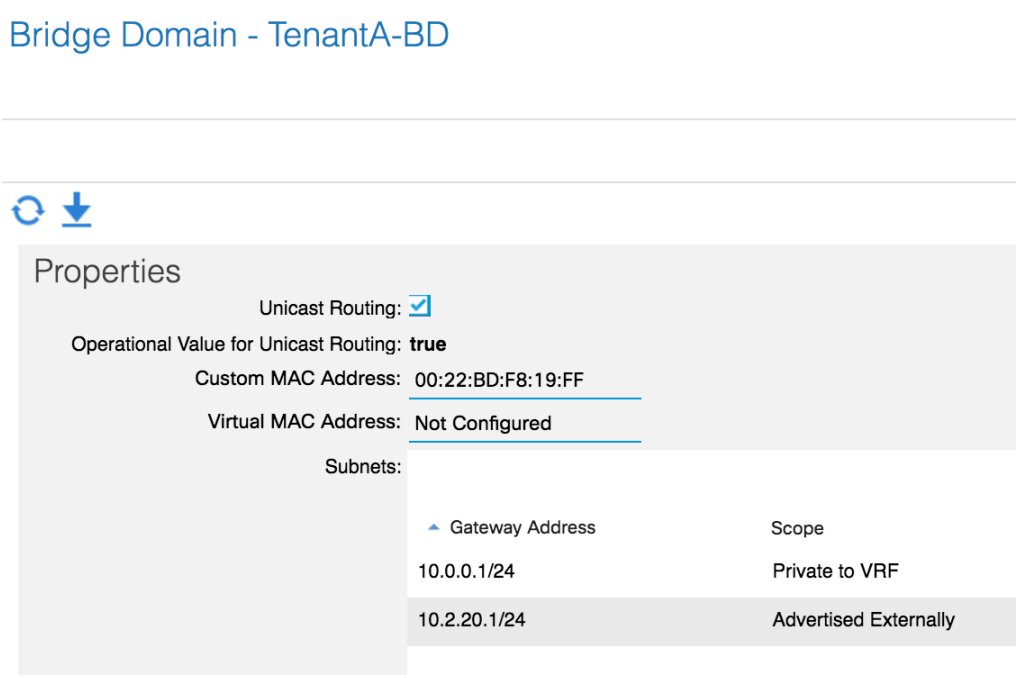
6. Click **Submit**. One issue here is that we will not be advertising any routes. If you have a successful peering to another BGP speaker, you will see that you will receive routes advertised to you, but the other speaker will not receive any prefixes from the tenant. Therefore we must make a small change.
7. Navigate to TenantA > Networking > Bridge Domains > TenantA-BD > Subnets and click on the 10.2.20.1/24 subnet.



8. Tick the box **Advertised Externally**.



9. Click **Submit**.
10. Return to the TenantA-BD bridge domain, and look at the L3 Configurations tab. You will now see that the scope for the 10.2.20.1/24 subnet has now changed to "Advertised Externally."



Using Route Reflectors

We will now enable (on a very limited scale) route reflection in BGP. To do this, we will be reusing the PoD-Policy we created earlier, back in *chapter 2*.

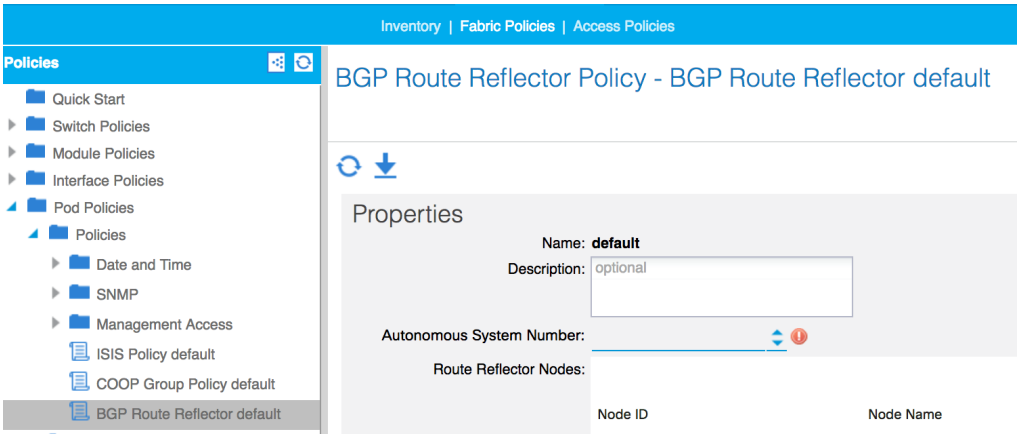
This is the same method you would use if you wanted to implement multiprotocol BGP. Strictly speaking, however, route reflection is not the same as multiprotocol BGP (MP-BGP). Route reflection reflects routes from one route reflector client to another, through a "server," bypassing the need for a full mesh between BGP speakers. MP-BGP is used to carry both IPv4 and IPv6 traffic (as well as MPLS VPN traffic), using "address-families." The two can be implemented together; you can have route reflection within MP-BGP, but to say that configuring BGP route reflectors also implements MP-BGP, is not quite correct. We are looking at this from an ACI-standpoint, though, so let's refer to this as an "alternative fact" and move on to configuring MP-BGP.

Within ACI, MP-BGP serves to distribute the routes received from external sources, within the ACI fabric (from spine to spine). It is not used to connect to external sources.

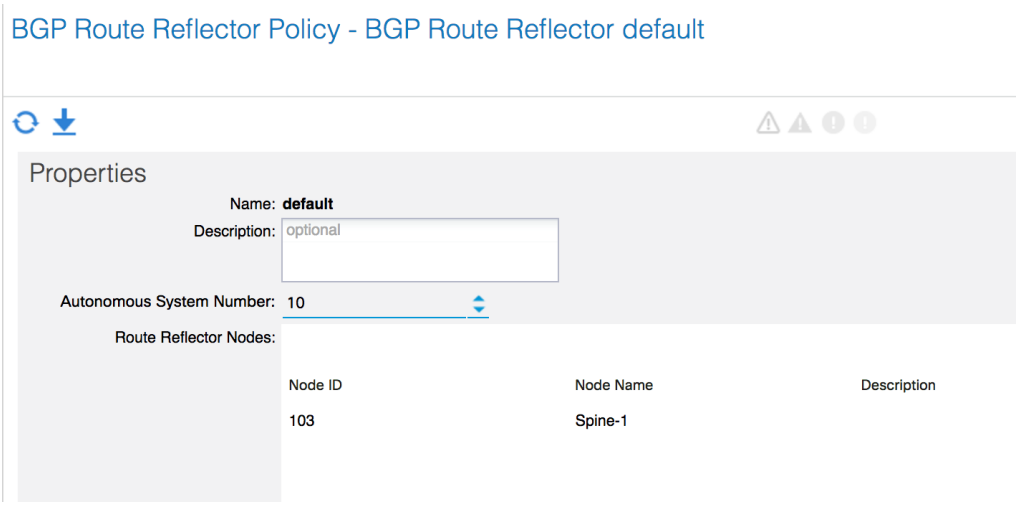
By default, MP-BGP is disabled. We should enable it.

How to do it…

1. Navigate to Fabric > Fabric Policies > Pod Policies > Policies > BGP Route Reflector default.

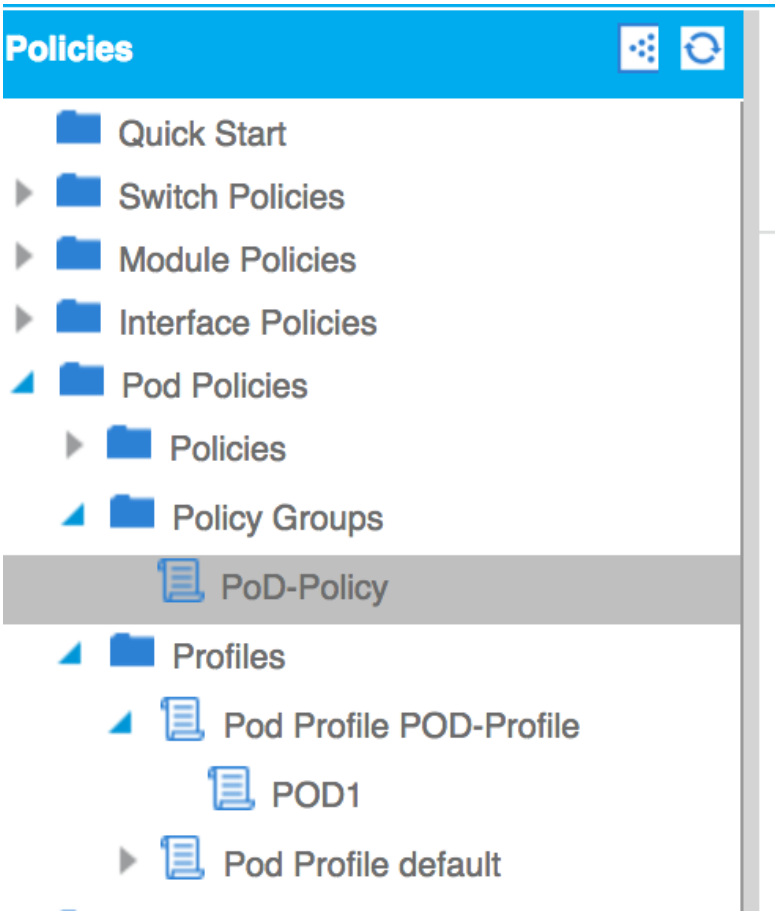


2. Set the AS number and add the spine nodes using their Node ID, which you can find from the **Fabric** > **Inventory page** (Fabric Membership).

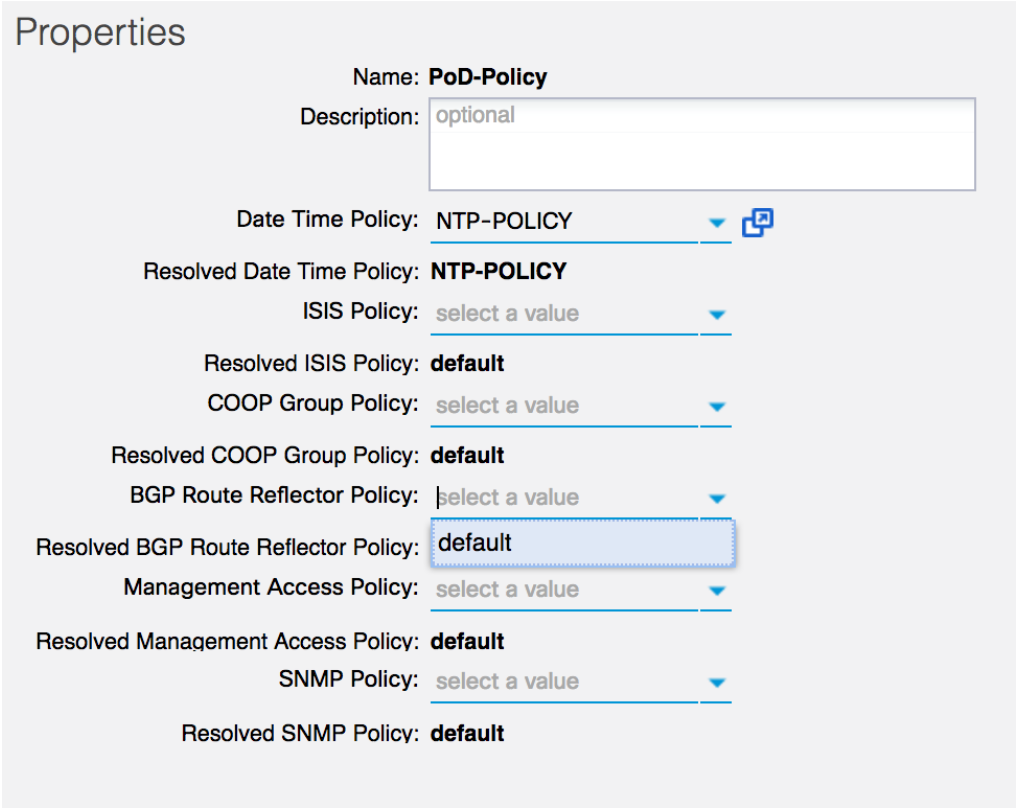


3. Click Submit.
4. Navigate to **Fabric** > **fabric Policies** > **Pod Policies** > **Policy Groups**.

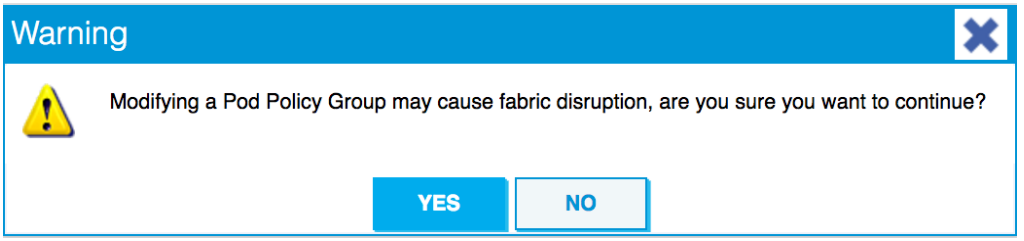
5. Select the PoD-Policy created in the second chapter.



6. Select **default** from the drop down next to **BGP Route Reflector Policy**.



7. Click Submit.
8. You will get a warning (because you are editing an existing, in-use policy group).



9. Depending on whether you are running a production environment, or not, you may want to make this change out of hours. Or just click Yes.

How it works…

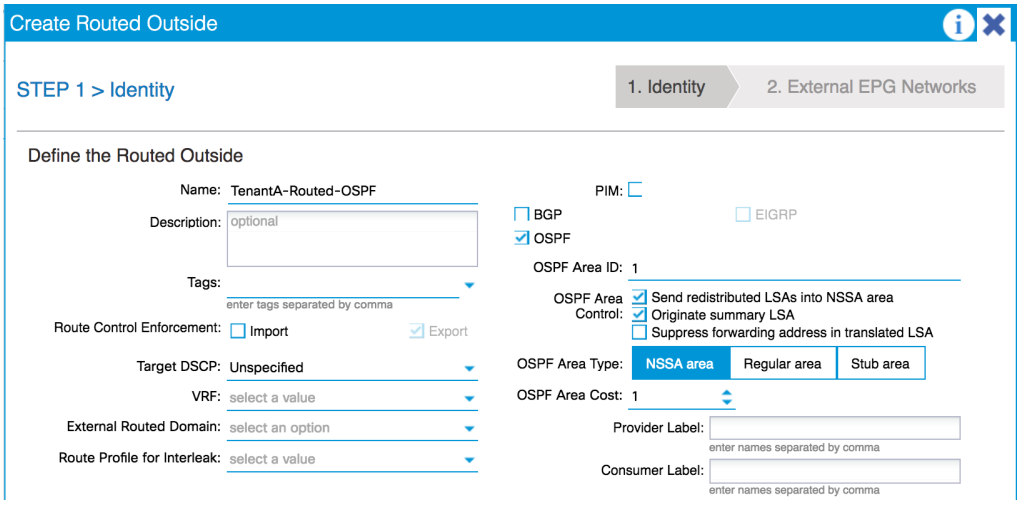
If you have added more than one node (as you should do), then the BGP relationships should form between the spines and the leaf nodes will become route reflector clients of the spines. To check the state, navigate to **Fabric** > **Inventory** > **Fabric** > **Pod 1** > **Spine-1** > **Protocols** > **BGP** > **BGP** for VRF-overlay-1 > Sessions. The state should show "Established."

Routing With OSPF

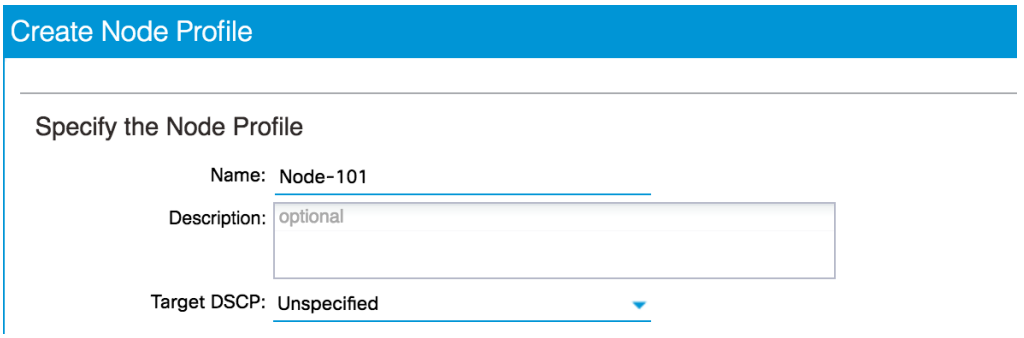
In this recipe, we will cover routing with OSPF, including configuring an OSPF interface Policy and Profile.

How to do it…

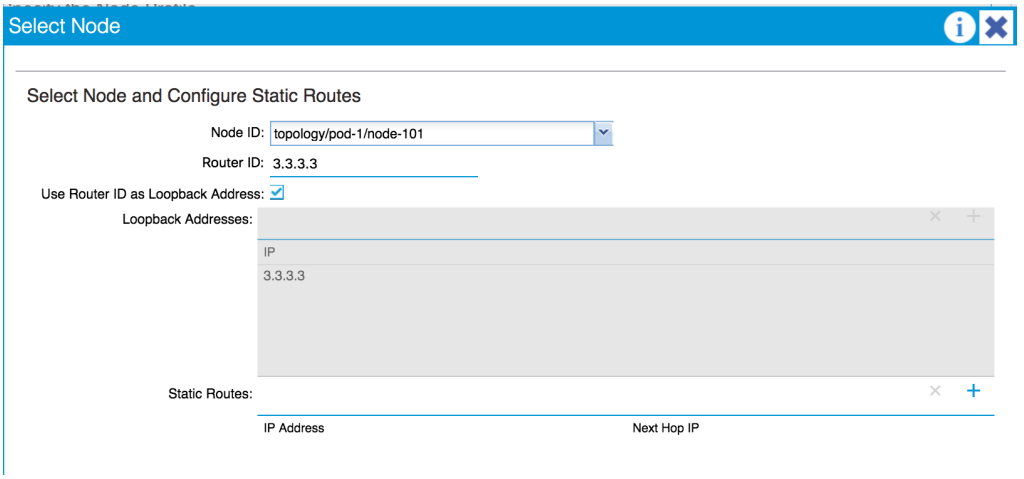
1. Start by creating an External Routed Network, navigate to Tenants > TenantA > Networking > External Routed Networks. Right-click on this and select "Create Routed Outside."
2. Name it, choose OSPF and set the OSPF area ID. Cisco ACI supports NSSA, Regular areas, and Stub areas.



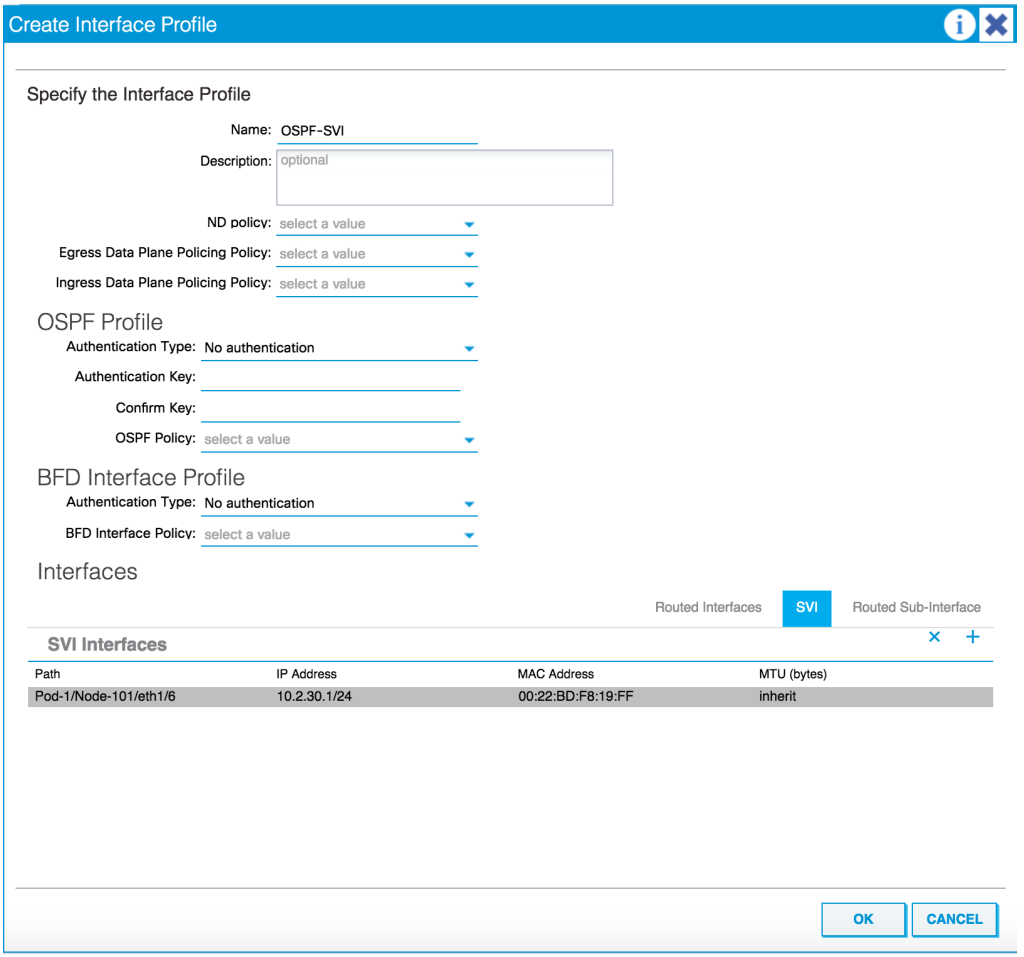
3. Click the plus sign next to "Nodes And Interfaces Protocol Policies."
4. Name the policy.



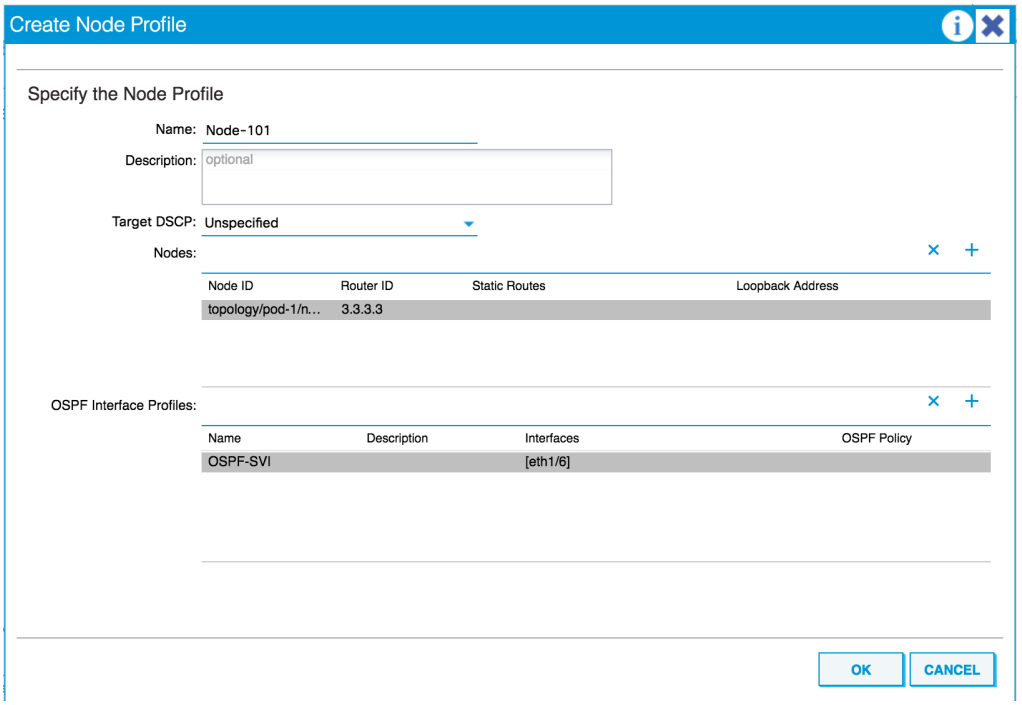
5. Click the plus sign next to Nodes.
6. Select the node and set a router ID.
7. Set any static routes that may be required.



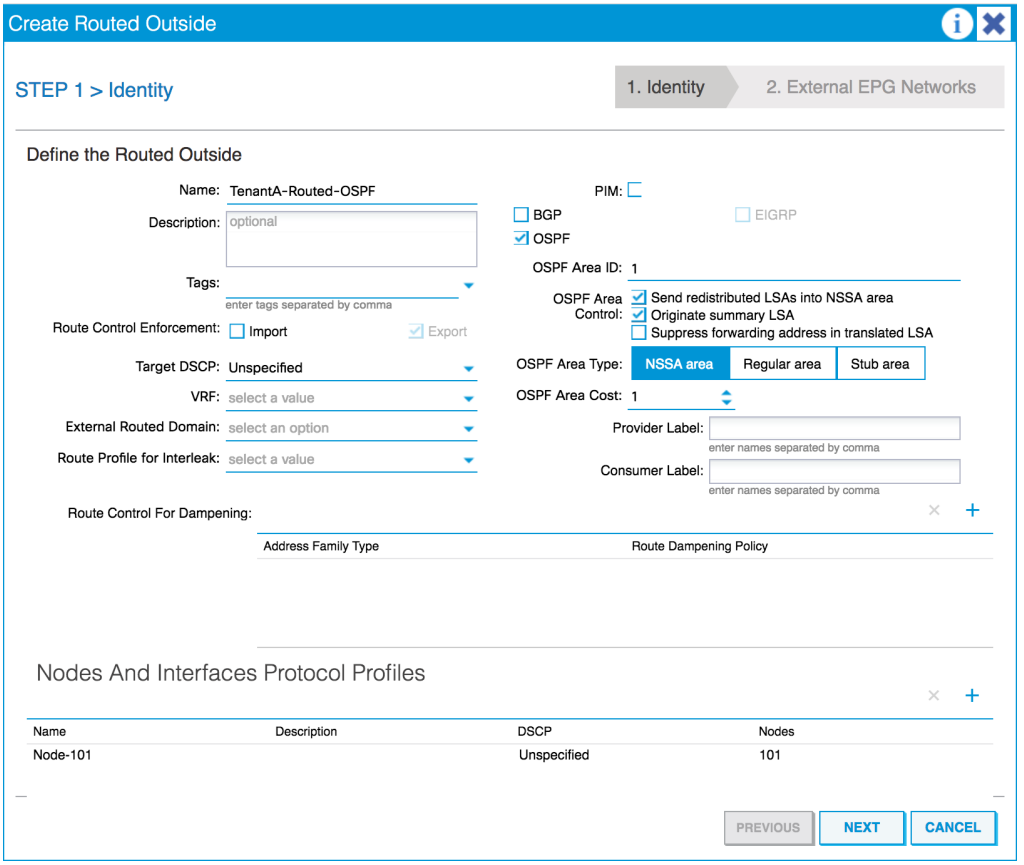
8. Click OK.
9. Click on the plus sign next to "OSPF Interface Profiles."
10. Name the profile and set any authentication settings and BFD (BiForwarding Detection) configuration.
11. Configure the interface type (again choosing from Routed, SVI or Routed Sub-Interface).
12. Click on the plus sign to create the interface.



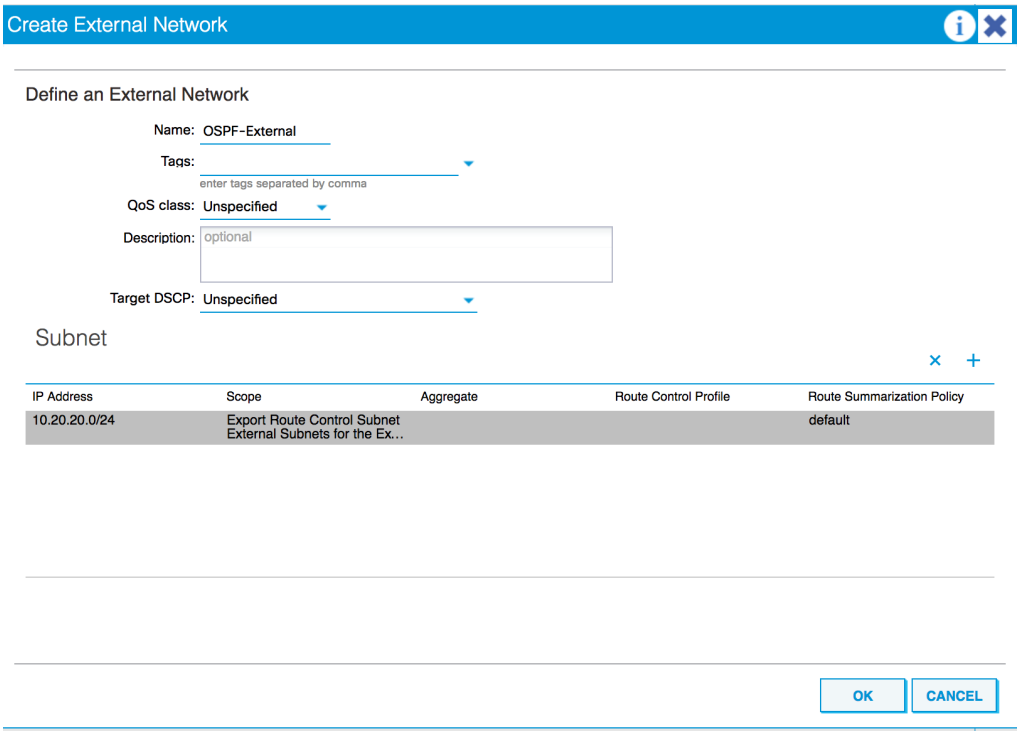
13. Click OK.



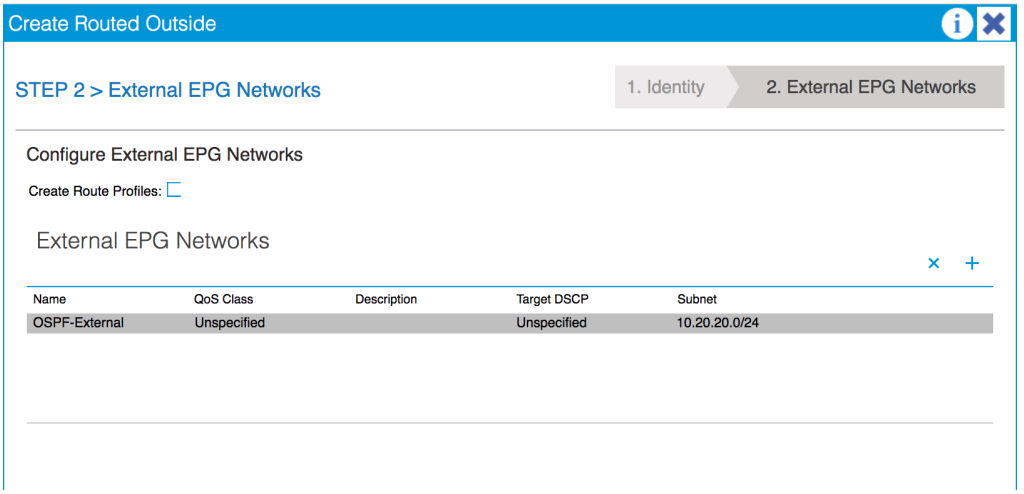
14. Click OK.



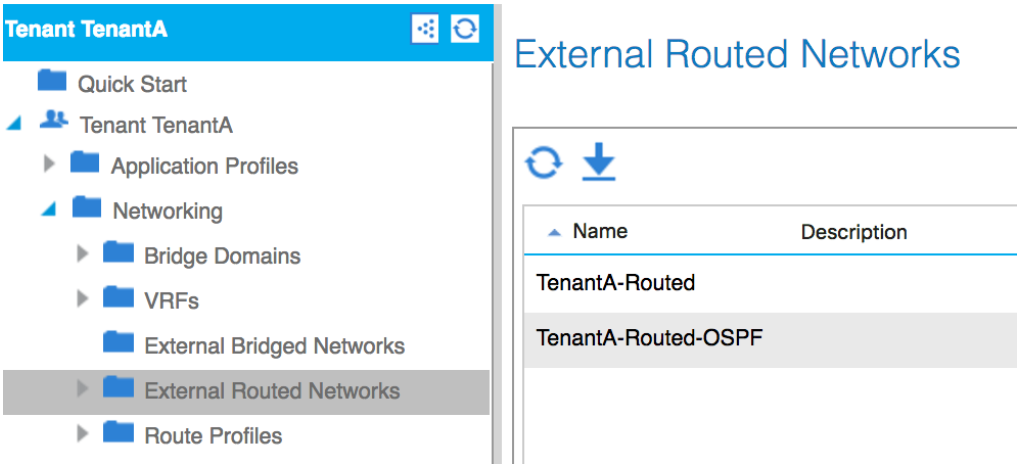
15. Click Next.
16. On the Configure External EPG Networks page, enter a name.
17. Click on the plus sign to create the subnet.
18. Click OK.



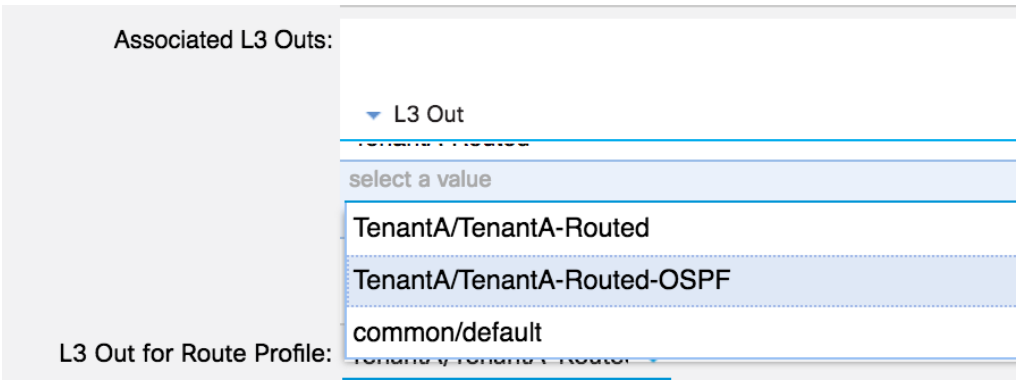
19. Click OK.



20. Click Finish.



21. Return to the TenantA-BD and select the L3 Configurations tab.
22. Click on the plus sign next to Associated L3 Outs.
23. Select the TenantA/TenantA-Routed-OSPF L3 Out.



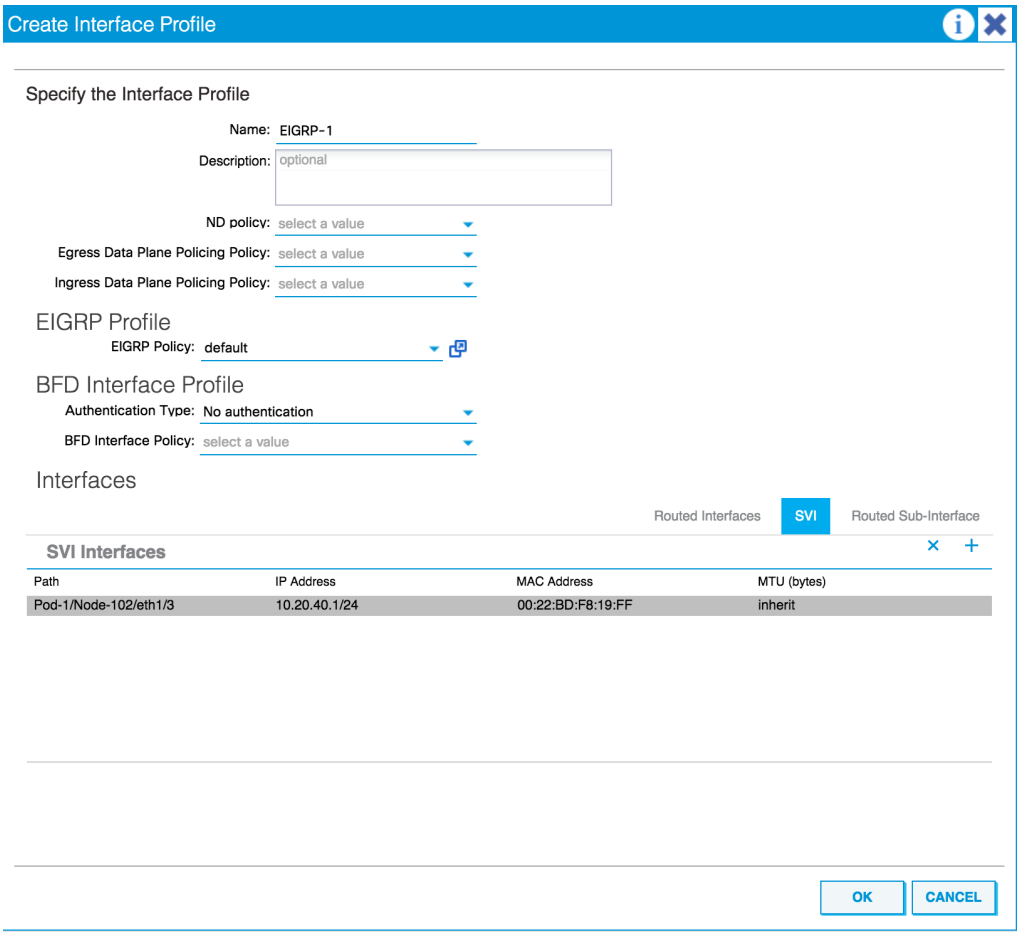
24. Click Update.
25. Change the L3 Out for Routed Profile to be TenantA/TenantA-Routed-OSPF.
26. Click Submit.

Routing with EIGRP

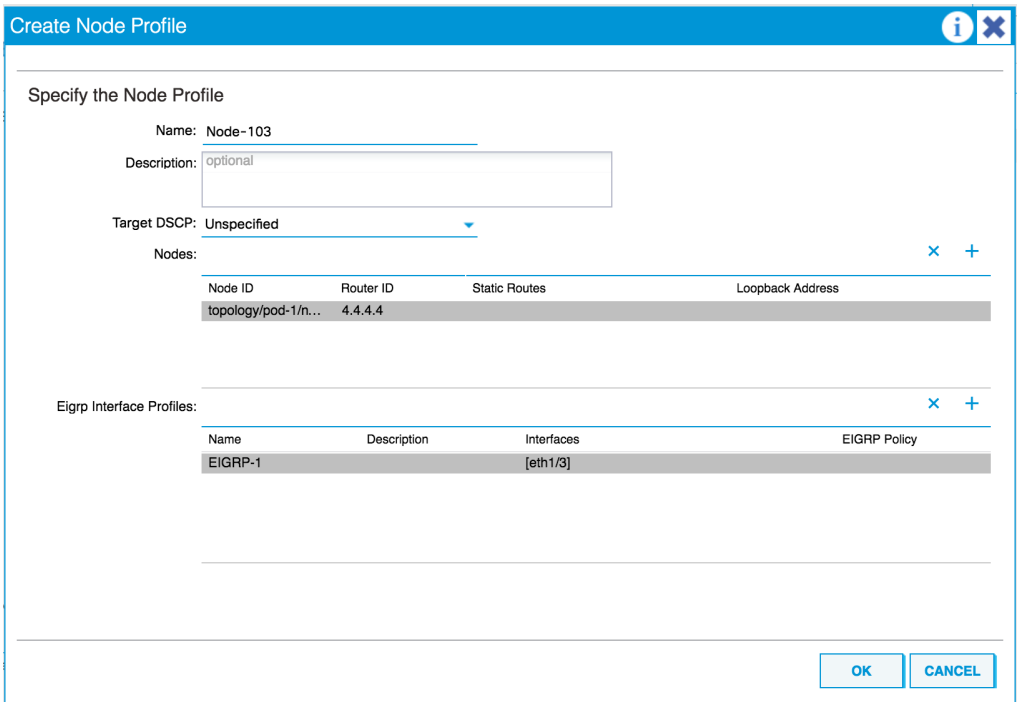
Configuring EIGRP within ACI is no different than configuring BGP or OSPF.

How to do it…

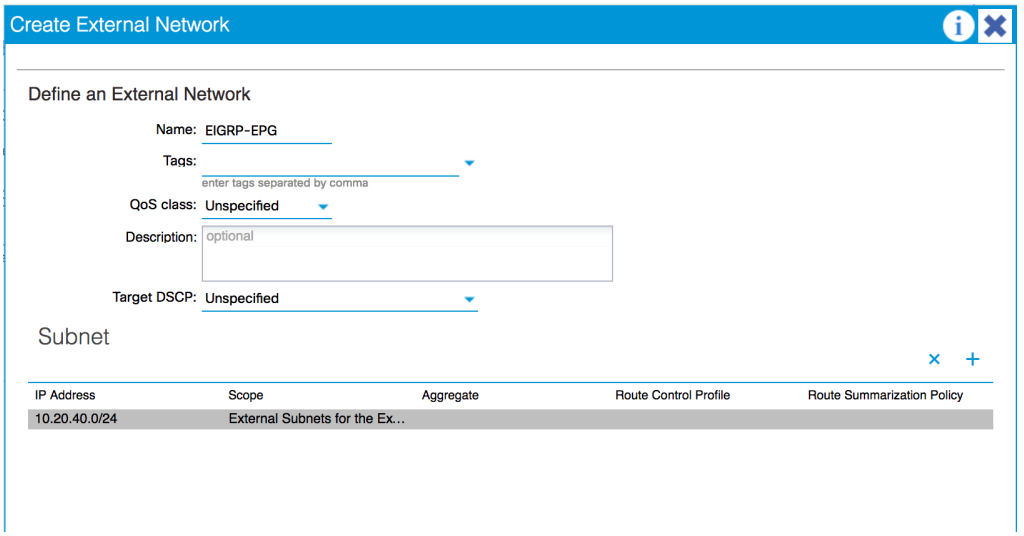
1. Create the External Routed Network (TenantA > Networking > External Routed Networks > Create Routed Outside).
2. Name the new identity.
3. Select the EIGRP checkbox.
4. Set the AS number.
5. Click on the plus sign next to “Nodes And Interfaces Protocol Policies.”
6. Name the Node Profile.
7. Click on the plus sign to add the node.
8. Select the Node ID.
9. Set the router ID.
10. Set any static routes needed.
11. Click OK.
12. Click on the plus sign next to EIGRP Interface Profiles.
13. Name the new profile.
14. Now we need to create an EIGRP Policy or use the default one. This field will show the red circle next to it, showing that it is a required field.
15. Create the interface.
16. Set the IP address.
17. Click OK.



18. Click OK.



19. Click **OK**.
20. Click Next.
21. Configure the External EPG Networks.



22. Click OK.
23. Click Finish.
24. Add it to the bridge domain (as per steps 21-26 from the OSPF recipe above).

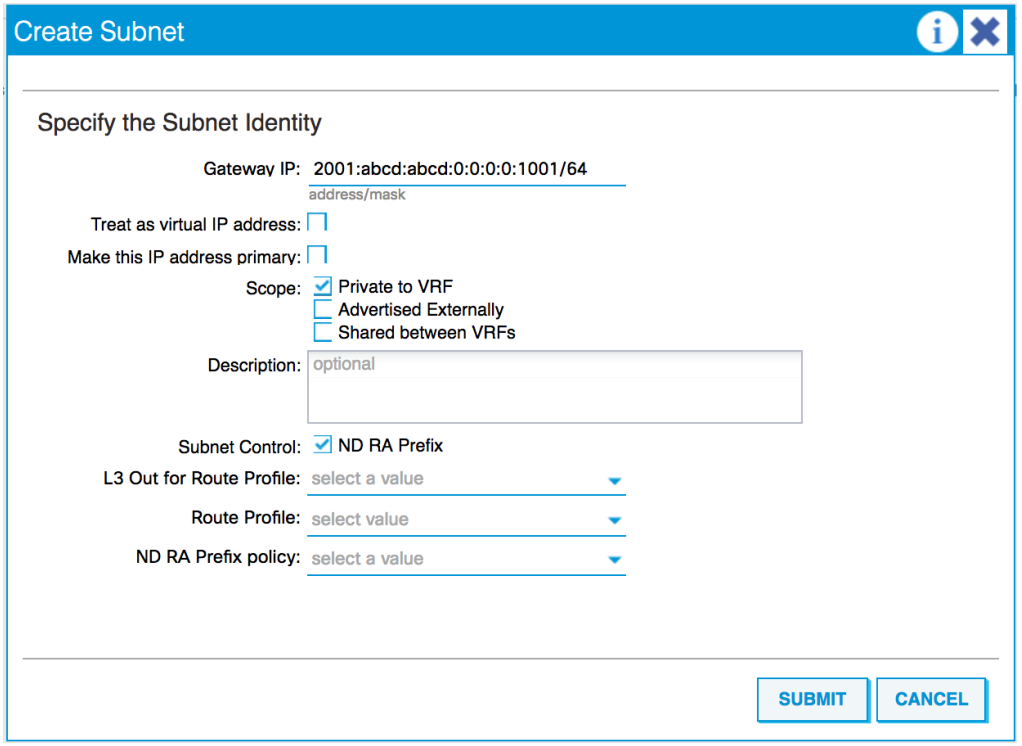
Using IPv6 within ACI

Implementing IPv6 is very simple when compared to traditional IOS routers. It is so simple that Cisco has not even made any distinction between IPv4 addresses and IPv6 addresses in the GUI.

How to do it…

We will add another subnet to TenantA. This time it will be an IPv6 subnet.

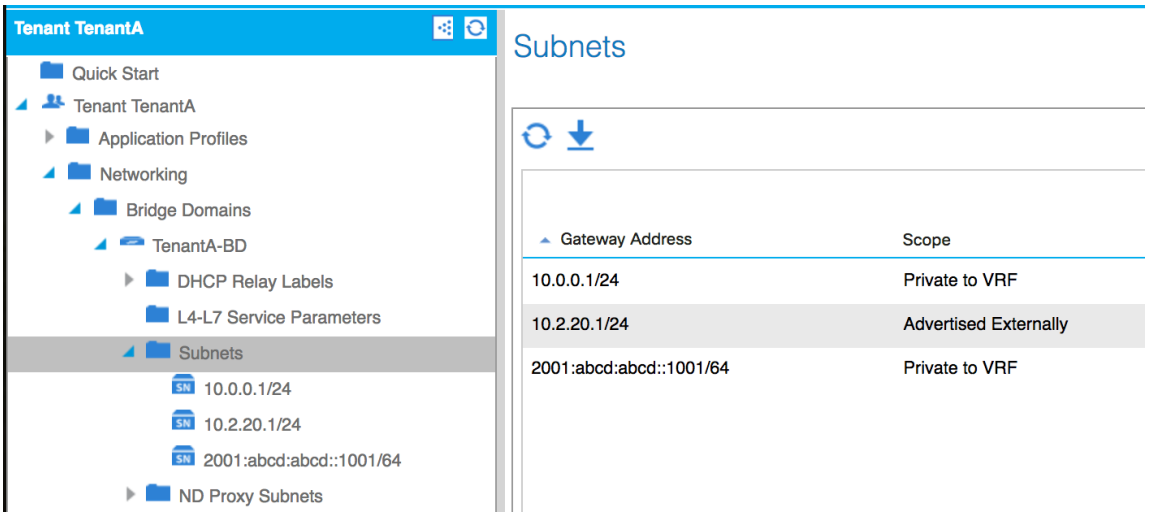
1. Navigate to TenantA > Networking > Bridge Domains > TenantA-BD > Subnets.
2. Click on Actions and select “Create Subnet.”
3. Enter the IPv6 address and subnet mask.



4. Click on **Submit**.

How it works…

The new IPv6 subnet is added in the same way that we added IPv4 subnets.



As you will have noticed from the other recipes in this chapter, routing with IPv6 is treated no differently to IPv4 routing, there is no graphical distinction between the two.

If we switch to the command line, using the NX-OS CLI, we can see that the subnets are all configured in one area (just SSH to the APIC controller):

```
apic1# sh run tenant TenantA
# Command: show running-config tenant TenantA
 tenant TenantA
   vrf context TenantA_VRF
     exit
   bridge-domain TenantA-BD
     vrf member TenantA_VRF
     exit
   application TenantA_AP1
     epg TenantA_EPG1
       bridge-domain member TenantA-BD
       exit
     exit
 interface bridge-domain TenantA-BD
   ip address 10.0.0.1/24 secondary
   ip address 10.2.20.1/24 secondary scope public
   ipv6 address 2001:abcd:abcd::1001/64
   exit
 exit
apic1#
```

Easy, right? Possibly not as easy as multicast, though!

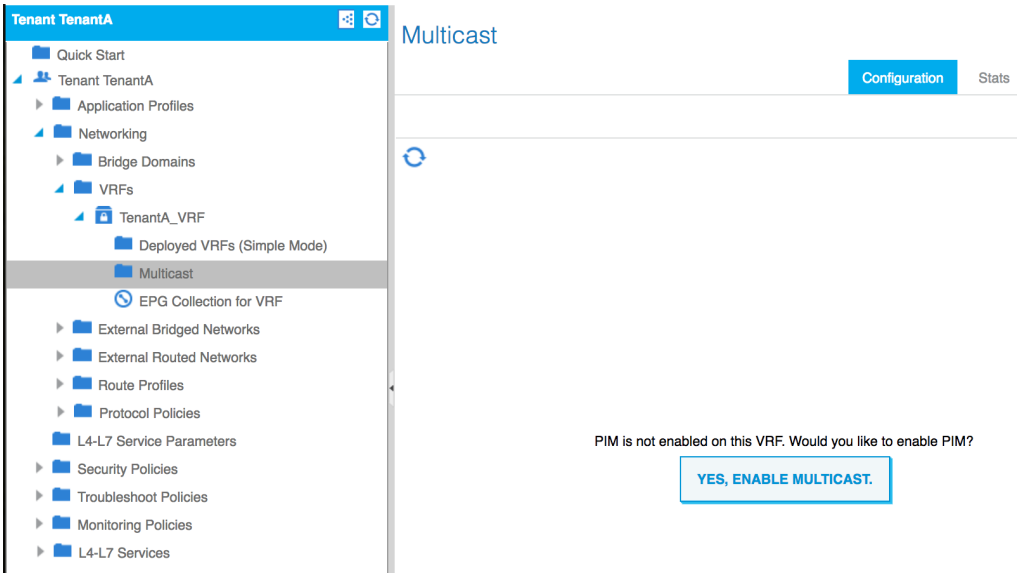
Setting up Multicast for ACI tenants

Let's set up Multicast on the fabric for TenantA.

Ready?

How to do it…

1. Navigate to TenantA > Networking > Bridge Domains > VRFs > TenantA_VRF > Multicast.
2. Click on the button that says “YES, ENABLE MULTICAST”.



How it works…

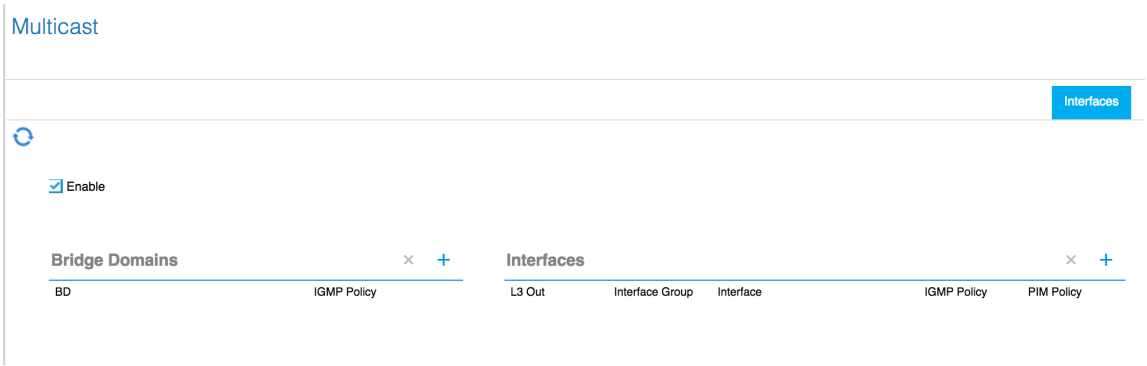
From the NX-OS CLI we can see that **Protocol Independent Multicast**(**PIM**) is enabled for the VRF:

```
apic1# sh run tenant TenantA
# Command: show running-config tenant TenantA
 tenant TenantA
```

```
    vrf context TenantA_VRF
      ip pim
      exit
    bridge-domain TenantA-BD
      vrf member TenantA_VRF
      exit
    application TenantA_AP1
      epg TenantA_EPG1
        bridge-domain member TenantA-BD
        exit
      exit
  interface bridge-domain TenantA-BD
    ip address 10.0.0.1/24 secondary
    ip address 10.2.20.1/24 secondary scope public
    ipv6 address 2001:abcd:abcd::1001/64
    exit
  exit
apic1#
```

So, maybe I was a little over-enthusiastic about the simplicity that ACI offers traditionally complex tasks, but this is not without reason. ACI is very easy to learn.

We have not quite finished with multicast, however. Multicast must be enabled at three levels; the VRF (which we have covered) and also at the bridge domain and L3 out levels, which, if you have clicked the button you will now see:



Configuring Multicast on the bridge domain and interfaces

The second step in configuring multicast is to set it up on the bridge domain and at the

interface level.

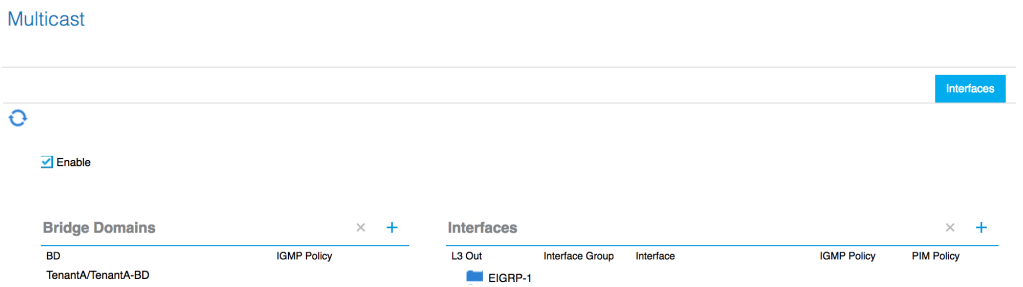
How it works…

We will start by adding the bridge domain:

1. Click on the plus sign next to Bridge Domains.
2. From the drop-down menu, select the TenantA/TenantA-BD bridge domain.
3. Click on Select.

Next, we will add an L3 Out.

4. Click on the plus sign next to Interfaces.
5. Select an L3 Out from the drop down.



How it works…

We now have a multicast-enabled interface with which we would be able to join a multicast group.

There's more…

The configuration (from the CLI) now looks like this:

```
apic1# sh run tenant TenantA
# Command: show running-config tenant TenantA
   tenant TenantA
      vrf context TenantA_VRF
       ip pim
       exit
     l3out EIGRP-1
       vrf member TenantA_VRF
       ip pim
       exit
     bridge-domain TenantA-BD
       vrf member TenantA_VRF
       exit
     application TenantA_AP1
        epg TenantA_EPG1
          bridge-domain member TenantA-BD
         exit
     exit
     external-l3 epg __int_EIGRP-1_topo l3out EIGRP-1
       vrf member TenantA_VRF
       exit
    interface bridge-domain TenantA-BD
      ip address 10.0.0.1/24 secondary
      ip address 10.2.20.1/24 secondary scope public
      ip multicast
      ipv6 address 2001:abcd:abcd::1001/64
    exit
  exit
apic1#
```

As you can see, we now have the VRF, the L3 Out and the bridge domain all enabled for multicast.

That was not too difficult, right?

Let's kick it up a notch and talk about transit routing and route peering.

ACI transit routing and route peering

ACI transit routing allows the ACI fabric to pass routing information from one routing "domain" to another. An example of this would be a server connected to one leaf sending and receiving data from a network segment connected to another leaf. The way this works in a very similar to MPLS, in that the ACI fabric does not appear as a hop within the routes.

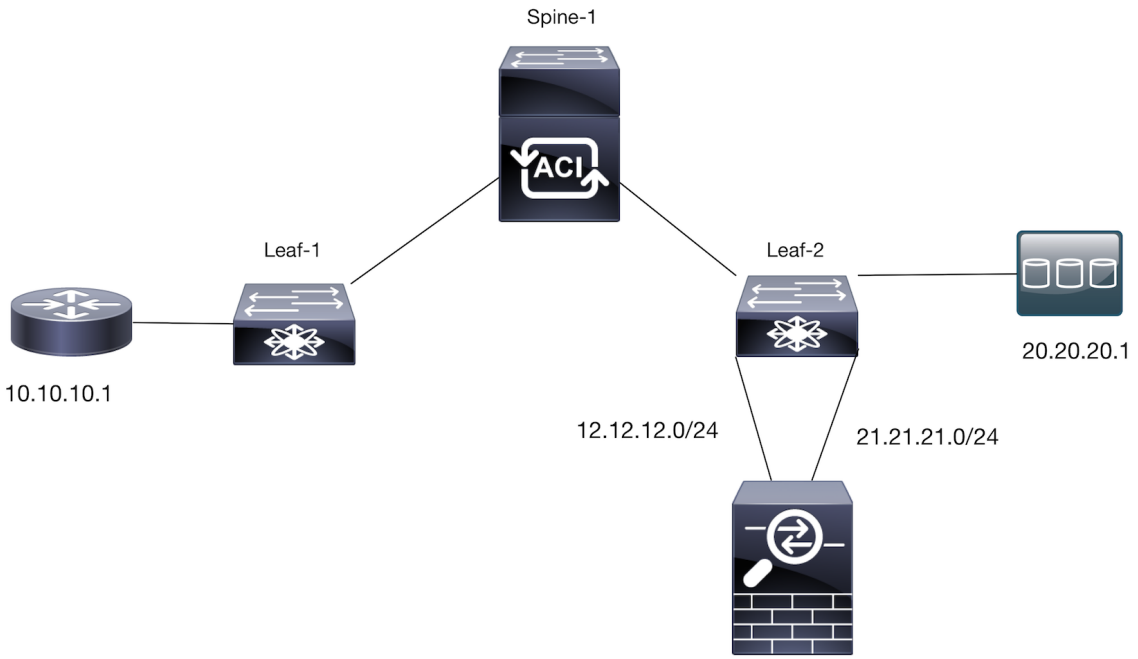
Route Peering is where the ACI fabric is used for BGP or OSPF transit between pods.

Many of the steps in configuring this have already been covered in this chapter and chapters two and three, so instead of reinventing the wheel let's cover some of the theory and less-discussed specifics.

We have a router connected to leaf-1. It is in the subnet 10.10.10.0/24.

We also have a database server connected to another leaf (leaf-2), in the subnet 20.20.20.0/24. The router needs to be able to reach this server by ICMP. The router and the database server are in OSPF area 100, advertising their subnets.

An ASA is connected to leaf-2 by two interfaces.



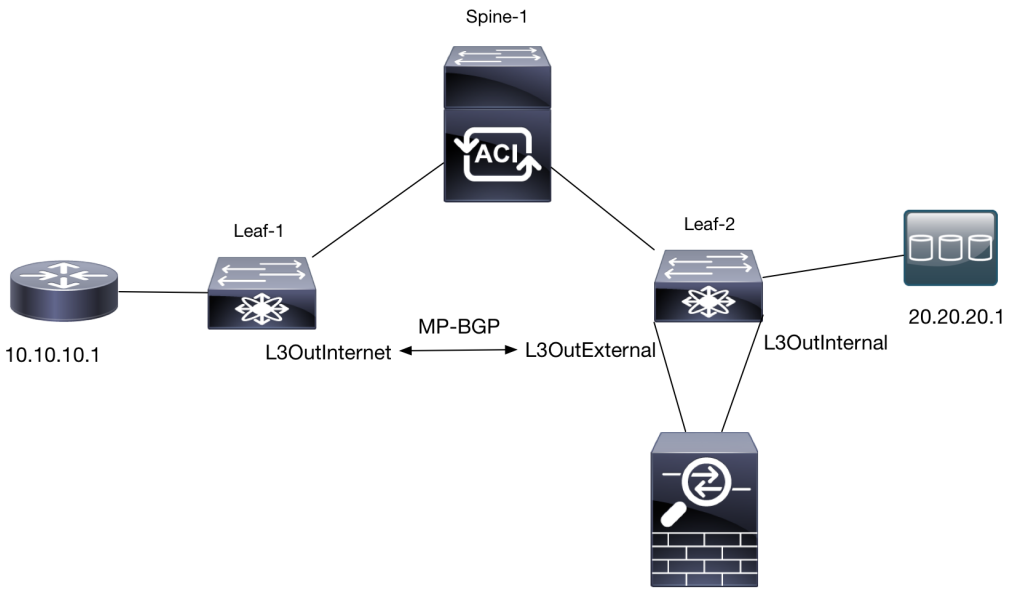
So, how do we get from the router at 10.10.10.1 to the web server at 20.20.20.1?

How to do it…

1. Create a tenant (DB-Tenant).
2. We will need to add the ASAv package, following the recipe in *Chapter 3*.
3. We need to create three L3 Outs.

Name	VRF	Subnets
L3OutInternet	CommonVRF	10.10.10.0/24 (import) 20.20.20.0/24 (import)
L3OutExternal	CommonVRF	12.12.12.0/24 (import) 20.20.20.0/24 (import) 10.10.10.0/24 (export)
L3OutInternal	DB-Tenant	21.21.21.0/24 (import) 10.10.10.0/24 (import) 20.20.20.0/24 (export)

4. The ASA's 12.12.12.0/24 and 21.21.21.0/24 networks will act as the “transit”.
5. Route redistribution needs to be enabled, so that, through MP-BGP, the routes between the router and the database server are exchanged between the L3OutInternet and L3OutExternal interfaces.



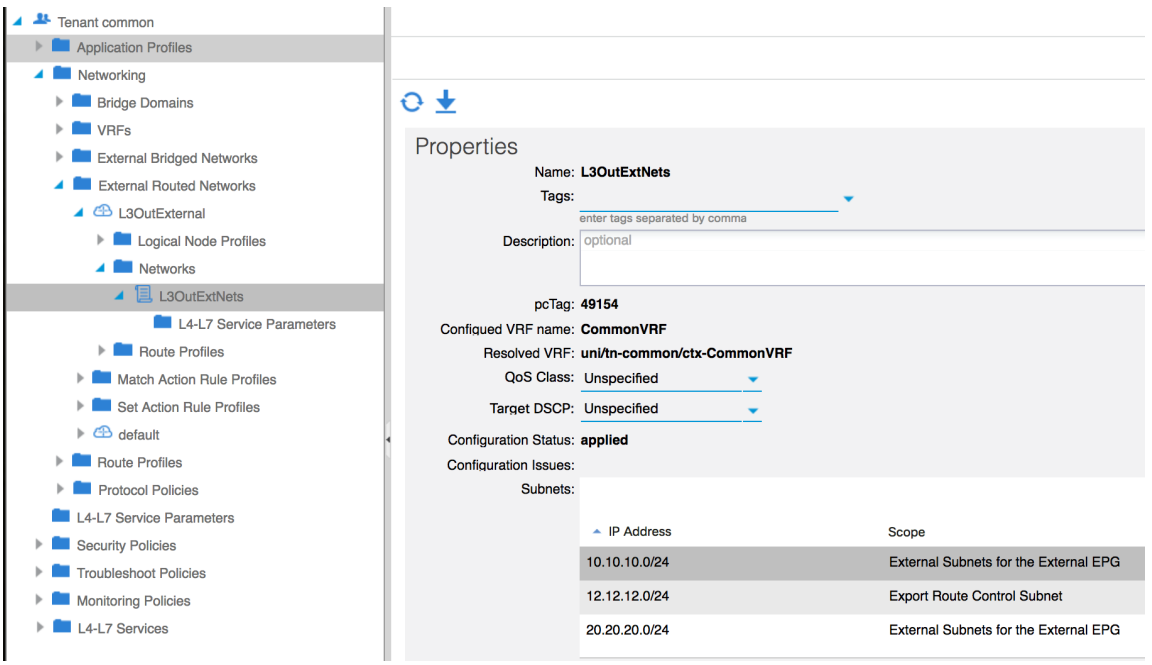
6. A contract will be required for the traffic to pass between the router and the database server.

How it works…

Much of the theory in this recipe has been covered in previous recipes. The ASA was covered in chapter three, we covered tenant creation, as well as bridge domain and VRFs in chapter two, and we looked at creating L3 out interfaces in this chapter. So, why reinvent the wheel with this recipe?

While many of the mechanics of transit routing and route peering happen behind the scenes, there are a few aspects we need to pay attention to. The main one is making sure that we are exporting and importing the correct subnets in the correct direction and with the correct scope.

Looking at the example of L3OutExternal. We are importing the 12.12.12.0/24 and 20.20.20.0/24 subnets and exporting the 10.10.10.0/24 subnet. Route direction is important. If we do not export the 10.10.10.0/24 subnet, the database server will never see it; the reverse is true for the 20.20.20.0/24 subnet. If we do not import the 12.12.12.0/24 subnet, then we will not be able to act as a transit. We also need to set the scope to act as a transit:



Notice that the 12.12.12.0/24 subnet has a scope of "Export Route Control Subnet." This means that the route will be a transit route. The other routes are set as "External Subnets for the External EPG." There is no control of the routing information coming in or going out of the fabric. If the subnet is not marked like this, then although the routes exported from one EPG will reach the EPG that is importing them (showing that the control plane is working), the actual traffic (data plane) will be dropped. This is due to the whitelisting behavior of the fabric, whereby the default is to drop traffic unless it is permitted by the relevant policy. Therefore we would need to implement a contract between the two to permit the traffic.

There's more…

Refer to this link for more information on transit routing and the control flags for the subnet scope feature:

`http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/kb/b_KB_Transit_Routing.html#id_30901`

5
ACI Security

In this chapter, we will be looking at securing the ACI fabric by using the recipes below:

- Creating local users
- Creating security domains
- Limiting users to tenants
- Connecting to a RADIUS server
- Connecting to an LDAP server
- Connecting to a TACACS+ server

Introduction

Given that there will be more than one person administering the ACI fabric, it makes sense that each has their own user account. This is a necessity for certifications such as PCI-DSS, and also just makes sense from an auditing perspective.

In this chapter, we will look at how we can connect to third-party authentication sources, such as RADIUS, TACACS+, and LDAP, and how we can limit the users down by a per-tenant or per-function basis.

AAA and Multiple Tenant Support

ACI has been built with security in mind. Adding local users and connecting to external authentication services (such as RADIUS, TACACS+, and LDAP) is all very straight-forward. Security is a constant theme through working with ACI, just look at contracts for an example.

Because of this security focus, we can perform actions such as limiting the abilities of a user on a per-tenant basis and being very granular on the aspects of the fabric that they can and cannot read or write to. The abilities of a user can be dictated in different ways, for example, a user can have full access to the entire fabric and the tenants within it, or full access to one or more tenants, or even the ability to perform specific actions on one or more tenants.

This is referred to as **Role-Base Access Control** (**RBAC**).

Understanding ACI Role-Based Access Control (RBAC)

There are several preconfigured rules and with these rules come different privileges. Below is a table listing the roles and a brief description of the different privileges that are contained within it.

Role	Description
AAA	For configuring Authentication, Authorization, and Accounting, as well as import and export policies.
Admin	Full access to all fabric features.
Access-Admin	Layer 1-3 configuration, including protocols for tenants, as well as fabric-wide settings (NTP, SNMP, and DNS)
Fabric-Admin	Layer 1-3 configuration, including protocols for the fabric
NW-SVC-Admin & NW-SVC-Params	Managing L4-L7 services
OPS	Monitoring and troubleshooting
Read-All	Read-all access (to everything)
Tenant-Admin	Administer all aspects of a tenant
Tenant-Ext-Admin	Externally-focused policy and configuration items (such as L3 and L2 Outs).
VMM-Admin	Virtual Machine Manager connectivity, inventory, and policies.

You can also create custom roles if you need to.

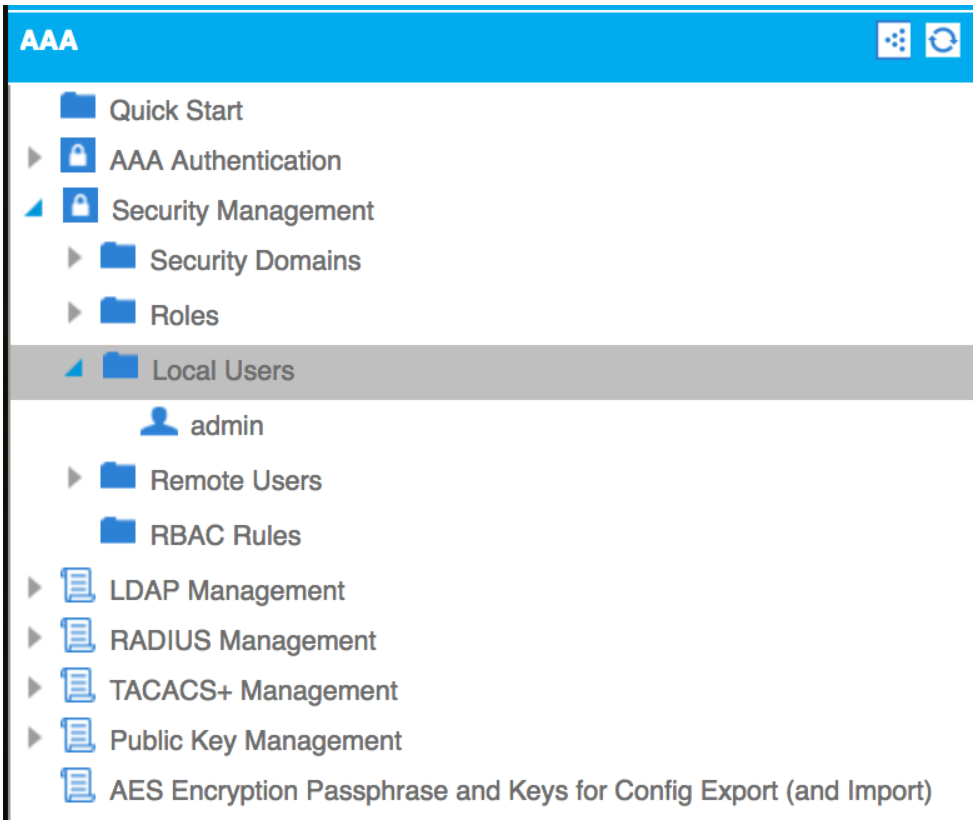
The table above gives a very brief overview. For a full list with all of the role privileges, take a look at `http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/kb/b_KB_AAA-RBAC-roles-privileges.html`.

Creating local users

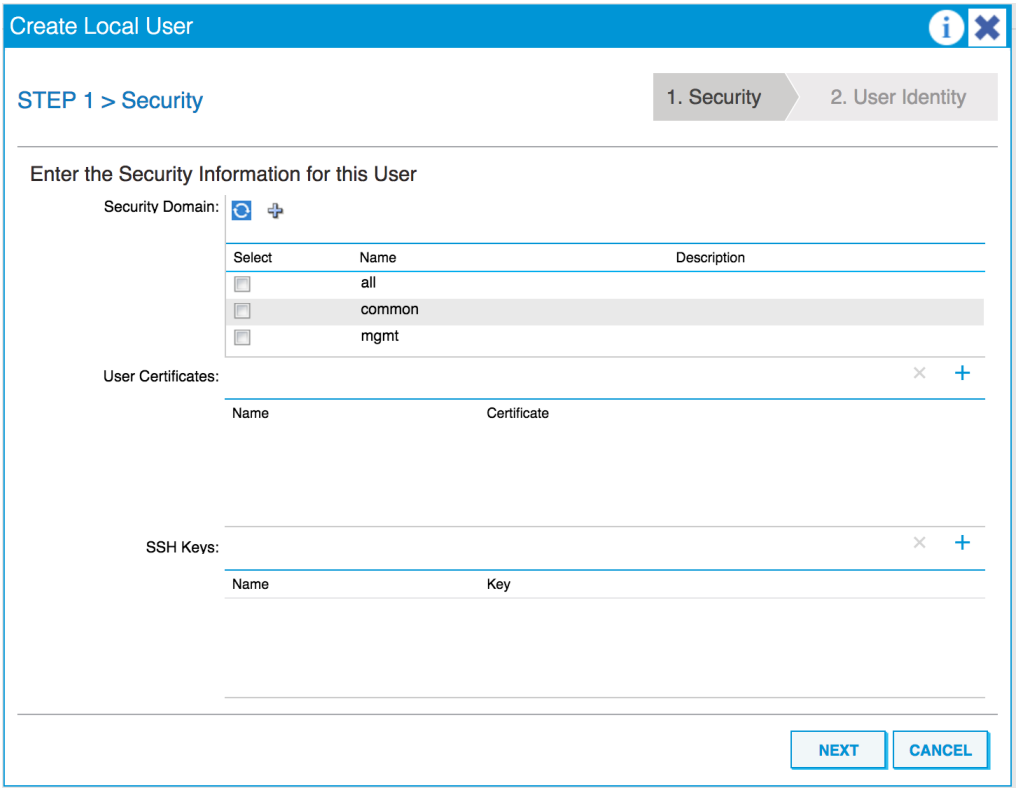
Local users are the easiest way to start segregating users and leveraging some form of accountability. We will have a bigger administrative overhead, and clearly, this would not be the preferred solution. Instead one would look to a centralized system, such as RADIUS or LDAP. However, local users are a good place for us to start.

How to do it…

1. Navigate to Admin > AAA > Security Management > Local Users.

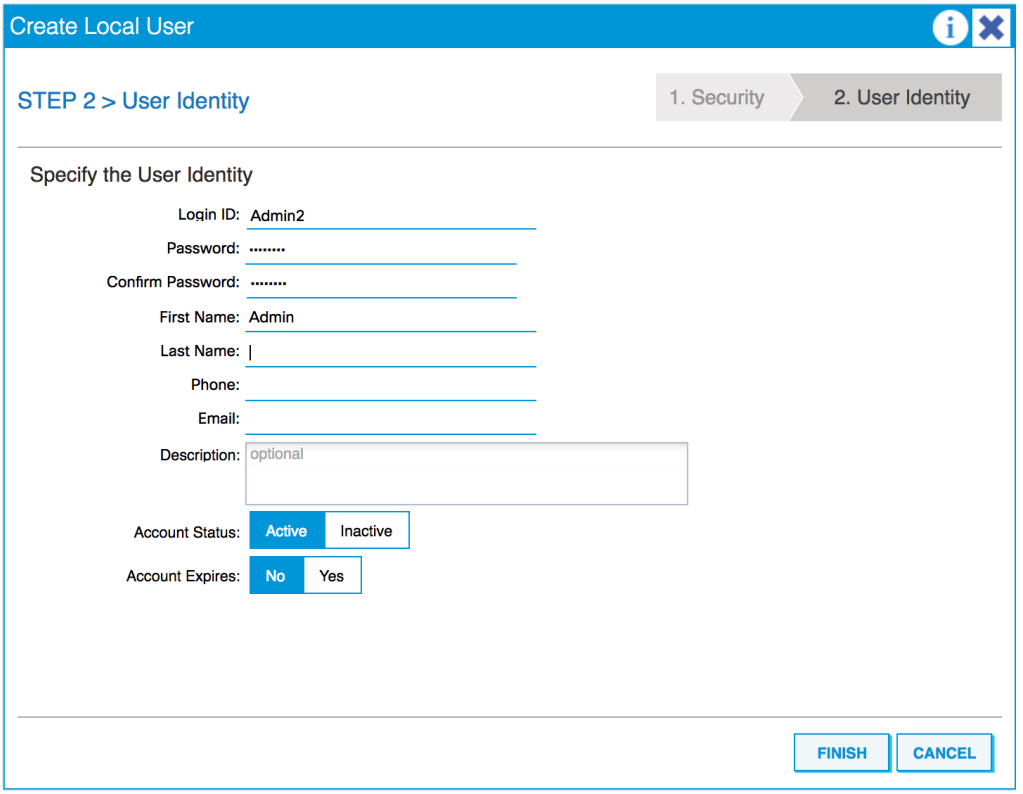


2. Click on **Actions** > **Create Local User**.
3. Select a Security Domain, or leave it at the default (all unticked).



1. Figure 218.

2. Click **Next**.
3. Enter the Login ID, and the password, fill in any other fields if desired.



4. Click **Finish**.

How it works…

We can test the new user's access by connecting to the APIC with SSH.

```
Stuarts-MacBook-Pro:~ stuart$ ssh Admin2@192.168.1.205
 Application Policy Infrastructure Controller
Admin2@192.168.1.205's password:
 apic1#
```

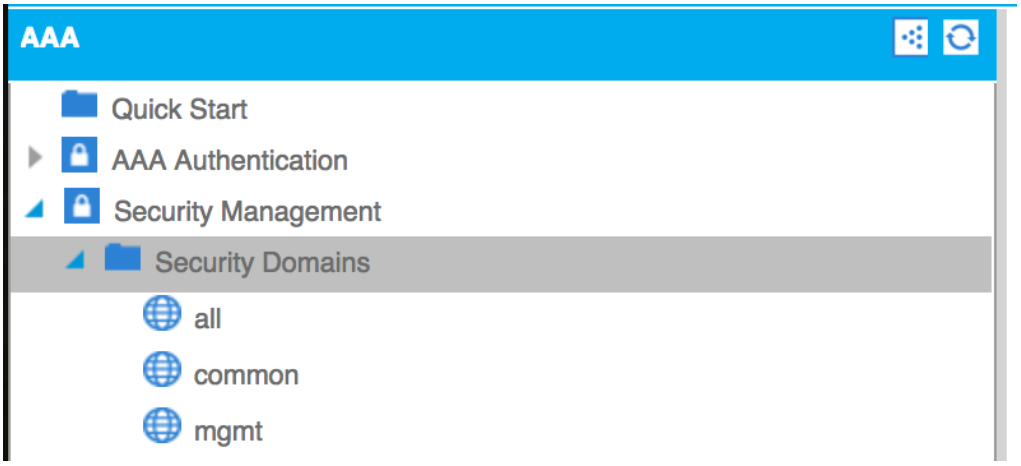
The connection is successful. But what if need to limit down access to a particular tenant? For that, we need to create a security domain.

Creating security domains

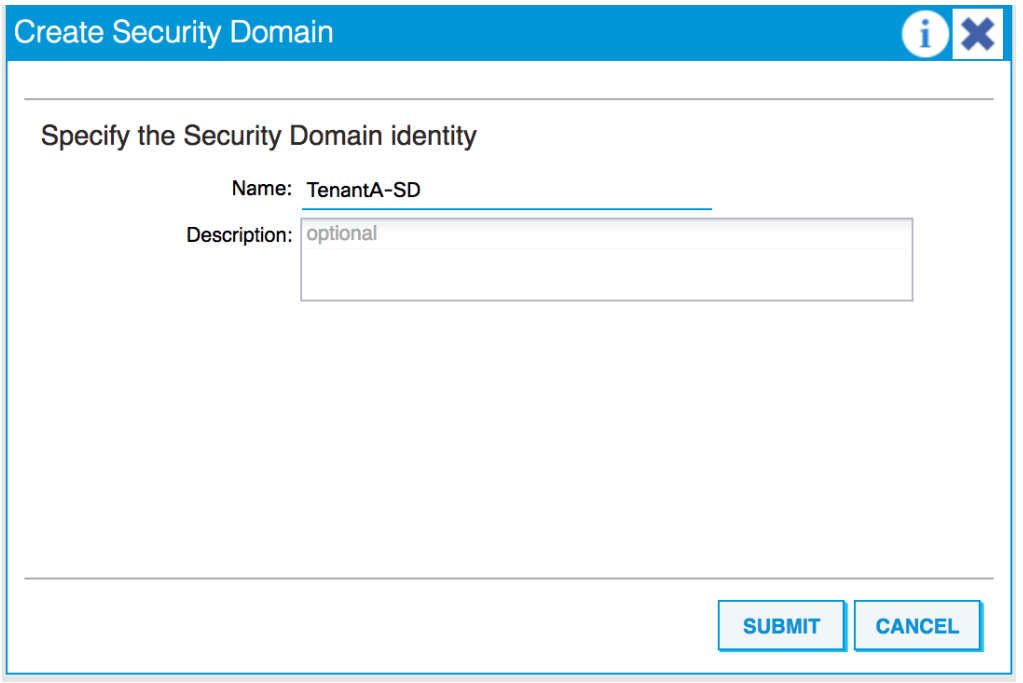
Security domains allow us to permit or deny administrators based on the tenants added as "associated objects" within the domain.

How to do it…

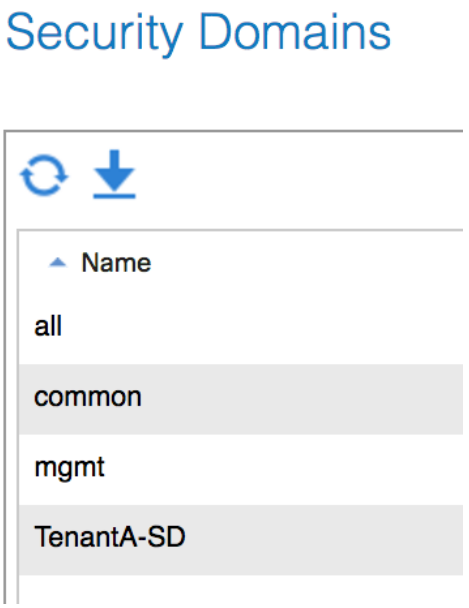
1. Navigate to Admin > AAA > Security Management > Security Domains.



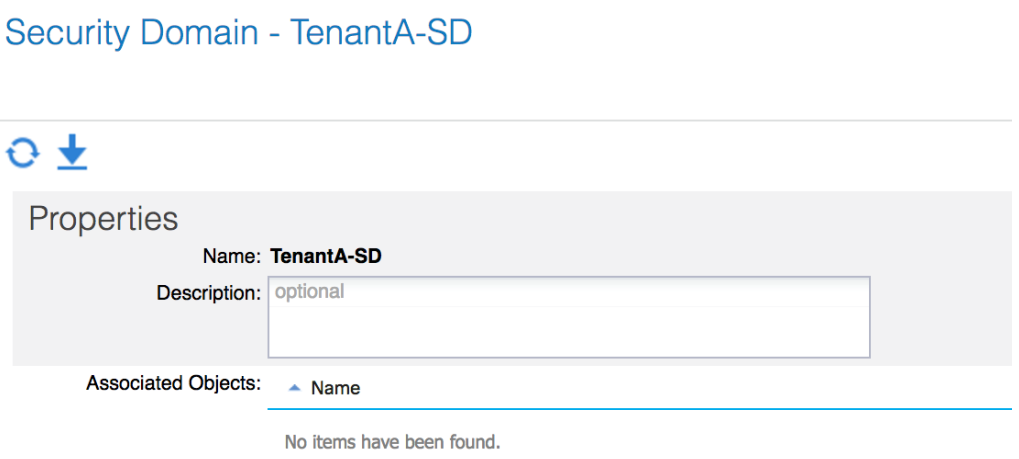
2. Click on Actions, then select **Create Security Domain**.
3. Name the new Security Domain.



4. Click **Submit**.
5. New Security Domain will be listed with the default ones.

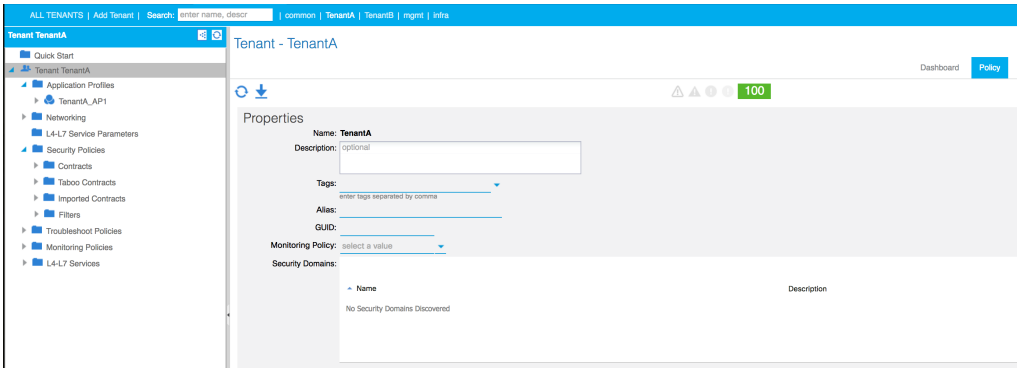


6. If you click on the security domain, you will see that there are no associated objects (tenants).

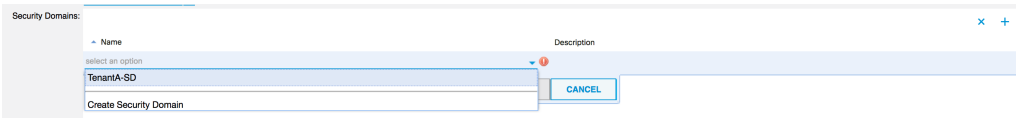


7. To associate a tenant to a security domain, navigate to the tenant (TenantA) and

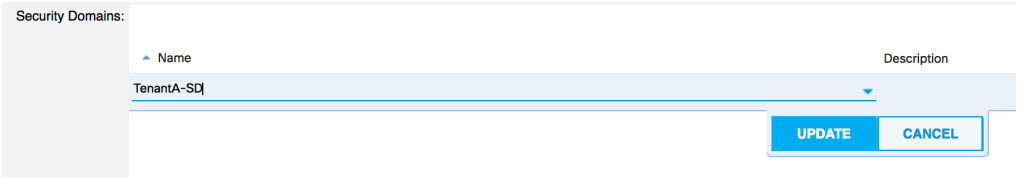
click the Policy tab.



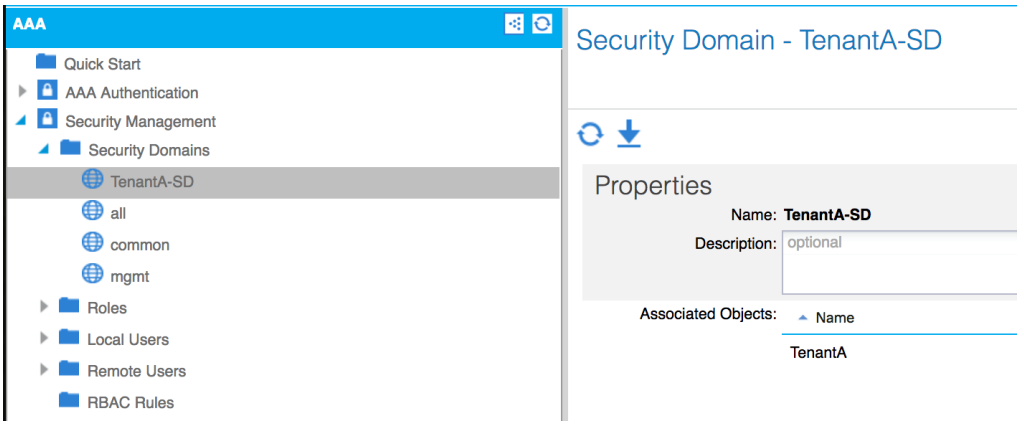
8. Click the plus sign next to security domains, and select the Tenant-SD from the drop-down menu.



9. Click on Update



10. If you return to the TenantA-SD security domain in the AAA tab (step 1). You can see that TenantA is now listed under associated objects.

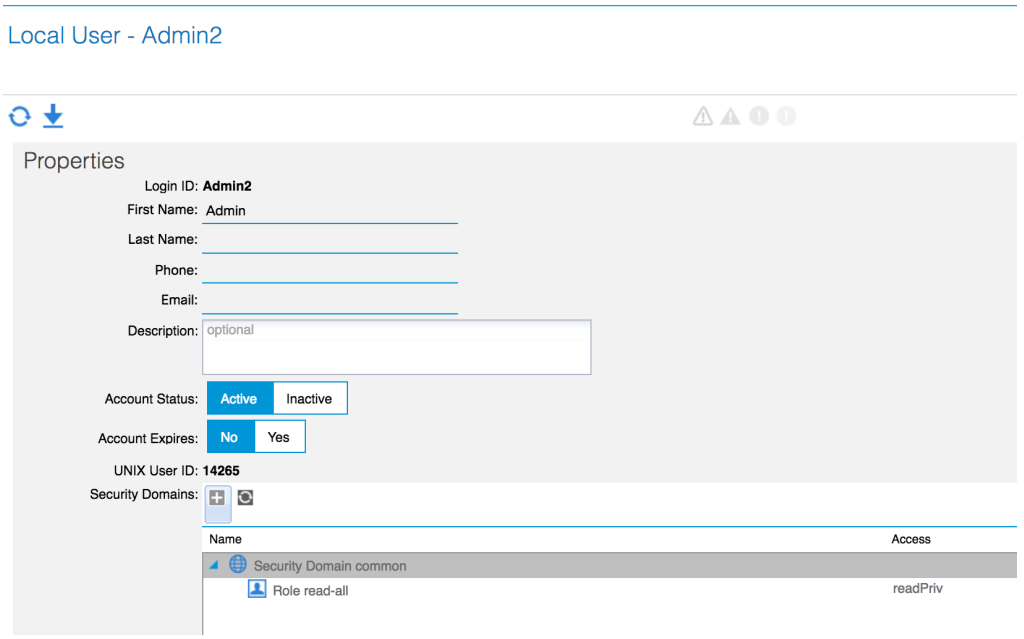


Limiting users to tenants

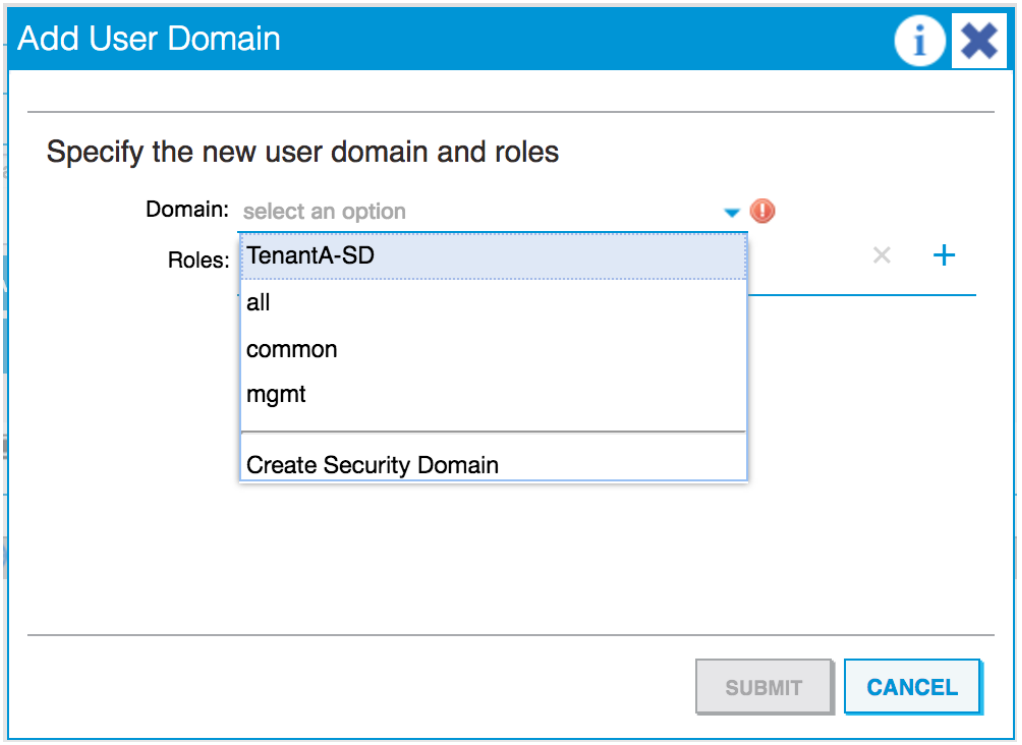
Now that we have a new security domain let's set up the Admin2 user to use it.

How to do it…

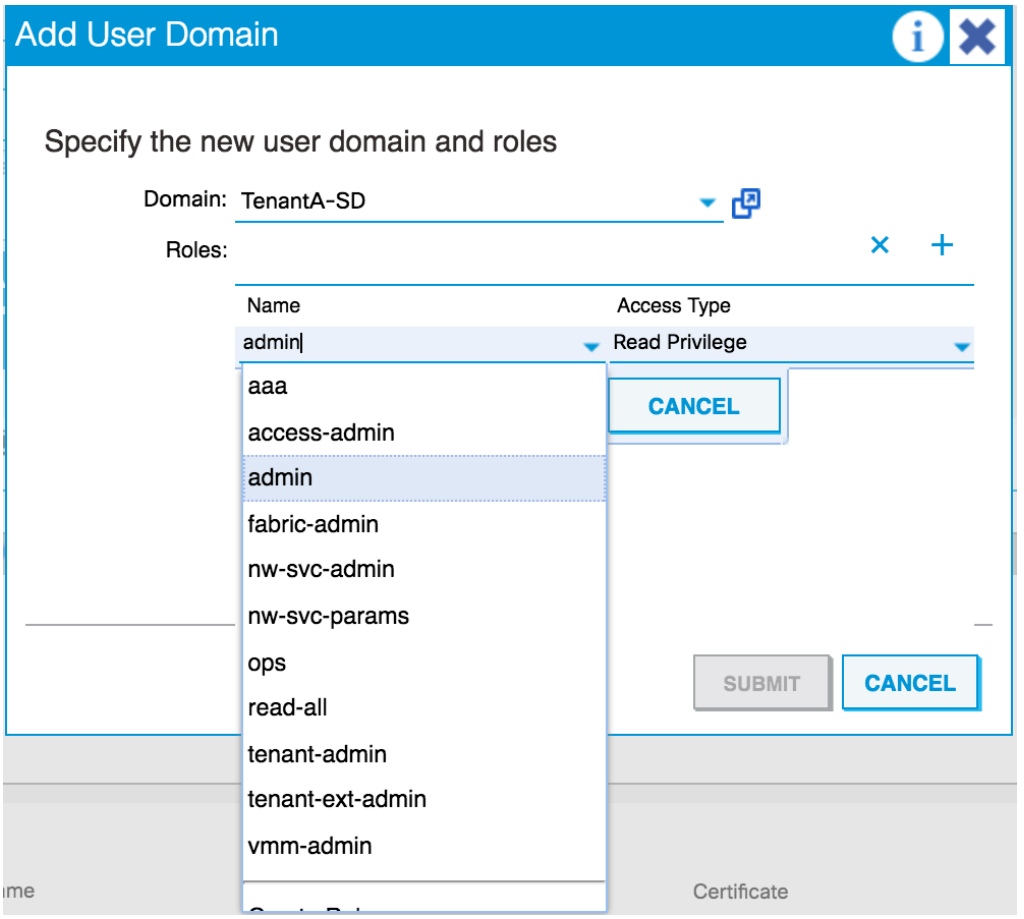
1. Namvigate t0 **Admin** > **AAA** > **Security Management** > **Local Users**.
2. Select the **Admin2** user.



3. Click on the plus sign next to Security Domains.
4. Select TenantA-SD from the Domain drop-down.



5. Select the required role and access type.



6. Click **Submit**.

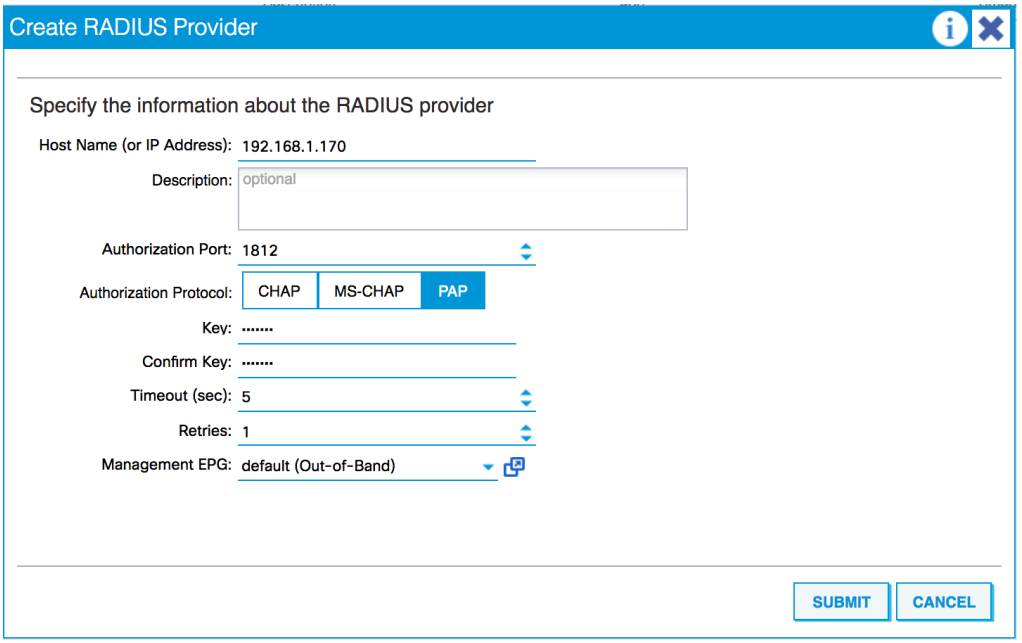
Connecting to a RADIUS server

The ACI fabric supports CHAP, MS-CHAP, and PAP as authorization protocols. In this recipe, we will use PAP to authenticate to a Windows 2008 server, running the RADIUS protocol.

How to do it…

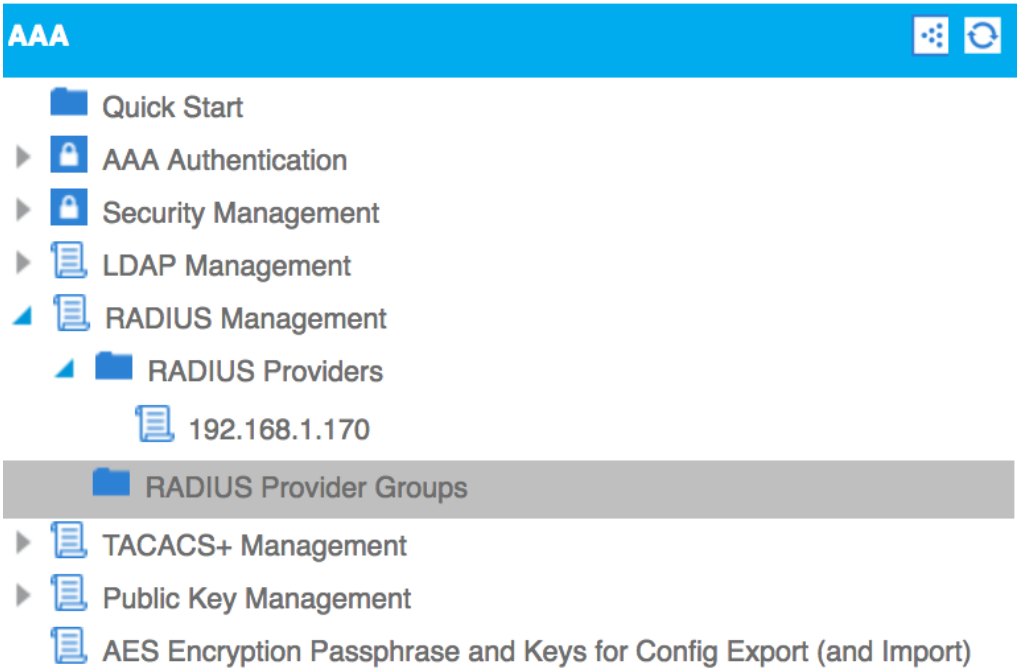
1. Navigate to **Admin** > **AAA** > RADIUS Management. Select RADIUS Providers.

2. From the Actions menu, select **Create RADIUS Provider**.
3. Enter the IP address of the RADIUS server, choose the authorization protocol and enter the key, along with the Management EPG.

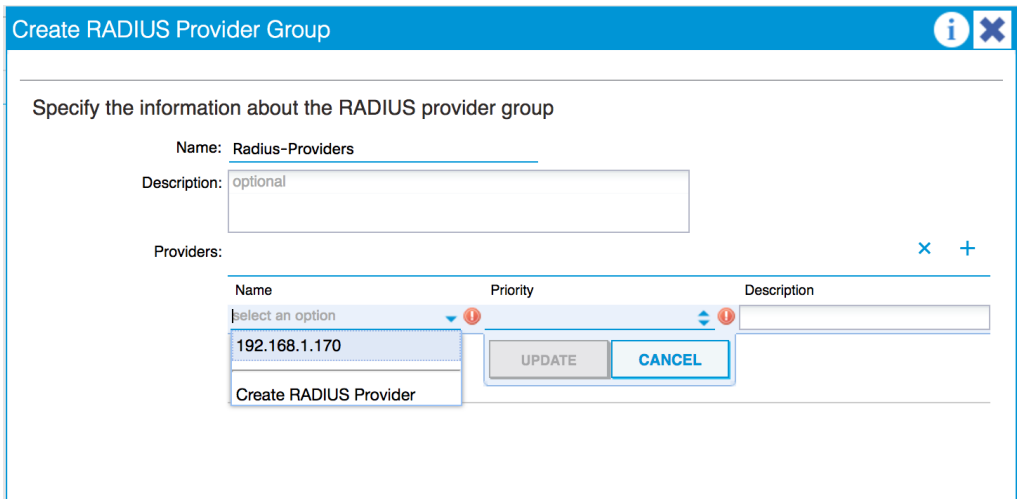
1. 

1. If you use a management EPG other than the default (Out-of-Band) one, make sure that it has access to the RADIUS, LDAP, or TACACS+ server!

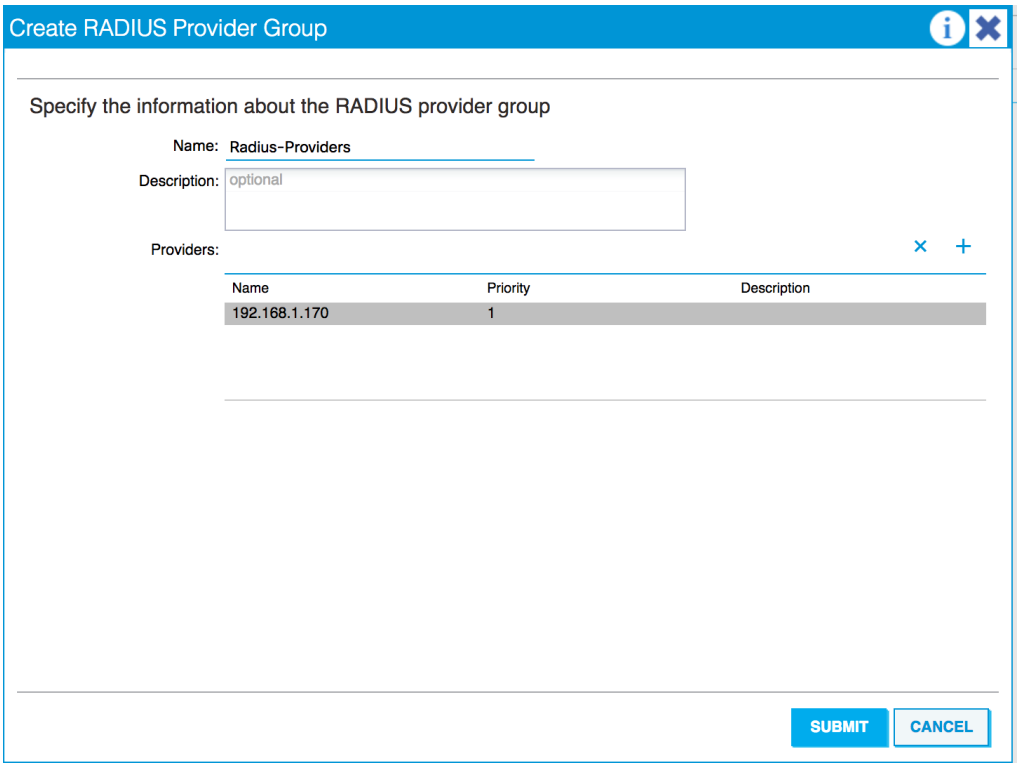
2. Click **Submit**.
3. Select **Admin** > **AAA** > RADIUS Management > RADIUS Provider Groups.



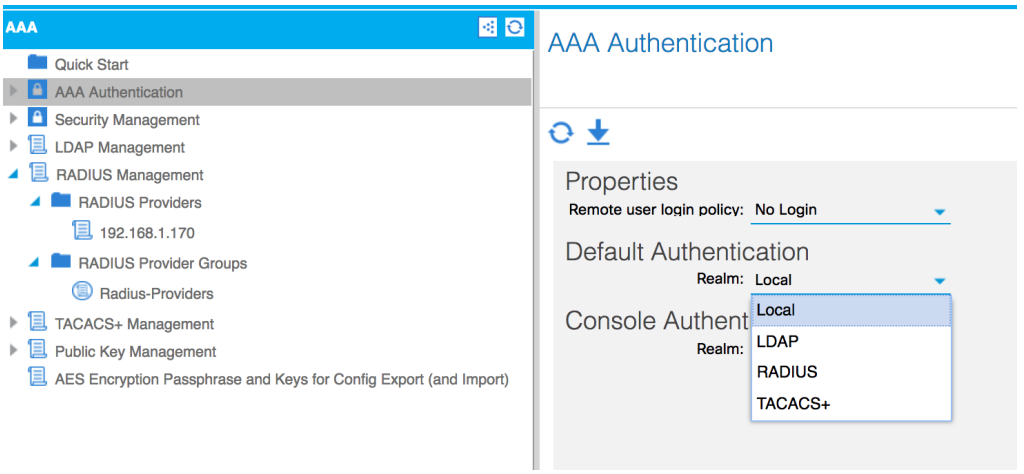
4. Click on **Actions** > **Create RADIUS Provider Group**.
5. Name the group, and select the provider created previously from the drop-down.



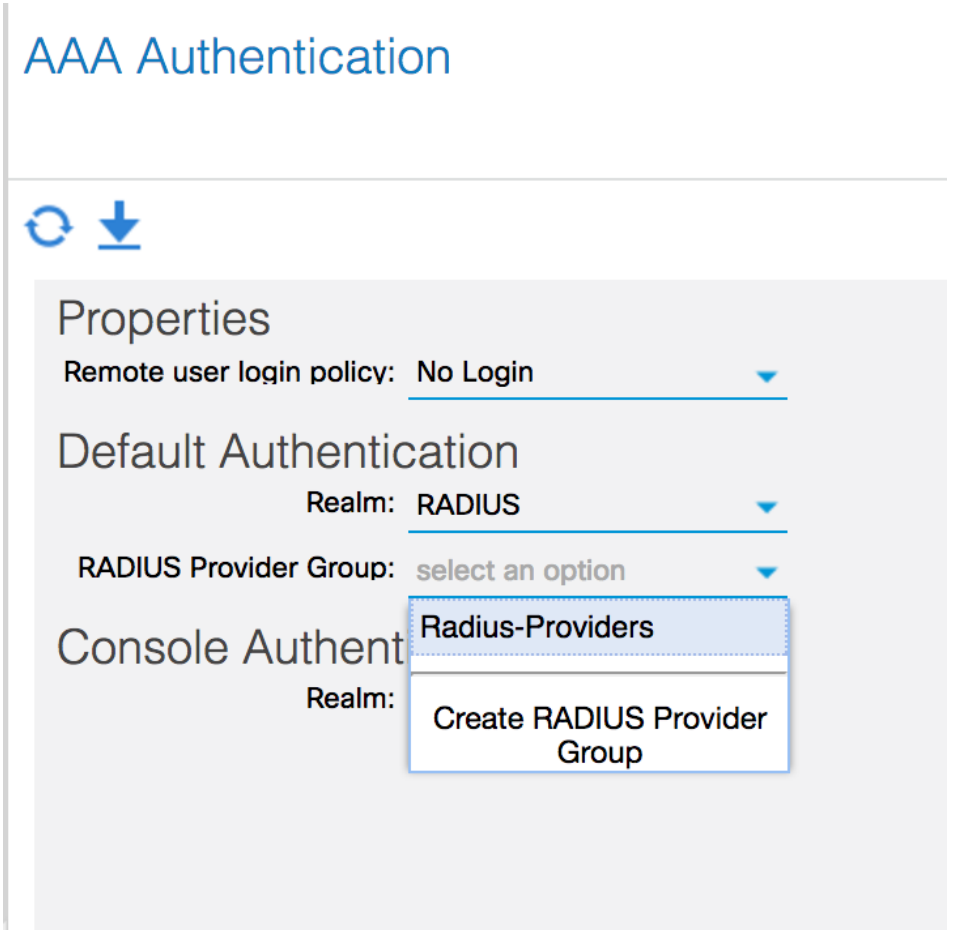
6. Set the priority and click **Update**.



7. Click **Submit.**
8. From the main AAA Authentication menu, change the default authentication realm to RADIUS.



9. Select the RADIUS provider group created in step 7 from the drop-down list.



10. Optionally, you can set the console to use RADIUS authentication as well. It is wise to make sure that you can log in before setting console authentication.

How it works…

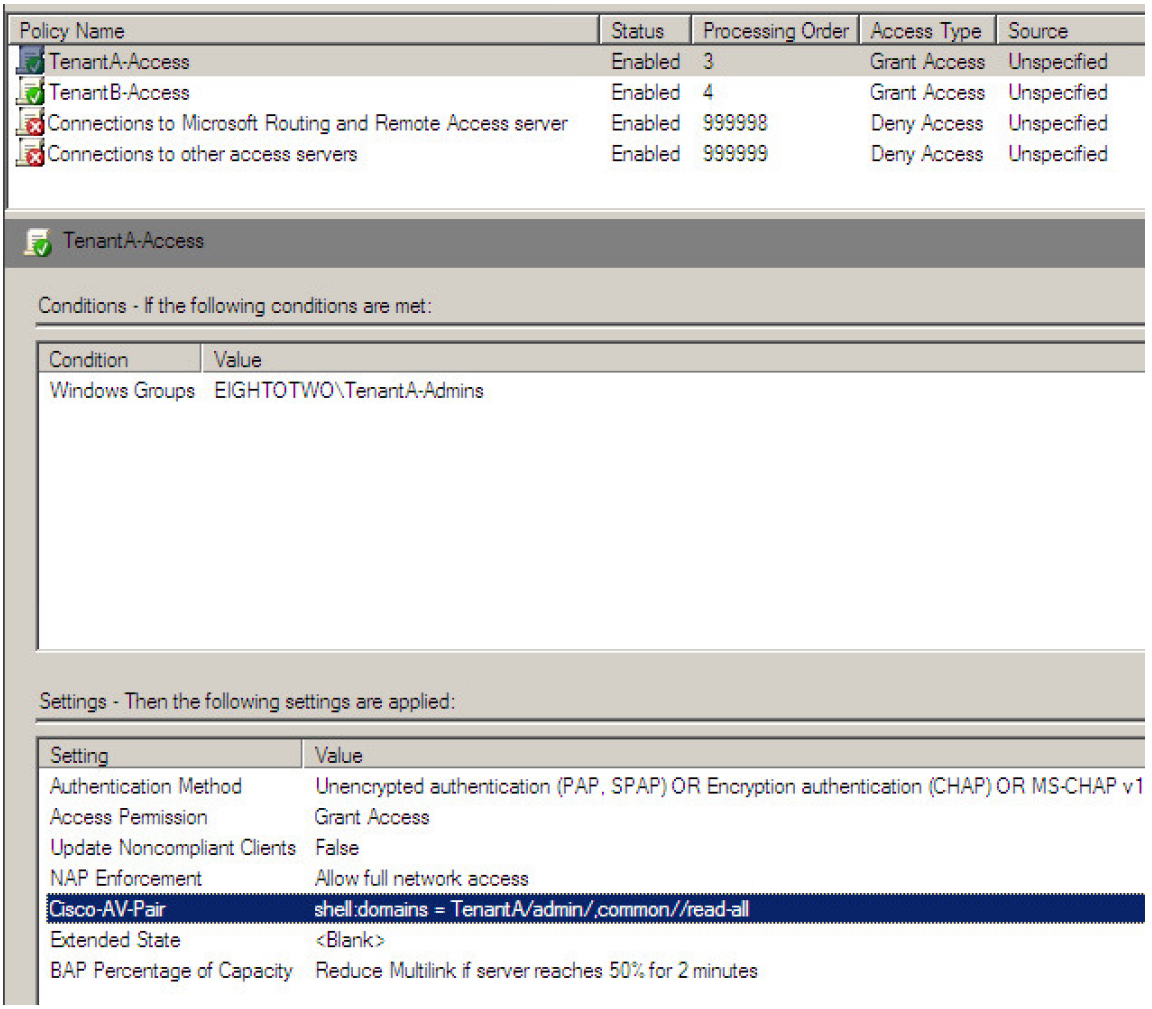
If we have a Windows 2008 server, we can use the **NPS** role (**Network Policy Server**).

With this role installed, and with two AD users (AdminA and AdminB), each in different AD groups (TenantA-Admins and TenantB-Admins, respectively), we can test RADIUS access.

Create a NAP client, specifying the IP address of the APIC, along with the password

entered in step 3.

Create a Network Policy with the following settings:



Policy Name	Status	Processing Order	Access Type	Source
TenantA-Access	Enabled	3	Grant Access	Unspecified
TenantB-Access	Enabled	4	Grant Access	Unspecified
Connections to Microsoft Routing and Remote Access server	Enabled	999998	Deny Access	Unspecified
Connections to other access servers	Enabled	999999	Deny Access	Unspecified

TenantA-Access

Conditions - If the following conditions are met:

Condition	Value
Windows Groups	EIGHTOTWO\TenantA-Admins

Settings - Then the following settings are applied:

Setting	Value
Authentication Method	Unencrypted authentication (PAP, SPAP) OR Encryption authentication (CHAP) OR MS-CHAP v1
Access Permission	Grant Access
Update Noncompliant Clients	False
NAP Enforcement	Allow full network access
Cisco-AV-Pair	shell:domains = TenantA/admin/,common//read-all
Extended State	<Blank>
BAP Percentage of Capacity	Reduce Multilink if server reaches 50% for 2 minutes

The Cisco AV-Pair controls what we have access to.

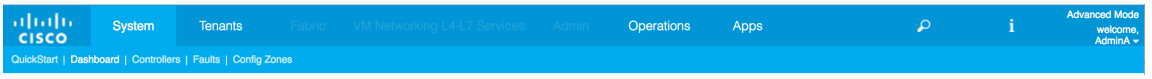
Let's try logging into the APIC.

```
Stuarts-MacBook-Pro:~ stuart$ ssh AdminA@192.168.1.205
Application Policy Infrastructure Controller
AdminA@192.168.1.205's password:
apic1#
```

We can log into the APIC through SSH. Let's try the GUI.

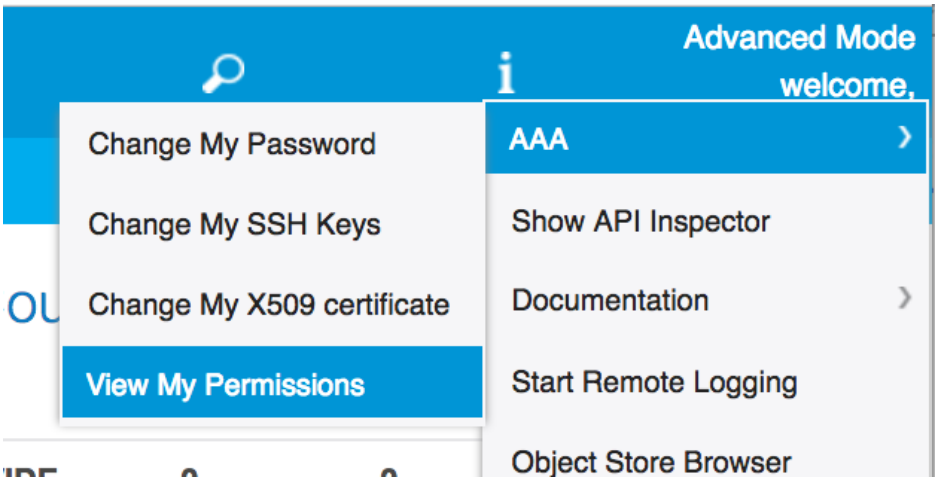
If you find that you are locked out, then you can get in by logging into the GUI with the following method:
Username: **apic:fallback<username>**
Here, we can use our local user accounts (we would replace "<username>" with "admin") because we are bypassing the RADIUS server using the prefix "apic:fallback."

If we login to the GUI, we can see the following:

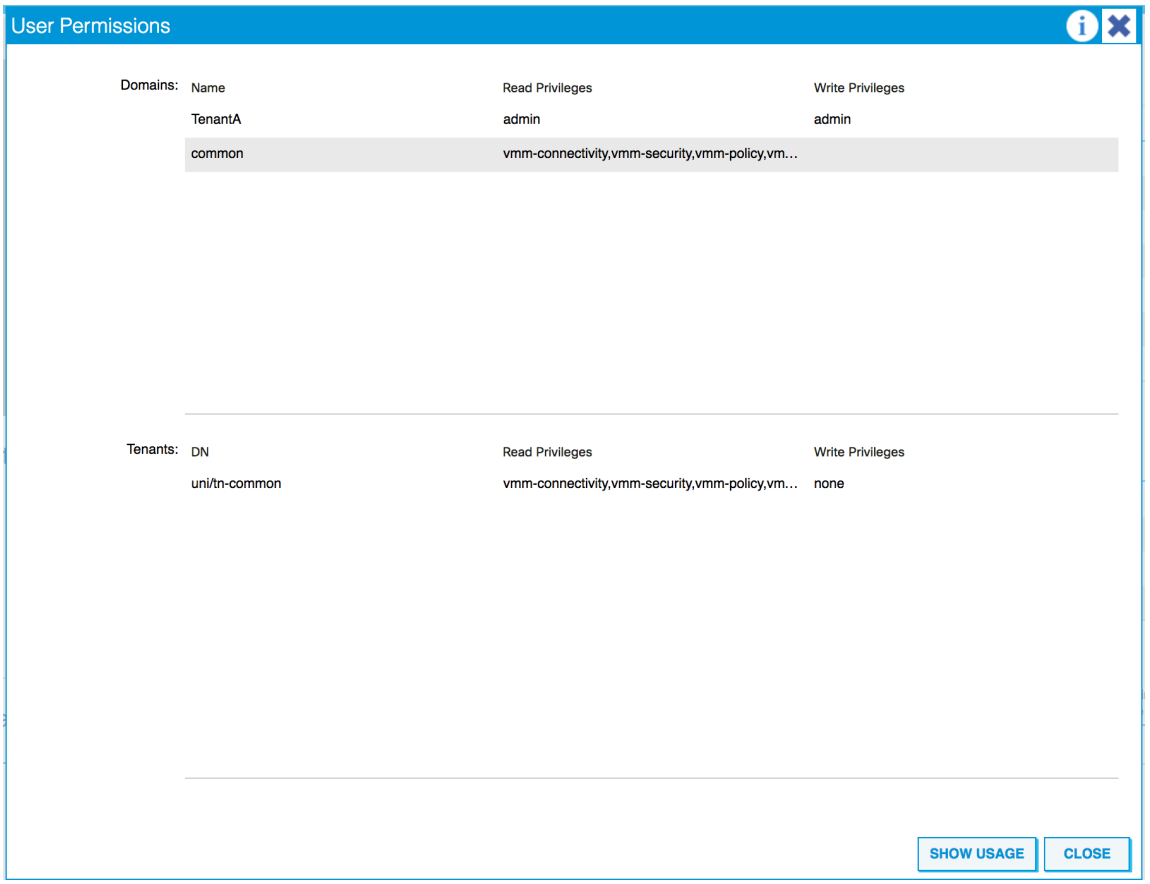


Our abilities are significantly reduced, as you can see because the Fabric, VM Networking, L4-L7 Services, and Admin tabs are all grayed out (or blued out to be more precise).

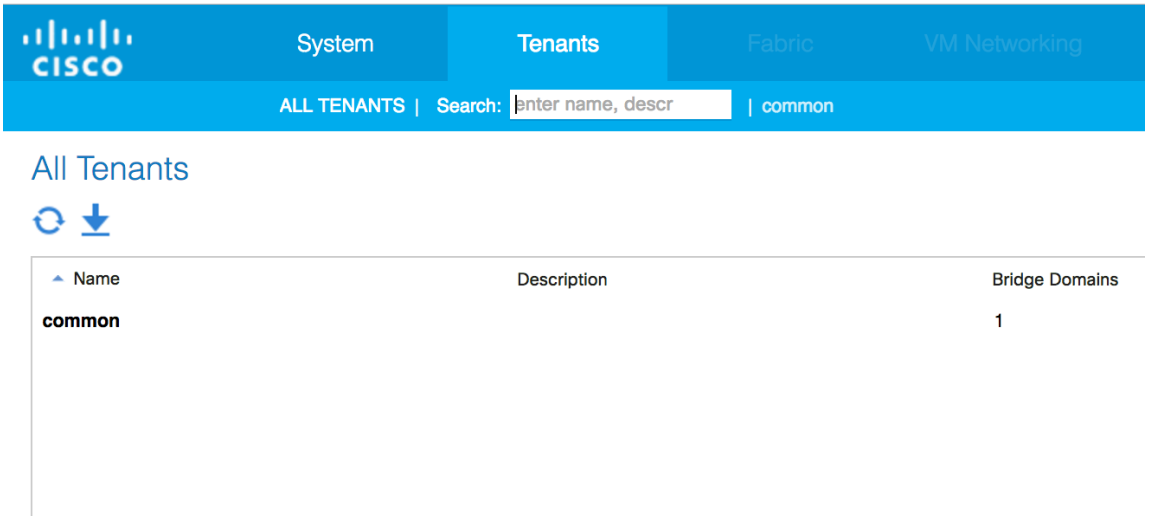
We can look at our permissions from the drop-down arrow next to our username:



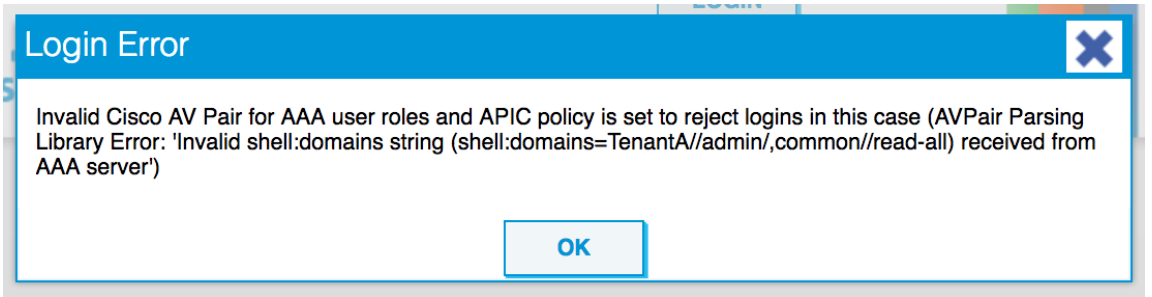
Clicking **View My Permissions** shows us our permissions:



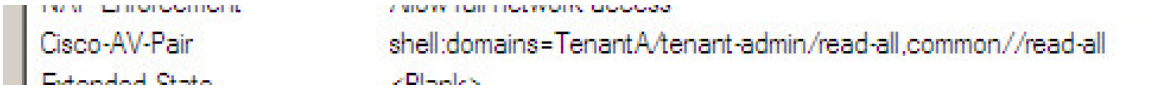
As you can see, we do not have access to the TenantA tenant, as we can see by clicking the Tenants tab:



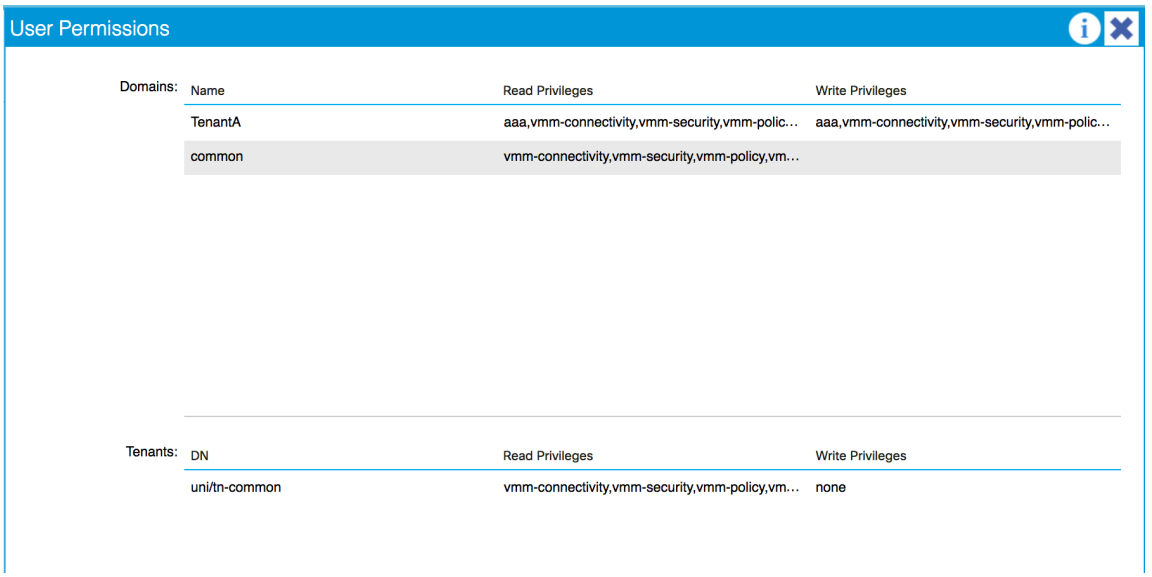
Syntax-wise, the AP-pair is correct. If we introduce an error in it (double-slash instead of single), we are told that the AV pair is invalid when we try and log in:



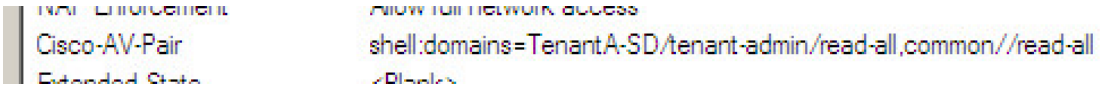
Let's try a different AP-pair.



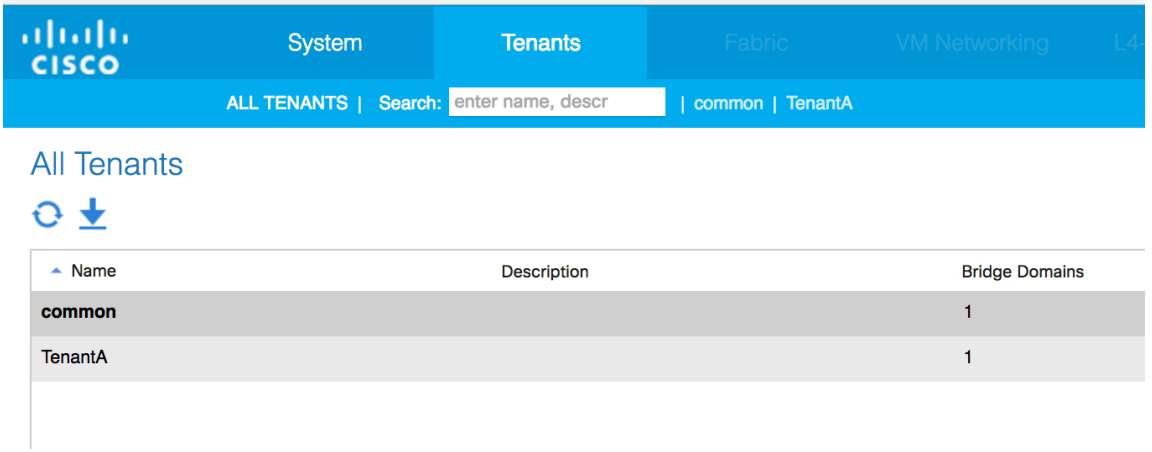
Now can we see the tenant?



No, but let's step back and think about this logically. I have been adding a Tenant in the domain, and not in the tenant area. Instead of referencing TenantA, I should have been referencing TenantA-SD:



Does it work now?



Bingo!

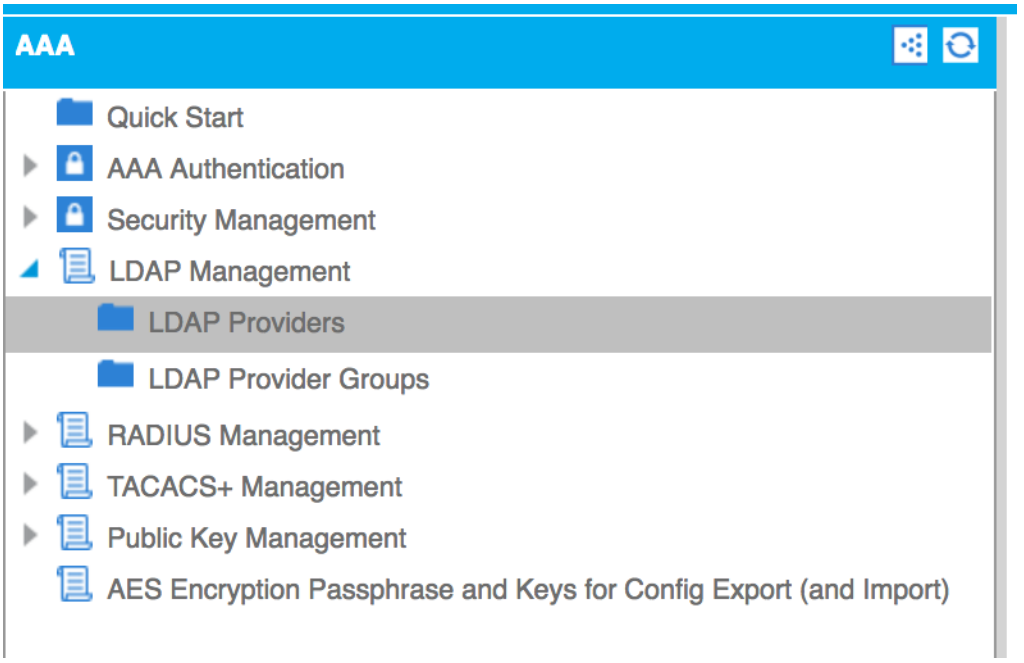
Getting the AP-Pair correct is probably the trickiest part of using an external authentication source, such as RADIUS. Setting up the providers is very similar, as we will see when we create an LDAP provider.

Connecting to an LDAP server

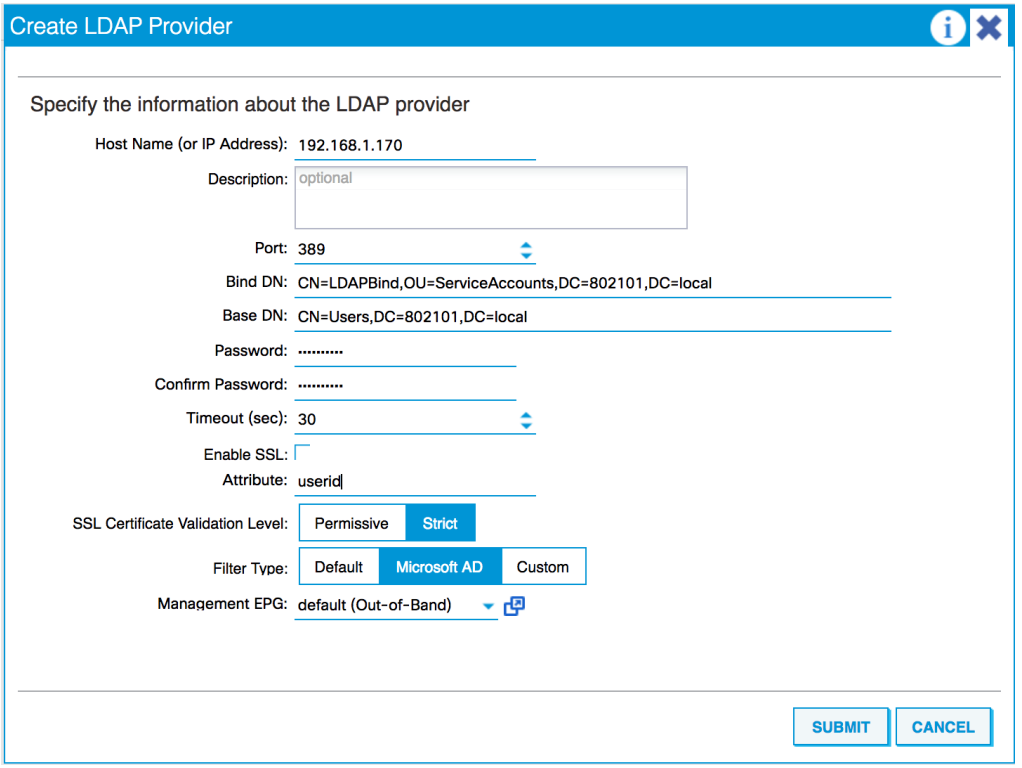
As well as RADIUS and TACACS+, we can connect to an LDAP server for authentication.

How to do it…

1. Navigate to **Admin** > **AAA** > **LDAP Management** > **LDAP Providers**.



2. Actions > Create LDAP Provider.
3. Enter the settings to connect to the AD server.



4. Click **Submit**.
5. Navigate to Admin > AAA > LDAP Management > LDAP Provider Groups.
6. Select Actions > Create LDAP Provider Group.
7. Add the server created in step 3 and set the priority.
8. Click Submit.
9. Select AAA Authentication and set the default authentication to LDAP and the LDAP provider group to the provider group created in step 7.
10. Optionally, set the console authentication to LDAP.
11. Click **Submit**.

Connecting to a TACACS+ server

The steps for adding a TACACS+ server are similar to both RADIUS and LDAP.

How to do it…

1. Navigate to Admin > AAA > TACACS+ Management > TACACS+ Providers
2. Select Actions > Create TACACS+ Provider
3. Set the IP address, port (if different from the default of 49), authorization protocol, key, and select the management EPG.
4. Click Submit.
5. Navigate to Admin > AAA > TACACS+ Management > TACACS+ Provider Groups
6. Select Actions > Create TACACS+ Provider Group
7. Name it and add the provider created in step 3.
8. Click Submit.
9. Select AAA Authentication and set the default authentication to TACACS+ and the TACACS+ provider group to the provider group created in step 7.
10. Optionally, set the console authentication to TACACS+.
11. Click Submit.

If you have any feedback on this eBook or are struggling with something we haven't covered, let us know at `https://goo.gl/du9Y3e`.

.

If you have any concerns you can also get in touch with us at `customercare@packtpub.com`

We will send you the next chapters when they are ready……!

Hope you like the content presented.