CHAPTER 4
VLANs

Virtual LANs, or VLANs, are virtual separations within a switch that provide distinct logical LANs that each behave as if they were configured on a separate physical switch. Before the introduction of VLANs, one switch could serve only one LAN. VLANs enabled a single switch to serve multiple LANs. Assuming no vulnerabilities exist in the switch's operating system, there should be no way for a frame that originates on one VLAN to make its way to another.

When I wrote the first edition of *Network Warrior*, I had not yet learned of VLAN-hopping exploits. There are ways to circumvent the VLAN barrier. Attacks such as switch spoofing and double tagging are methods used to gain access to one VLAN from another. Though reading data from another VLAN is not so easy, denial-of-service attacks could be accomplished through VLAN hopping, especially where trunks interconnect switches. Don't rely on VLANs as your sole means of security, especially in high-risk environments.

Connecting VLANs

Figure 4-1 shows a switch with multiple VLANs. The VLANs have been numbered 10, 20, 30, and 40. In general, VLANs can be named or numbered; Cisco's implementation uses numbers to identify VLANs by default. The default VLAN is numbered 1. If you plug a number of devices into a switch without assigning its ports to specific VLANs, all the devices will reside in VLAN 1.

Frames cannot leave the VLANs from which they originate. This means that in the example configuration, Jack can communicate with Jill, and Bill can communicate with Ted, but Bill and Ted cannot communicate with Jack or Jill in any way.

For a packet on a Layer-2 switch to cross from one VLAN to another, an outside router must be attached to each of the VLANs to be routed. Figure 4-2 shows an external router connecting VLAN 20 with VLAN 40. Assuming a proper configuration on the

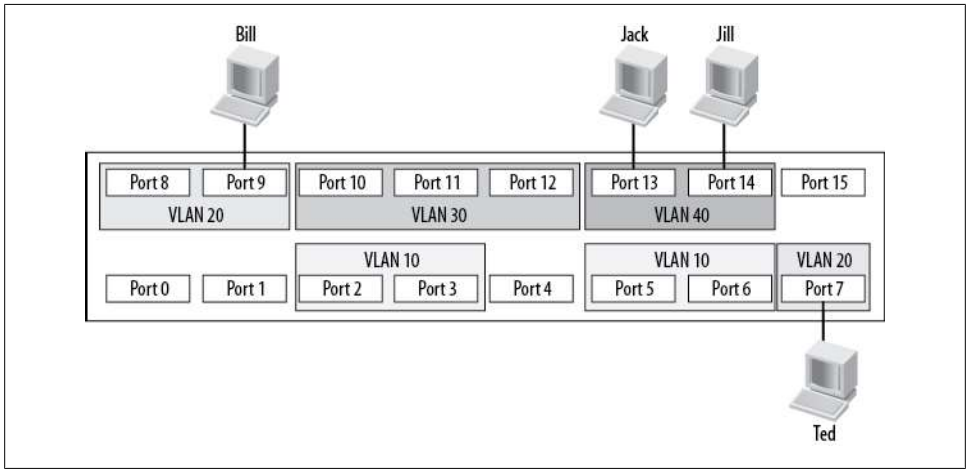

Figure 4-1. VLANs on a switch

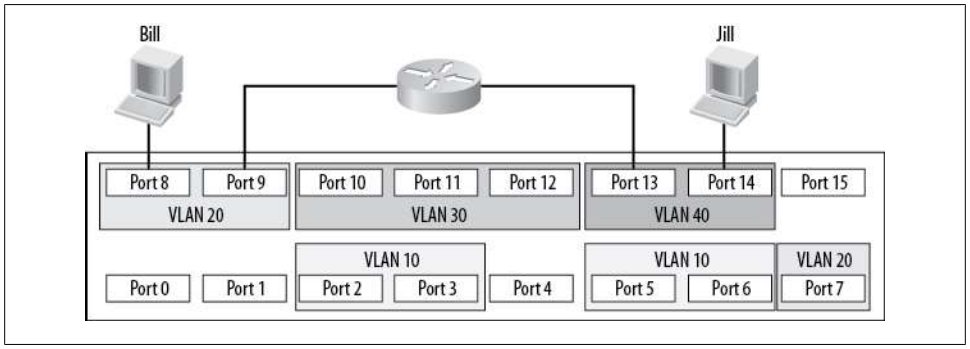

Figure 4-2. External routing between VLANs

router, Bill will now be able to communicate with Jill, but neither workstation will show any indication that they reside on the same physical switch.

When expanding a network using VLANs, you face the same limitations. If you connect another switch to a port that is configured for VLAN 20, the new switch will be able to forward frames only to or from VLAN 20. If you wanted to connect two switches, each containing four VLANs, you would need four links between the switches: one for each VLAN. A solution to this problem is to deploy *trunks* between switches. Trunks are links that carry frames for more than one VLAN.

Figure 4-3 shows two switches connected with a trunk. Jack is connected to VLAN 20 on Switch B, and Diane is connected to VLAN 20 on Switch A. Because there is a trunk connecting these two switches together, assuming the trunk is allowed to carry traffic for all configured VLANs, Jack will be able to communicate with Diane. Notice that the ports to which the trunk is connected are not assigned VLANs. These ports are *trunk ports* and, as such, do not belong to a single VLAN.

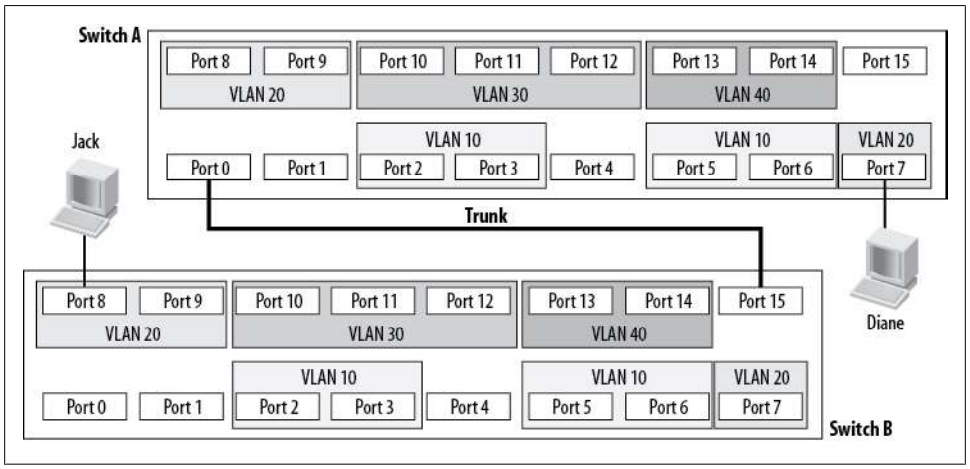

Figure 4-3. Two switches connected with a trunk

Trunks also allow another possibility with switches. Figure 4-2 shows how two VLANs can be connected with a router, as if the VLANs were separate physical networks. Imagine you want to route between *all* of the VLANs on the switch. How would you go about implementing such a design? Traditionally, the answer would be to provide a single connection from the router to each network to be routed. On this switch, each network is a VLAN, so you'd need a physical connection between the router and each VLAN.

As you can see in Figure 4-4, with this setup, four interfaces are being used both on the switch and on the router. Smaller routers rarely have four Ethernet interfaces, though, and Ethernet interfaces on routers can be costly. Additionally, users buy switches with a certain port density in mind. In this configuration, a quarter of the entire switch has been used up just for routing between VLANs.

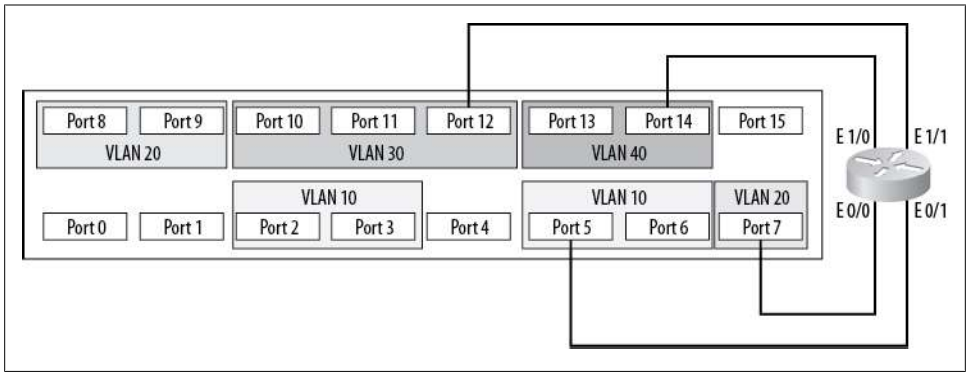

Figure 4-4. Routing between multiple VLANs

Another way to route between VLANs is commonly known as the *router-on-a-stick* configuration. Instead of running a link from each VLAN to a router interface, you can run a single trunk from the switch to the router. All the VLANs will then pass over a single link, as shown in Figure 4-5.

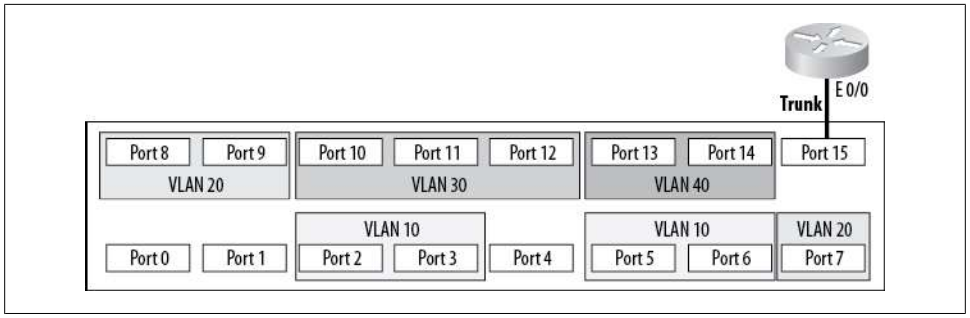

Figure 4-5. Router on a stick

Deploying a router on a stick saves a lot of interfaces on both the switch and the router. The downside is that the trunk is only one link, and the total bandwidth available on that link is only 10 Mbps. In contrast, when each VLAN has its own link, each VLAN has 10 Mbps to itself. Also, don't forget that the router is passing traffic between VLANs, so chances are each frame will be seen twice on the same link—once to get to the router, and once to get back to the destination VLAN.

When I edited this chapter for the second edition, I briefly contemplated updating all the drawings to bring them more in line with currently common interface speeds. I decided against it because the last time I saw anyone doing router on a stick, the fastest switches only had 10M interfaces. It had nothing to do with me being too lazy to change the drawings.

To be painfully accurate, running a trunk to a firewall and having the firewall perform default gateway functions for multiple VLANs employs the same principle, so you could argue that I'm just too lazy after all. I would then counter with the fact that the latest switching technology from Cisco, the Nexus line, has done away with interface names that describe their speed, and instead names all Ethernet interfaces as "`e slot/port`". Therefore, I'm not lazy, but rather forward-thinking.

Then I started writing the VoIP chapter where, lo and behold, I configured router on a stick in order to get my Voice-VLAN trunked to the router. Good thing I didn't pull the router-on-a-stick section. It looks like I was being forward-thinking after all.

Figure 4-6 shows conceptually how the same design would be accomplished with a Layer-3 switch. Because the switch contains the router, no external links are required.

With a Layer-3 switch, every port can be dedicated to devices or trunks to other switches.

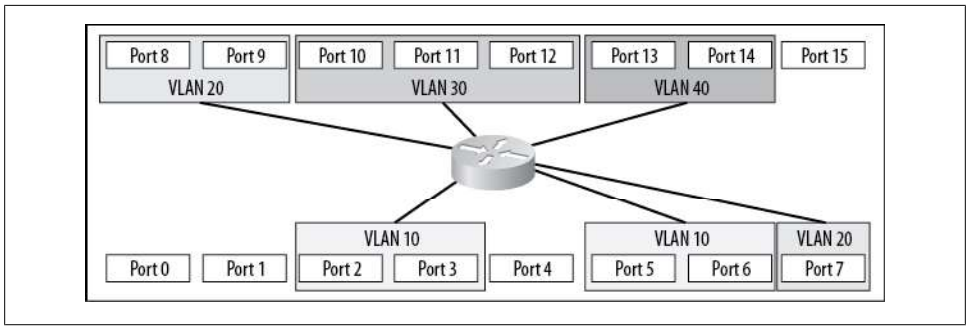

Figure 4-6. Layer-3 switch

Configuring VLANs

VLANs are typically configured via the CatOS or IOS command-line interpreter (CLI), like any other feature. However, some IOS models, such as the 2950 and 3550 switches, have a configurable *VLAN database* with its own configuration mode and commands. This can be a challenge for the uninitiated, especially because the configuration for this database is completely separate from the configuration for the rest of the switch. Even a `write erase` followed by a `reload` will not clear the VLAN database on these switches. Configuring through the VLAN database is a throwback to older models that offered no other way to manage VLANs. Luckily, all newer switches (including those with a VLAN database) offer the option of configuring the VLANs through the normal IOS CLI. Switches like the 6500, when running in native IOS mode, only support IOS commands for switch configuration. The Nexus line does not have a configurable VLAN database.

Cisco recommends that you configure the VLAN Trunking Protocol (VTP) as a first step when configuring VLANs. This idea has merit, as trunks will not negotiate without a VTP domain. However, setting a VTP domain is not required to make VLANs function on a single switch. Configuring VTP is covered later (see Chapter 5, *Trunking* and Chapter 6, *VLAN Trunking Protocol*).

CatOS

For CatOS, create a VLAN with the `set vlan` command:

```
Switch1-CatOS# (enable)set vlan 10 name Lab-VLAN
VTP advertisements transmitting temporarily stopped,
and will resume after the command finishes.
Vlan 10 configuration successful
```

There are a lot of options when creating a VLAN, but for the bare minimum, this is all you need. To show the status of the VLANs, execute the `show vlan` command:

```
Switch1-CatOS# (enable)sho vlan
VLAN Name                             Status    IfIndex Mod/Ports, Vlans
---- -------------------------------- --------- ------- ------------------------
1    default                          active    7       1/1-2
                                                        2/1-2
                                                        3/5-48
                                                        6/1-48
10   Lab-VLAN                         active    112
20   VLAN0020                         active    210     3/1-4
1002 fddi-default                     active    8
1003 token-ring-default               active    11
1004 fddinet-default                  active    9
1005 trnet-default                    active    10
1006 Online Diagnostic Vlan1          active    0       internal
1007 Online Diagnostic Vlan2          active    0       internal
1008 Online Diagnostic Vlan3          active    0       internal
1009 Voice Internal Vlan              active    0       internal
1010 Dtp Vlan                         active    0       internal
1011 Private Vlan Reserved Vlan       suspend   0       internal
1016 Online SP-RP Ping Vlan           active    0       internal
```

Notice that VLAN 10 has the name you assigned; VLAN 20's name, which you did not assign, defaulted to VLAN0020. The output shows which ports are assigned to VLAN 20 and that most of the ports still reside in VLAN 1 (because VLAN 1 is the default VLAN, all ports reside there by default).

There are no ports in VLAN 10 yet, so add some, again using the `set vlan` command:

```
Switch1-CatOS# (enable)set vlan 10 6/1,6/3-4
VLAN 10 modified.VLAN 1 modified.
VLAN  Mod/Ports
---- ------------------------
10  6/1,6/3-4
```

You've now added ports 6/1, 6/3, and 6/4 to VLAN 10. Another `show vlan` will reflect these changes:

```
Switch1-CatOS# (enable)sho vlan
VLAN Name                             Status    IfIndex Mod/Ports, Vlans
---- -------------------------------- --------- ------- ------------------------
1    default                          active    7       1/1-2
                                                        2/1-2
                                                        3/5-48
                                                        6/2,6/5-48
10   Lab-VLAN                         active    112     6/1,6/3-4
20   VLAN0020                         active    210     3/1-4
1002 fddi-default                     active    8
1003 token-ring-default               active    11
1004 fddinet-default                  active    9
1005 trnet-default                    active    10
1006 Online Diagnostic Vlan1          active    0       internal
1007 Online Diagnostic Vlan2          active    0       internal
1008 Online Diagnostic Vlan3          active    0       internal
```

```
1009 Voice Internal Vlan              active    0       internal
1010 Dtp Vlan                         active    0       internal
1011 Private Vlan Reserved Vlan       suspend   0       internal
1016 Online SP-RP Ping Vlan           active    0       internal
```

The output indicates that VLAN 1 was modified as well. This is because the ports had to be removed from VLAN 1 to be added to VLAN 10.

IOS Using VLAN Database

This method is included for the sake of completeness. Older switches that require this method of configuration are no doubt still deployed. IOS switches that support the VLAN database, such as the 3750, actually display this message when you enter VLAN database configuration mode:

```
3750-IOS#vlan database
% Warning: It is recommended to configure VLAN from config mode,
  as VLAN database mode is being deprecated. Please consult user
  documentation for configuring VTP/VLAN in config mode.
```

If you have an IOS switch with active VLANs, but no reference is made to them in the running configuration, it's possible they were configured in the VLAN database. Another possibility is that they were learned via VTP (we will cover this in Chapter 6). On 3750s, when the switch is in VTP server mode, even when you configure VLANs in CLI, they do not appear in the running configuration.

Since you're more likely to see the VLAN database in older switches, I'll continue with examples from a 2950, though they all behave pretty similarly. If you find any switch configured using the VLAN database, my advice is to convert it to an IOS configuration.

To configure VLANs in the VLAN database, you must enter VLAN database configuration mode with the command `vlan database`. Requesting help (`?`) lists the commands available in this mode:

```
2950-IOS#vlan database
2950-IOS(vlan)# ?
VLAN database editing buffer manipulation commands:
  abort  Exit mode without applying the changes
  apply  Apply current changes and bump revision number
  exit   Apply changes, bump revision number, and exit mode
  no     Negate a command or set its defaults
  reset  Abandon current changes and reread current database
  show   Show database information
  vlan   Add, delete, or modify values associated with a single VLAN
  vtp    Perform VTP administrative functions.
```

To create a VLAN, give the `vlan` command followed by the VLAN number and name:

```
2950-IOS(vlan)#vlan 10 name Lab-VLAN
VLAN 10 added:
    Name: Lab-VLAN
```

You can show the VLANs configured from within VLAN database mode with the command `show`. You have the option of displaying the current database (`show current`), the differences between the current and proposed database (`show changes`), or the proposed database as it will look after you apply the changes using the `apply` command or exit VLAN database configuration mode. The default behavior of the `show` command is `show proposed`:

```
2950-IOS(vlan)#show
  VLAN ISL Id: 1
    Name: default
    Media Type: Ethernet
    VLAN 802.10 Id: 100001
    State: Operational
    MTU: 1500
    Backup CRF Mode: Disabled
    Remote SPAN VLAN: No

  VLAN ISL Id: 10
    Name: Lab-VLAN
    Media Type: Ethernet
    VLAN 802.10 Id: 100010
    State: Operational
    MTU: 1500
    Backup CRF Mode: Disabled
    Remote SPAN VLAN: No
```

Nothing else is required to create a simple VLAN. The database will be saved upon exit:

```
2950-IOS(vlan)#exit
APPLY completed.
Exiting....
```

Now, when you execute the `show vlan` command in IOS, you'll see the VLAN you've created:

```
2950-IOS#sho vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                Gi0/1, Gi0/2
10   Lab-VLAN                         active
1002 fddi-default                     active
1003 token-ring-default               active
```

```
1004 fddinet-default                 active
1005 trnet-default                   active
```

Adding ports to the VLAN is accomplished in IOS interface configuration mode, and is covered in the next section.

IOS Using Global Commands

Adding VLANs in IOS is relatively straightforward when all of the defaults are acceptable, which is usually the case. Here I'll revert to a 3750, since you're likely to encounter modern switches using this method.

First, enter configuration mode. From there, issue the `vlan` command with the identifier for the VLAN you're adding or changing. Next, specify a name for the VLAN with the `name` subcommand (as with CatOS, a default name of `VLAN`*xxxx* is used if you do not supply one):

```
3750-IOS#conf t
Enter configuration commands, one per line. End with CNTL/Z.
3750-IOS(config)# vlan 10
3750-IOS(config-vlan)# name Lab-VLAN
```

Exit configuration mode and then issue the `show vlan` command to see the VLANs present:

```
3750-IOS#sho vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Gi1/0/1, Gi1/0/2, Gi1/0/3
                                                Gi1/0/4, Gi1/0/5, Gi1/0/6
                                                Gi1/0/7, Gi1/0/8, Gi1/0/9
                                                Gi1/0/10, Gi1/0/11, Gi1/0/12
                                                Gi1/0/13, Gi1/0/14, Gi1/0/15
                                                Gi1/0/16, Gi1/0/17, Gi1/0/18
                                                Gi1/0/21, Gi1/0/22, Gi1/0/23
                                                Gi1/0/24, Gi1/0/25, Gi1/0/26
                                                Gi1/0/27, Gi1/0/28, Gi1/0/29
                                                Gi1/0/30, Gi1/0/31, Gi1/0/32
                                                Gi1/0/33, Gi1/0/34, Gi1/0/35
                                                Gi1/0/36, Gi1/0/37, Gi1/0/38
                                                Gi1/0/39, Gi1/0/40, Gi1/0/41
                                                Gi1/0/42, Gi1/0/43, Gi1/0/44
                                                Gi1/0/46, Gi1/0/49, Gi1/0/50
                                                Gi1/0/51, Gi1/0/52
10   Lab-VLAN                         active
100  VLAN0100                         active
200  VLAN0200                         active
300  VLAN0300                         active

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
```

```
1004 fddinet-default                act/unsup
1005 trnet-default                  act/unsup

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1 Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------ ------
1    enet  100001     1500  -      -      -        -    -        0      0
10   enet  100010     1500  -      -      -        -    -        0      0
100  enet  100100     1500  -      -      -        -    -        0      0
200  enet  100200     1500  -      -      -        -    -        0      0
300  enet  100300     1500  -      -      -        -    -        0      0
1002 fddi  101002     1500  -      -      -        -    -        0      0
1003 tr    101003     1500  -      -      -        -    -        0      0
1004 fdnet 101004     1500  -      -      -        ieee -        0      0
1005 trnet 101005     1500  -      -      -        ibm  -        0      0

Remote SPAN VLANs
------------------------------------------------------------------------------


Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------------
```

You assign ports to VLANs in IOS in interface configuration mode. Each interface must be configured individually with the `switchport access` command (this is in contrast to the CatOS switches, which allow you to add all the ports at once with the `set vlan` command):

```
3750-IOS(config)#int g1/0/1
3750-IOS(config-if)#switchport access vlan 10
3750-IOS(config-if)#int g1/0/2
3750-IOS(config-if)#switchport access vlan 10
```

Modern versions of IOS allow you to apply commands to multiple interfaces with the `interface range` command. Using this command, you can accomplish the same result as before while saving some precious keystrokes:

```
3750-IOS(config)#interface range g1/0/1 - 2
3750-IOS(config-if-range)#switchport access vlan 10
```

Now, when you execute the `show vlan` command, you'll see that the ports have been assigned to the proper VLAN:

```
3750-IOS#sho vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Gi1/0/3, Gi1/0/4, Gi1/0/5
                                                Gi1/0/6, Gi1/0/7, Gi1/0/8
                                                Gi1/0/9, Gi1/0/10, Gi1/0/11
                                                Gi1/0/12, Gi1/0/13, Gi1/0/14
                                                Gi1/0/15, Gi1/0/16, Gi1/0/17
                                                Gi1/0/18, Gi1/0/21, Gi1/0/22
                                                Gi1/0/23, Gi1/0/24, Gi1/0/25
                                                Gi1/0/26, Gi1/0/27, Gi1/0/28
                                                Gi1/0/29, Gi1/0/30, Gi1/0/31
```

```
                                                  Gi1/0/32, Gi1/0/33, Gi1/0/34
                                                  Gi1/0/35, Gi1/0/36, Gi1/0/37
                                                  Gi1/0/38, Gi1/0/39, Gi1/0/40
                                                  Gi1/0/41, Gi1/0/42, Gi1/0/43
                                                  Gi1/0/44, Gi1/0/46, Gi1/0/49
                                                  Gi1/0/50, Gi1/0/51, Gi1/0/52
10   Lab-VLAN                     active    Gi1/0/1, Gi1/0/2
100  VLAN0100                     active
200  VLAN0200                     active
300  VLAN0300                     active
1002 fddi-default                 act/unsup
```

Nexus and NX-OS

NX-OS uses a command interface similar to IOS. NX-OS behaves a little bit differently, especially concerning the configuration of interfaces. The methods used for configuring VLANs are very similar to IOS. First we create the VLAN with the `vlan` *`vlan-#`* command:

```
NX-7K-1-Cozy(config)# vlan 10
```

Once you've created the VLAN, enter VLAN configuration mode and name the VLAN with the `name` *`vlan-name`* command:

```
NX-7K-1-Cozy(config-vlan)# name Lab-VLAN
```

One of the cool features of NX-OS is that you no longer need the `do` command to run show commands from configuration mode. This behavior is similar to the PIX and ASA configuration mode, and is a most welcome change. Here, I've executed the `show vlan` command from within VLAN configuration mode:

```
NX-7K-1-Cozy(config-if)# sho vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Po1, Po10
10   Lab-VLAN                         active    Po1, Eth3/2

VLAN Type
---- -----
1    enet
10   enet

Remote SPAN VLANs
------------------------------------------------------------------------------

Primary  Secondary  Type             Ports
-------  ---------  ---------------  ------------------------------------------
```

Another new feature in NX-OS is the capability to configure a range of interfaces without using the `interface-range` command. Simply enter the range you want to configure as if you were using the `interface-range` command in IOS:

```
NX-7K-1-Cozy(config-vlan)# int e3/1 - 2
```

This automatically puts us into interface range configuration mode. Now we assign the ports to a VLAN the same way we would in IOS—using the `switchport access vlan` *`vlan#`* command:

```
NX-7K-1-Cozy(config-if-range)# switchport access vlan 10
Warning: command rejected, Eth3/1 not a switching port
Warning: command rejected, Eth3/2 not a switching port
```

Now there's a message you don't see on a Catalyst switch by default. The Nexus 7000 switch behaves differently than a catalyst. By default, all switch ports are router ports! To perform switch port commands on a Nexus port, you must first put them into switchport mode with the `switchport` command:

```
NX-7K-1-Cozy(config-if-range)# int e3/1 - 2
NX-7K-1-Cozy(config-if-range)# switchport
NX-7K-1-Cozy(config-if-range)# no shut
```

Now that we have placed the ports into switchport mode, we can assign them to a VLAN without further interruption:

```
NX-7K-1-Cozy(config-if-range)# switchport access vlan 10
NX-7K-1-Cozy(config-if-range)#
```

The `show vlan` command now shows our ports assigned to VLAN 10. Notice once more how I've executed a `show` command from within configuration mode. I love this feature!

```
NX-7K-1-Cozy(config-if-range)# sho vlan

VLAN Name                             Status    Ports

---- -------------------------------- --------- -------------------------------

1    default                          active
10   Lab-VLAN                         active    Eth3/1, Eth3/2


VLAN Type
---- -----
1    enet
10   enet


Remote SPAN VLANs
------------------------------------------------------------------------------


Primary  Secondary  Type             Ports
-------  ---------  ---------------  ------------------------------------------
```

CHAPTER 5
Trunking

A *trunk*, using Cisco's terminology, is an interface or link that can carry frames for multiple VLANs at once. As you saw in the previous chapter, a trunk can connect two switches so that devices in VLANs on one switch can communicate with devices in the same VLANs on another switch. Unless there is only one VLAN to be connected, switches are connected at Layer 2 using trunks. Figure 5-1 shows two switches connected with a trunk.

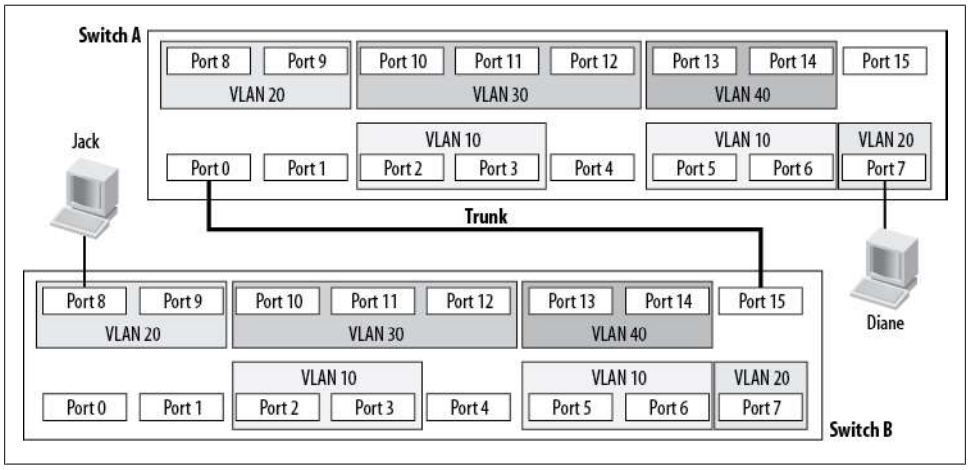

Figure 5-1. A trunk connecting two switches

Trunking is generally related to switches, but routers, firewalls, and all manner of devices can connect with trunks as well. The router-on-a-stick scenario described in Chapter 4 requires a router to communicate with a trunk port on a switch.

How Trunks Work

Figure 5-2 is a visual representation of a trunk. VLANs 10, 20, 30, and 40 exist on both sides of the trunk. Any traffic from VLAN 10 on Switch 1 that is destined for VLAN 10 on Switch 2 must traverse the trunk (of course, the reverse is true as well).

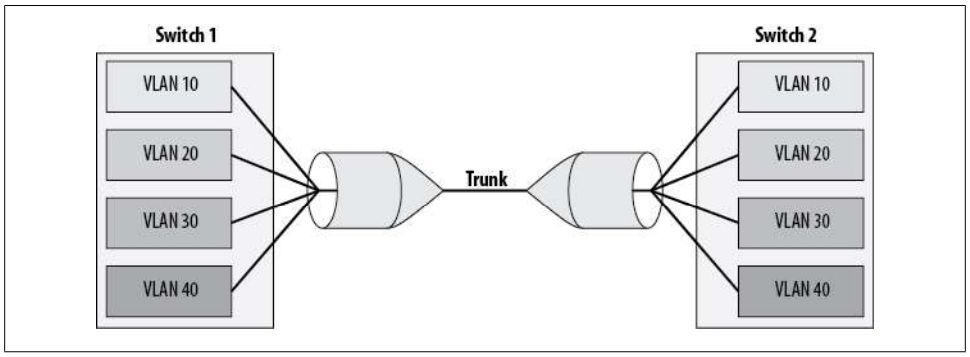

Figure 5-2. Visual representation of a trunk

For the remote switch to determine how to forward the frame, the frame must contain a reference to the VLAN to which it belongs. IP packets have no concept of VLANs, though, nor do TCP, UDP, ICMP, or any other protocol above Layer 2. Remember that a VLAN is a Layer-2 concept, so any reference to a VLAN would happen at the data-link layer. Ethernet was invented before VLANs, so there is no reference to VLANs in any Ethernet protocols, either.

To mark, or *tag*, frames to be sent over a trunk, both sides must agree to a protocol. Currently, the protocols for trunking supported on Cisco switches are Cisco's Inter-Switch Link (ISL) and the IEEE standard 802.1Q. Not all Cisco switches support both protocols. For example, the Cisco 2950 and 4000 switches only support 802.1Q. To determine whether a switch can use a specific trunking protocol, use the IOS or NX-OS command `show interface capabilities`, or the CatOS command `show port capabilities`:

```
NX-7K-1-Cozy(config)# sho interface capabilities | inc Trunk
Ethernet3/1
  Trunk encap. type:     802.1Q
```

As you can see from the preceding output, the Nexus supports only the open standard, 802.1Q. Here, you can see that a 3750 supports both 802.1Q and ISL:

```
3750-IOS#sho int g1/0/45 capabilities | inc Trunk
  Trunk encap. type:     802.1Q,ISL
  Trunk mode:            on,off,desirable,nonegotiate
```

ISL differs from 802.1Q in a couple of ways. First, ISL is a Cisco-proprietary protocol, whereas 802.1Q is an IEEE standard. Second, ISL encapsulates Ethernet frames within an ISL frame, while 802.1Q alters existing frames to include VLAN tags. Furthermore,

ISL is capable of supporting only 1,000 VLANs, while 802.1Q is capable of supporting 4,096.

On switches that support both ISL and 802.1Q, either may be used. The protocol is specific to each trunk. While both sides of the trunk must agree on a protocol, you may configure ISL and 802.1Q trunks on the same switch and in the same network.

ISL

To add VLAN information to a frame to be sent over a trunk, ISL encapsulates the entire frame within a new frame. An additional header is prepended to the frame and a small suffix is added to the end. The header contains information regarding the VLAN number and some other information, while the footer includes a checksum of the frame. A high-level overview of an ISL frame is shown in Figure 5-3.

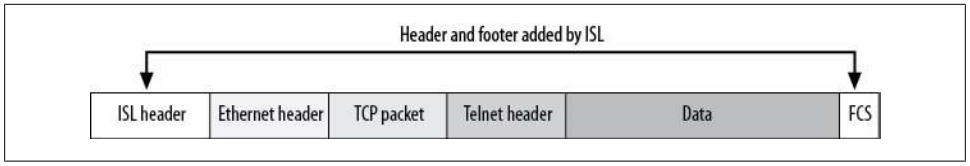

Figure 5-3. ISL encapsulated frame

The frame check sequence (FCS) footer is in addition to the FCS field already present in Ethernet frames. The ISL FCS frame computes a checksum based on the frame including the ISL header; the Ethernet FCS checksum does not include this header.

Adding more information to an Ethernet frame can be problematic. If an Ethernet frame has been created at the maximum size of 1,518 bytes, ISL will add an extra 30 bytes, for a total frame size of 1,548 bytes. These frames may be counted as "giant" frame errors, though Cisco equipment has no problem accepting them.

802.1Q

802.1Q takes a different approach to VLAN tagging. Instead of adding more headers to a frame, 802.1Q inserts data into existing headers. An additional four-byte tag field is inserted between the Source Address and Type/Length fields. Because 802.1Q has altered the frame, the FCS of the frame is altered to reflect the change.

Because only four bytes are added, the maximum size for an 802.1Q frame is 1,522 bytes. This may result in "baby giant" frame errors, though the frames will still be supported on Cisco devices.

Which Protocol to Use

Why are there two protocols at all? There is a history at work here. Cisco developed and implemented ISL before 802.1Q existed. Older switches from Cisco only support ISL. Oddly enough, other switches, like the Nexus, only support 802.1Q. In some cases, the blade within a switch chassis may be the deciding factor. As an example, some 10 Gb blades available for the Catalyst 6509 only support 802.1Q, while the switch itself supports 802.1Q and ISL; in this case, 802.1Q must be used.

In many installations, you can use either protocol, and the choice is not important. When you're trunking between Cisco switches, there is no real benefit of using one protocol over the other, except for the fact that 802.1Q can support 4,096 VLANs, whereas ISL can only support 1,000. Some purists may argue that ISL is better because it doesn't alter the original frame, and some others may argue that 802.1Q is better because the frames are smaller and there is no encapsulation. What usually ends up happening is that whoever installs the trunk uses whichever protocol he is used to.

Cisco has recommendations on how to set up trunks between Cisco switches. This document (Cisco document ID 24067) is titled "System Requirements to Implement Trunking."

When you're connecting Cisco switches to non-Cisco devices, the choice is 802.1Q. Remember, there are no restrictions regarding protocol choice on a switch that supports both. If you need to connect a 3Com switch to your Cisco network, you can do so with an 802.1Q trunk even if your Cisco network uses ISL trunks elsewhere. The trunking protocol is local to each individual trunk. If you connect to a 3Com switch using 802.1Q, the VLANs on that switch will still be accessible on switches connected using ISL elsewhere.

The Cisco Nexus line, which appears to embrace open standards more than previous devices, does not support ISL.

Trunk Negotiation

Some Cisco switches support the Dynamic Trunking Protocol (DTP). This protocol attempts to determine which trunking protocols are supported on each side and to establish a trunk, if possible.

Trunk negotiation includes the VLAN Trunking Protocol (VTP) domain name in the process. For DTP to successfully negotiate, both switches must have the same VTP domain name. See Chapter 6 for details on configuring VTP domains.

An interface running DTP sends frames every 30 seconds in an attempt to negotiate a trunk. If a port has been manually set to either "trunk" or "prevent trunking," DTP is unnecessary and can be disabled. The IOS command `switchport nonegotiate` disables DTP:

```
3750-IOS(config-if)#switchport mode trunk
3750-IOS(config-if)#switchport nonegotiate
```

Many networking engineers prefer to issue the `switchport nonegotiate` command to all ports as a security precaution. If someone were to attach a rogue switch to the network and manage to negotiate a trunk, she would potentially have access to every VLAN on the switch. I prefer to disable all ports by default so that any attached device requires human intervention to gain access to the network. Both approaches have merit, and they can even be used together.

Figure 5-4 shows the possible `switchport` modes. Remember, not all switches support all modes. A port can be set to the mode `access`, which means it will never be a trunk; `dynamic`, which means the port may become a trunk; or `trunk`, which means the port will be a trunk regardless of any other settings.

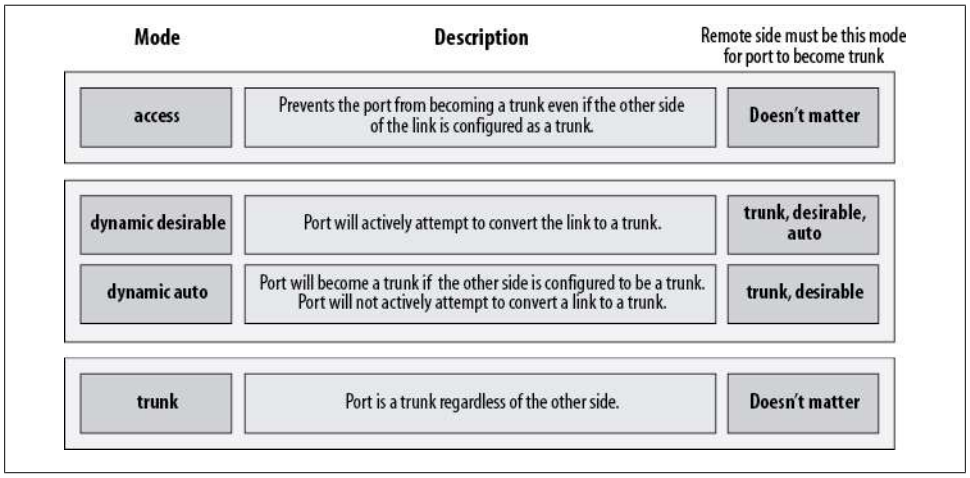

Figure 5-4. Possible switch port modes related to trunking

The two `dynamic` modes, `desirable` and `auto`, refer to the method in which DTP will operate on the port. `desirable` indicates that the port will initiate negotiations and try to make the link a trunk. `auto` indicates that the port will listen for DTP but will not actively attempt to become a trunk.

The default mode for most Cisco switches is `dynamic auto`. A port in this condition will automatically become a trunk should the remote switch port connecting to it be hard-coded as a trunk or set to `dynamic desirable`.

Configuring Trunks

Configuring a trunk involves determining which port will be a trunk, which protocol the trunk will run, and whether and how the port will negotiate. Optionally, you may also wish to limit which VLANs are allowed on the trunk link.

IOS

The Cisco 3750 is an excellent example of an IOS switch. This section will walk you through configuring one of the Gigabit ports to be an 802.1Q trunk using a 3750 switch.

You might think the first step is to specify that the port is a trunk, but as you're about to see, that's not the case:

```
3750-IOS(config-if)#switchport mode trunk
Command rejected: An interface whose trunk encapsulation is "Auto" can not be
configured to "trunk" mode.
```

On an IOS switch capable of both ISL and 802.1Q, you must specify a *trunk encapsulation* before you can configure a port as a trunk (`trunk encapsulation` is an unfortunate choice for the command because, as you now know, 802.1Q does not encapsulate frames like ISL does; still, you must follow Cisco's syntax). Once you've chosen a trunking protocol, you are free to declare the port a trunk:

```
3750-IOS(config)#int g1/0/45
3750-IOS(config-if)#switchport trunk encapsulation dot1q
3750-IOS(config-if)#switchport mode trunk
```

Should you wish to subsequently remove trunking from the interface, you'd use the command `switchport mode access`.

By default, all VLANs on a switch are included in a trunk. But you may have 40 VLANs and only need to trunk 3 of them. Because broadcasts from all allowed VLANs will be sent on every trunk port, excluding unneeded VLANs can save a lot of bandwidth on your trunk links. You can specify which VLANs are able to traverse a trunk with the `switchport trunk allowed` command. These are the options for this command:

```
3750-IOS(config-if)#switchport trunk allowed vlan ?
  WORD    VLAN IDs of the allowed VLANs when this port is in trunking mode
  add     add VLANs to the current list
  all     all VLANs
  except  all VLANs except the following
```

```
  none    no VLANs
  remove  remove VLANs from the current list
```

To allow only one VLAN (VLAN 100, in this case) on a trunk, use a command like this:

```
3750-IOS(config-if)#switchport trunk allowed vlan 100
```

As you can see from the output of the `show interface trunk` command, only VLAN 100 is now allowed. IOS has removed the others:

```
3750-IOS#sho int trunk

Port        Mode             Encapsulation  Status        Native vlan
Gi1/0/45    on               802.1q         trunking      1

Port        Vlans allowed on trunk
Gi1/0/45    100

Port        Vlans allowed and active in management domain
Gi1/0/45    100

Port        Vlans in spanning tree forwarding state and not pruned
Gi1/0/45    100
```

If you want to allow all VLANs except VLAN 100, do it with the following command:

```
3750-IOS(config-if)#switchport trunk allowed vlan except 100
```

This command will override the previous command specifying VLAN 100 as the only allowed VLAN, so now all VLANs *except* VLAN 100 will be allowed. Reexecuting the `switchport trunk allowed vlan 100` command would again reverse the state of the VLANs allowed on the trunk. `show interface trunk` shows the status:

```
3750-IOS#sho int trunk

Port        Mode             Encapsulation  Status        Native vlan
Gi1/0/45    on               802.1q         trunking      1

Port        Vlans allowed on trunk
Gi1/0/45    1-99,101-4094

Port        Vlans allowed and active in management domain
Gi1/0/45    1

Port        Vlans in spanning tree forwarding state and not pruned
Gi1/0/45    none
```

VLANs 1-99 and 101-4096 are now allowed on the trunk. Let's say you want to remove VLANs 200 and 300 as well. Using the `remove` keyword, you can do just that:

```
3750-IOS(config-if)#switchport trunk allowed vlan remove 200
3750-IOS(config-if)#switchport trunk allowed vlan remove 300
```

The output of `show interface trunk` now shows that all VLANs—except 100, 200, and 300—are allowed on the trunk:

```
3750-IOS#sho int trunk

Port        Mode             Encapsulation  Status        Native vlan
Gi1/0/45    on               802.1q         trunking      1

Port        Vlans allowed on trunk
Gi1/0/45    1-99,101-199,201-299,301-4094

Port        Vlans allowed and active in management domain
Gi1/0/45    1

Port        Vlans in spanning tree forwarding state and not pruned
Gi1/0/45    1
```

CatOS

You configure a trunk on a CatOS switch via the `set trunk` command. Options for the `set trunk` command are as follows:

```
Switch1-CatOS# (enable)set trunk 3/1 ?
  none                     No vlans
  <mode>                   Trunk mode (on,off,desirable,auto,nonegotiate)
  <type>                   Trunk type (isl,dot1q,dot10,lane,negotiate)
  <vlan>                   VLAN number
```

The mode `on` indicates that the port has been hardcoded to be a trunk, and the mode `off` indicates that the port will never be a trunk. The modes `desirable` and `auto` are both dynamic and refer to the method with which DTP will operate on the port. `desirable` indicates that the port will initiate negotiations and try to make the link a trunk. `auto` indicates that the port will listen for DTP, but will not actively attempt to become a port. You can use the mode `nonegotiate` to turn off DTP in the event that either `on` or `off` has been chosen as the mode on the opposing port.

The trunk types `isl` and `dotq1` specify ISL and 802.1Q as the protocols, respectively; `negotiate` indicates that DTP should be used to determine the protocol. The trunk types `dot10` and `lane` are for technologies such as ATM, and will not be covered here.

One of the nice features of CatOS is that it allows you to stack multiple arguments in a single command. This command sets the port to mode `desirable` and the protocol to 802.1Q:

```
Switch1-CatOS# (enable)set trunk 3/5 desirable dot1q
Port(s)  3/1 trunk mode set to desirable.
Port(s)  3/1 trunk type set to dot1q.
Switch1-CatOS# (enable)
2006 May 23 11:29:31 %ETHC-5-PORTFROMSTP:Port 3/5 left bridge port 3/5
2006 May 23 11:29:34 %DTP-5-TRUNKPORTON:Port 3/5 has become dot1q trunk
```

The other side of the link was not configured, but the trunk became active because the default state of the ports on the other side is `auto`.

The command to view trunk status on CatOS is `show port trunk`:

```
Switch1-CatOS#sho port trunk
* - indicates vtp domain mismatch
# - indicates dot1q-all-tagged enabled on the port
$ - indicates non-default dot1q-ethertype value
Port      Mode         Encapsulation  Status        Native vlan
--------  -----------  -------------  ------------  -----------
 3/5      desirable    dot1q          trunking      1
15/1      nonegotiate  isl            trunking      1
16/1      nonegotiate  isl            trunking      1

Port      Vlans allowed on trunk
--------  ---------------------------------------------------------------------
 3/5      1-4094
15/1      1-4094
16/1      1-4094

Port      Vlans allowed and active in management domain
--------  ---------------------------------------------------------------------
 3/5      1,10,20
15/1
16/1

Port      Vlans in spanning tree forwarding state and not pruned
--------  ---------------------------------------------------------------------
 3/5      1,10,20
15/1
16/1
```

The trunks 15/1 and 16/1 shown in this output are internal trunks. On a 6500 switch running CatOS, trunks exist from the supervisors to the multilayer switch feature cards (MSFCs). The MSFCs are known as slots 15 and 16 when two supervisors are installed.

To specify which VLANs can traverse a trunk, use the same `set trunk` command and append the VLANs you wish to allow. CatOS works a little differently from IOS in that it will not remove all of the active VLANs in favor of ones you specify:

```
Switch-2# (enable)set trunk 3/5 100
Vlan(s) 100 already allowed on the trunk
Please use the 'clear trunk' command to remove vlans from allowed list.
```

Remember that all VLANs are allowed by default. Preventing a single VLAN from using a trunk is as simple as using the `clear trunk` command:

```
Switch-2# (enable)clear trunk 3/5 100
Removing Vlan(s) 100 from allowed list.
Port  3/5 allowed vlans modified to 1-99,101-4094.
```

You don't have to use a `show trunk` command to see which VLANs are allowed, because the `clear trunk` tells you the new status of the port.

To limit a CatOS switch so that only one VLAN is allowed, disallow all the remaining VLANs. Just as you removed one VLAN with the `clear trunk` command, you can remove all of the VLANs *except* the one you want to allow:

```
Switch-2# (enable)clear trunk 3/5 1-99,101-4094
Removing Vlan(s) 1-99,101-4094 from allowed list.
Port  3/5 allowed vlans modified to 100.
```

Finally, a `show trunk` will show you the status of the trunks. As you can see, only VLAN 100 is now allowed on trunk 3/5:

```
Switch-2# (enable)sho trunk
* - indicates vtp domain mismatch
# - indicates dot1q-all-tagged enabled on the port
$ - indicates non-default dot1q-ethertype value
Port      Mode         Encapsulation  Status        Native vlan
--------  -----------  -------------  ------------  -----------
 3/5      auto         dot1q          trunking      1
15/1      nonegotiate  isl            trunking      1
16/1      nonegotiate  isl            trunking      1

Port      Vlans allowed on trunk
--------  ---------------------------------------------------------------------
 3/5      100
15/1      1-4094
16/1      1-4094

Port      Vlans allowed and active in management domain
--------  ---------------------------------------------------------------------
 3/5
15/1
16/1

Port      Vlans in spanning tree forwarding state and not pruned
--------  ---------------------------------------------------------------------
 3/5
15/1
16/1
```

Nexus and NX-OS

Remembering that NX-OS on a Nexus 7000 places interfaces in routed mode by default, we must first issue the `switchport` interface command:

```
NX-7K-1-Cozy(config-if)# int e3/2
NX-7K-1-Cozy(config-if)# switchport
```

Now we put the port into `trunk` mode with the `switchport mode trunk` interface command:

```
NX-7K-1-Cozy(config-if)# switchport mode ?
  access  Port mode access
  trunk   Port mode trunk

NX-7K-1-Cozy(config-if)# int e3/2
NX-7K-1-Cozy(config-if)# switchport mode trunk
```

We limit the VLANs allowed over the trunk with the `switchport trunk allowed vlan` interface command:

```
NX-7K-1-Cozy(config-if)# switchport trunk allowed vlan ?
  <1-3967,4048-4093>  VLAN IDs of the allowed VLANs when this port in trunking
                      mode
  add                 Add VLANs to the current list
  all                 All VLANs
  except              All VLANs except the following
  none                No VLANs
  remove              Remove VLANs from the current list
```

This works similarly to the IOS version of the command. Here, if we allow VLAN 100, all other VLANs are prohibited.

```
NX-7K-1-Cozy(config-if)# switchport trunk allowed vlan 100
```

You can see which VLANs are allowed over the trunk with the `show interface trunk` command. The output of the command differs from IOS, but the information is all there.

```
NX-7K-1-Cozy(config-if)# sho int trunk

--------------------------------------------------------------------------------
Port       Native  Status        Port
           Vlan                  Channel
--------------------------------------------------------------------------------
Eth3/2     1       trunking      --
Eth8/1     1       trnk-bndl     Po1
Eth8/2     1       trnk-bndl     Po1
Po1        1       trunking      --

--------------------------------------------------------------------------------
Port       Vlans Allowed on Trunk
--------------------------------------------------------------------------------
Eth3/2     100
Eth8/1     1-3967,4048-4093
Eth8/2     1-3967,4048-4093
Po1        1-3967,4048-4093

--------------------------------------------------------------------------------
Port       Vlans Err-disabled on Trunk
--------------------------------------------------------------------------------
Eth3/2     none
Eth8/1     none
Eth8/2     none
Po1        none

--------------------------------------------------------------------------------
Port       STP Forwarding
--------------------------------------------------------------------------------
Eth3/2     none
Eth8/1     none
Eth8/2     none
Po1        1,10
```