Chapter 2

Layer 2 Support and Configurations

The Nexus line of switches provides a robust Layer 2 feature set. This chapter covers common implementations and syntax for Layer 2 features such as virtual local-area networks (VLANS), Private VLANs (PVLANS), Spanning Tree Protocol (STP), Unidirectional Link Detection (UDLD), and Virtual Port Channel (vPC).

This chapter covers the following topics, as they relate to the Nexus family of switches:

- **Layer 2 overview:** Describes the functionality of layer 2 features and interfaces for the Nexus family of switches
- **VLANs and Private VLANs:** Describes VLAN and Private VLAN support available within the Nexus family of switches
- **Spanning Tree Protocol:** Outlines the different STP options available within the Nexus switches and the configuration parameters
- **Virtual Port Channel:** Describes the functionality of configuring Virtual Port Channels between a pair of Nexus switches and provides configuration examples and best practices
- **Unidirectional Link Detection:** Describes how to use Unidirectional Link Detection to prevent undesirable conditions in a Layer 2 environment

Layer 2 Overview

Although NX-OS is a single operating system for the Nexus line of switches, the hardware architecture of the switches might differ slightly. This section begins by reviewing some basic switching concepts and then discuss the forwarding behavior of both the Nexus 5000 and Nexus 7000 switches.

Layer 2 forwarding deals with the capability to build and maintain Media Access Control (MAC) address tables that are stored in a Content Addressable Memory (CAM) table. MAC tables are built by learning the MAC address of the stations that are plugged into

them. The process of learning MAC addresses is done in large part dynamically; however, in certain cases, it might be necessary for a network administrator to create static MAC entries that are prepopulated into the CAM table. When populated, the CAM tables are used to make forwarding decisions at Layer 2 by analyzing the destination MAC (DMAC) address of the frame. When this table lookup occurs and any other decisions such as dropping the frame or flooding the frame, it determines whether the switch is said to implement a store-and-forward or cut-through method.

Store-and-Forward Switching

Store-and-forward switching waits until the entire frame has been received and then compares the last portion of the frame against the frame check sequence (FCS) to ensure that no errors have been introduced during physical transmission. If the frame is determined to be corrupt, it is immediately dropped. Store-and-forward switching also inherently addresses any issues that might arise when a packet's ingress and egress ports have dissimilar underlying physical characteristics, that is, 100 Mbps versus 1 Gbps versus 10 Gbps. Latency measurements in a store-and-forward switch are typically measured on a Last In First Out (LIFO) basis. The Nexus 7000 series of switches implements store-and-forward switching.

Cut-Through Switching

Although store-and-forward switches wait for the entire frame to be received into a buffer, cut-through switches can perform the L2 lookup as soon as the DMAC is received in the first 6 bytes of the frame. Historically, cut-through switching provided a method for forwarding frames at high speeds while relying on another station to discard invalid frames. The latency of cut-through switching platforms is typically measured on a First In First Out (FIFO) basis. As application-specific integrated circuit (ASIC) process technology matured, cut-through switches gained the capability to look further into the frame without the performance penalty associated with store-and-forward switches. Additionally, over time, physical mediums have become more reliable than in the past. With the maturity of both ASIC process technology and physical transmission reliability, the industry has seen a reemergence of cut-through switching technology. The Nexus 5000 series of switches uses the cut-through switching method, except in the case in which there is dissimilar transmission speeds between the ingress and egress ports.

Fabric Extension via the Nexus 2000

The Nexus 5000 offers a unique capability by combining with the Nexus 2000 Fabric Extenders. The Nexus 2000 operates as a line card for the Nexus 5000, without being constrained to a physical chassis as is the case with most modular switch platforms. To

continue this analogy, the Nexus 5000 operates as a supervisor module for the *virtual chassis*. Although physically separate, the Nexus 5000 and 2000 are managed as a single entity from a software image, configuration, and spanning-tree perspective. This functionality enables data center engineers to gain the cabling benefits of an in-rack switching solution, with the simplicity of management of a centralized or end-of-row topology. The first product within the Nexus 2000 family is the 2148T that supports 48 fixed ports of Gigabit Ethernet, and four 10 Gigabit interfaces to connect to the Nexus 5000. You need to understand that the Nexus 2000 does not perform any local switching functions, and all traffic is switched in a centralized fashion by the Nexus 5000. The front panel ports on a Nexus 2000 do not operate the same way that a normal switch port would and should only be used for host connectivity. One of the most apparent differences in operations between the Nexus 2000 and other switches is the implementation of BPDUGuard on all front-panel ports. BPDUGuard is covered in depth later in this chapter.

The initial configuration of the Nexus 2000 is quite simple and when configured can then be treated as additional ports configurable on the Nexus 5000. The Nexus 2000 can be connected to the 5000 in one of two distinct modes:

- **Static pinning:** This configuration creates a direct relationship between front panel ports and their uplinks. The pinning is based on the number of uplinks available. For example, if one uplink is active, all front panel ports are mapped. Static pinning is a good option in which tight control over the bandwidth and oversubscription is desirable. The drawback of static pinning is that if uplinks are added or removed, the host ports are bounced to repin the hosts across the uplinks. One to four uplinks are supported in this configuration.
- **Etherchannel:** This configuration aggregates the uplink ports into a single logical interface that all front-panel ports are mapped to. As discussed later in the chapter, in an Etherchannel configuration only 1, 2, or 4 uplinks should be used. In this configuration hosts' ports remain up if uplinks are added or removed. The uplink port-channel can also be a VPC to two different Nexus 5000s.

Virtual Port channels can be used to connect Nexus 2000s to Nexus 5000s or to connect hosts to Nexus 2000; however, these two connectivity scenarios cannot be used together. Figure 2-1 illustrates the supported VPC topologies.

Configuring Nexus 2000 Using Static Pinning

This section demonstrates a basic Nexus 2000 configuration using the topology shown in Figure 2-2.

Example 2-1 demonstrates how to configure Nexus 2000 to 5000 connectivity using two uplinks in static pinning mode.

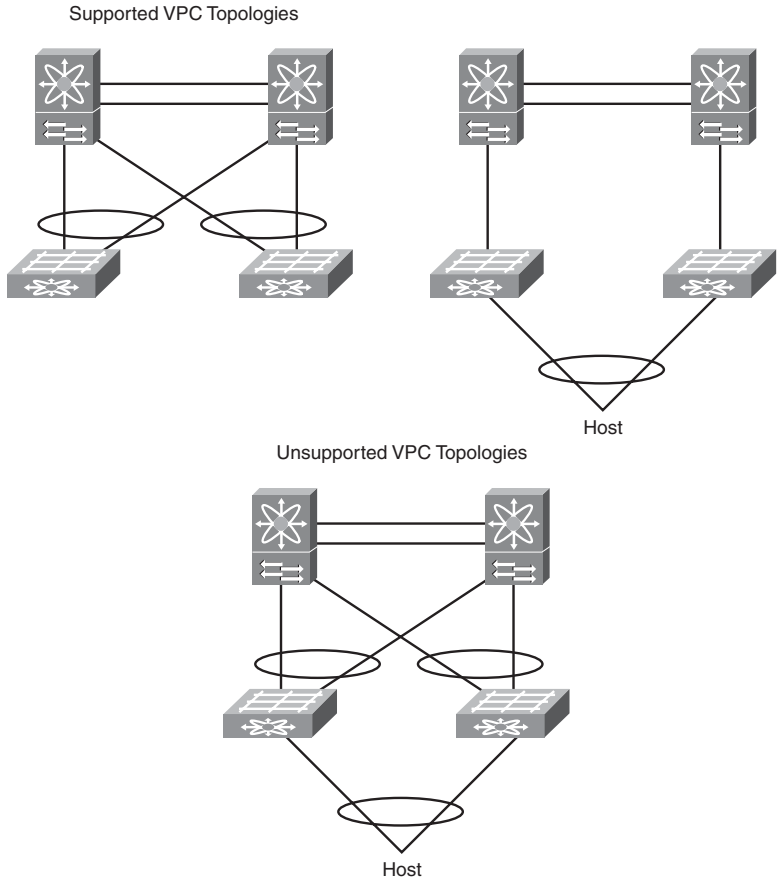

Figure 2-1 *Nexus 2000 Supported VPC Topologies*

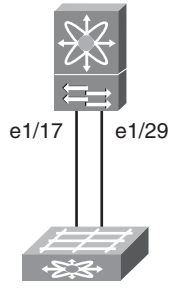

Figure 2-2 *Network Topology for Nexus 2000 Configuration*

Example 2-1 *Nexus 2000 Static Pinning Configuration*

```
NX5000(config)# fex 100
NX5000(config-fex)# pinning max-links 2
Change in Max-links will cause traffic disruption.
NX5000(config-fex)# description FEX100
NX5000(config-fex)# exit
NX5000(config)#


NX5000# conf t
NX5000(config)# interface ethernet 1/17
NX5000(config-if)# switchport mode fex-fabric
NX5000(config-if)# fex associate 100
NX5000(config-if)# no shutdown
NX5000(config-if)# exit
NX5000(config)# interface ethernet 1/29
NX5000(config-if)# switchport mode fex-fabric
NX5000(config-if)# fex associate 100
NX5000(config-if)# no shutdown
NX5000(config-if)# exit
NX5000(config)# exit
```

Nexus 2000 Static Pinning Verification

You can monitor the Nexus 2000 using the following commands:

- **show fex:** Displays a list of fabric extension (FEX) units, their description, state, model, and serial number.
- **show fex *fex-id* detail:** Provides verbose status information about a particular Nexus 2000. The output of this command provides details as to the software versions, operational configuration, uplink status, and much more.
- **show interface status fex *fex-id*:** Displays the status of front-panel host ports on a particular FEX.
- **show interface ethernet *mod/slot* fex-intf:** Displays front-panel hosts' ports pinned to a particular Nexus 5000 interface.

Example 2-2 shows sample output from the previous commands.

Example 2-2 *Nexus 2000 Static Pinning Verification*

```
NX5000# show fex
  FEX             FEX             FEX                  FEX
Number       Description         State      Model            Serial
------------------------------------------------------------------------
  100            FEX100        Online      N2K-C2148T-1GE   JAF1318AALS
```

```
NX5000# show fex 100 detail
FEX: 100 Description: FEX100   state: Online
  FEX version: 4.0(1a)N2(1a) [Switch version: 4.0(1a)N2(1a)]
  FEX Interim version: 4.0(1a)N2(1a)
  Switch Interim version: 4.0(1a)N2(1a)
  Extender Model: N2K-C2148T-1GE, Extender Serial: JAF1318AALS
  Part No: 73-12009-05
  Card Id: 70, Mac Addr: 00:0d:ec:cd:26:c2, Num Macs: 64
  Module Sw Gen: 19 [Switch Sw Gen: 19]
 pinning-mode: static    Max-links: 2
  Fabric port for control traffic: Eth1/17
  Fabric interface state:
    Eth1/17 - Interface Up. State: Active
    Eth1/29 - Interface Up. State: Active
  Fex Port        State  Fabric Port  Primary Fabric
       Eth100/1/1    Up     Eth1/17     Eth1/17
       Eth100/1/2    Up     Eth1/17     Eth1/17
       Eth100/1/3    Up     Eth1/17     Eth1/17
       Eth100/1/4  Down     Eth1/17     Eth1/17
       Eth100/1/5  Down     Eth1/17     Eth1/17
       Eth100/1/6  Down     Eth1/17     Eth1/17
       Eth100/1/7  Down     Eth1/17     Eth1/17
       Eth100/1/8  Down     Eth1/17     Eth1/17
       Eth100/1/9  Down     Eth1/17     Eth1/17
      Eth100/1/10    Up     Eth1/17     Eth1/17
      Eth100/1/11    Up     Eth1/17     Eth1/17
      Eth100/1/12    Up     Eth1/17     Eth1/17
      Eth100/1/13    Up     Eth1/17     Eth1/17
      Eth100/1/14  Down     Eth1/17     Eth1/17
      Eth100/1/15  Down     Eth1/17     Eth1/17
      Eth100/1/16  Down     Eth1/17     Eth1/17
      Eth100/1/17  Down     Eth1/17     Eth1/17
      Eth100/1/18  Down     Eth1/17     Eth1/17
      Eth100/1/19  Down     Eth1/17     Eth1/17
      Eth100/1/20  Down     Eth1/17     Eth1/17
      Eth100/1/21  Down     Eth1/17     Eth1/17
      Eth100/1/22  Down     Eth1/17     Eth1/17
      Eth100/1/23  Down     Eth1/17     Eth1/17
      Eth100/1/24  Down     Eth1/17     Eth1/17
      Eth100/1/25  Down     Eth1/29     Eth1/29
      Eth100/1/26  Down     Eth1/29     Eth1/29
      Eth100/1/27  Down     Eth1/29     Eth1/29
      Eth100/1/28  Down     Eth1/29     Eth1/29
      Eth100/1/29  Down     Eth1/29     Eth1/29
```

```
      Eth100/1/30  Down     Eth1/29     Eth1/29
      Eth100/1/31  Down     Eth1/29     Eth1/29
      Eth100/1/32  Down     Eth1/29     Eth1/29
      Eth100/1/33  Down     Eth1/29     Eth1/29
      Eth100/1/34  Down     Eth1/29     Eth1/29
      Eth100/1/35  Down     Eth1/29     Eth1/29
      Eth100/1/36  Down     Eth1/29     Eth1/29
      Eth100/1/37  Down     Eth1/29     Eth1/29
      Eth100/1/38  Down     Eth1/29     Eth1/29
      Eth100/1/39  Down     Eth1/29     Eth1/29
      Eth100/1/40  Down     Eth1/29     Eth1/29
      Eth100/1/41  Down     Eth1/29     Eth1/29
      Eth100/1/42  Down     Eth1/29     Eth1/29
      Eth100/1/43  Down     Eth1/29     Eth1/29
      Eth100/1/44    Up     Eth1/29     Eth1/29
      Eth100/1/45    Up     Eth1/29     Eth1/29
      Eth100/1/46    Up     Eth1/29     Eth1/29
      Eth100/1/47    Up     Eth1/29     Eth1/29
      Eth100/1/48    Up     Eth1/29     Eth1/29
Logs:
[02/08/2010 18:26:44.953152] Module register received
[02/08/2010 18:26:44.954622] Registration response sent
[02/08/2010 18:26:44.989224] Module Online Sequence
[02/08/2010 18:26:46.868753] Module Online
NX5000# sho interface status fex 100
--------------------------------------------------------------------------------
Port           Name               Status   Vlan      Duplex  Speed   Type
--------------------------------------------------------------------------------
Eth100/1/1     --                 up       trunk     full    1000    --
Eth100/1/2     --                 up       trunk     full    1000    --
Eth100/1/3     --                 up       trunk     full    1000    --
Eth100/1/4     --                 down     1         full    1000    --
Eth100/1/5     --                 down     1         full    1000    --
Eth100/1/6     --                 down     1         full    1000    --
Eth100/1/7     --                 down     1         full    1000    --
Eth100/1/8     --                 down     1         full    1000    --
Eth100/1/9     --                 down     1         full    1000    --
Eth100/1/10    --                 up       89        full    1000    --
Eth100/1/11    --                 up       trunk     full    1000    --
Eth100/1/12    --                 up       trunk     full    1000    --
Eth100/1/13    --                 up       trunk     full    1000    --
Eth100/1/14    --                 down     1         full    1000    --
Eth100/1/15    --                 down     1         full    1000    --
Eth100/1/16    --                 down     1         full    1000    --
```

```
Eth100/1/17    --                 down     1         full    1000    --
Eth100/1/18    --                 down     1         full    1000    --
Eth100/1/19    --                 down     1         full    1000    --
Eth100/1/20    --                 down     1         full    1000    --
Eth100/1/21    --                 down     1         full    1000    --
Eth100/1/22    --                 down     1         full    1000    --
Eth100/1/23    --                 down     89        full    1000    --
Eth100/1/24    --                 down     89        full    1000    --
Eth100/1/25    --                 down     100       full    1000    --
Eth100/1/26    --                 down     100       full    1000    --
Eth100/1/27    --                 down     1         full    1000    --
Eth100/1/28    --                 down     1         full    1000    --
Eth100/1/29    --                 down     1         full    1000    --
Eth100/1/30    --                 down     1         full    1000    --
Eth100/1/31    --                 down     1         full    1000    --
Eth100/1/32    --                 down     1         full    1000    --
Eth100/1/33    --                 down     1         full    1000    --
Eth100/1/34    --                 down     1         full    1000    --
Eth100/1/35    --                 down     1         full    1000    --
Eth100/1/36    --                 down     1         full    1000    --
Eth100/1/37    --                 down     89        full    1000    --
Eth100/1/38    --                 down     89        full    1000    --
Eth100/1/39    --                 down     89        full    1000    --
Eth100/1/40    --                 down     89        full    1000    --
Eth100/1/41    --                 down     89        full    1000    --
Eth100/1/42    --                 down     89        full    1000    --
Eth100/1/43    --                 down     89        full    1000    --
Eth100/1/44    --                 up       89        full    1000    --
Eth100/1/45    --                 up       89        full    1000    --
Eth100/1/46    --                 up       89        full    1000    --
Eth100/1/47    --                 up       89        full    1000    --
Eth100/1/48    --                 up       89        full    1000    --
NX5000#

NX5000# show interface ethernet 1/17 fex-intf
Fabric           FEX
Interface        Interfaces
---------------------------------------------------
 Eth1/17         Eth100/1/24   Eth100/1/23   Eth100/1/22   Eth100/1/21
                 Eth100/1/20   Eth100/1/19   Eth100/1/18   Eth100/1/17
                 Eth100/1/16   Eth100/1/15   Eth100/1/14   Eth100/1/10
                 Eth100/1/9    Eth100/1/8    Eth100/1/7    Eth100/1/6
                 Eth100/1/5    Eth100/1/4    Eth100/1/13   Eth100/1/12
                 Eth100/1/11   Eth100/1/3    Eth100/1/2    Eth100/1/1
```

```
NX5000# show interface ethernet 1/29 fex-intf
Fabric          FEX
Interface       Interfaces
-------------------------------------------------
 Eth1/29        Eth100/1/48   Eth100/1/47   Eth100/1/46   Eth100/1/45
                Eth100/1/44   Eth100/1/43   Eth100/1/42   Eth100/1/41
                Eth100/1/40   Eth100/1/39   Eth100/1/38   Eth100/1/37
                Eth100/1/36   Eth100/1/35   Eth100/1/34   Eth100/1/33
                Eth100/1/32   Eth100/1/31   Eth100/1/30   Eth100/1/29
                Eth100/1/28   Eth100/1/27   Eth100/1/26   Eth100/1/25
```

Configuring Nexus 2000 Using Port-Channels

This section demonstrates the configuration of the Nexus 2000 for the topology in Figure 2-3, using port-channels instead of static pinning.

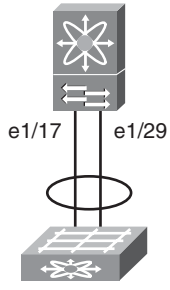

Figure 2-3 *Nexus 2000 Port-Channel Topology*

In the next example, we configure a similar topology using port-channels instead of static pinning. The configuration in Example 2-3 is similar to that of Example 2-1; however, in this method the **pinning max-links** parameter is set to one, and the individual interfaces are configured for a Port-Channel.

Example 2-3 *Nexus 2000 Port-Channel Configuration*

```
NX5000# config t
NX5000(config)# fex 100
NX5000(config-fex)# pinning max-links 1
Change in Max-links will cause traffic disruption.
NX5000(config-fex)# exit
NX5000(config)# interface port-channel 100
NX5000(config-if)# switchport mode fex-fabric
NX5000(config-if)# fex associate 100
```

```
NX5000(config-if)# no shutdown
NX5000(config-if)# exit
NX5000(config)# int e1/17,e1/29
NX5000(config-if-range)# channel-group 100 mode on
NX5000(config-if-range)# no shutdown
NX5000(config-if-range)# exit
```

Nexus 2000 Static Pinning Verification

Verification of the Nexus 2000 is similar whether port-channels or static pinning is used; however, all ports will now be pinned to the logical port-channel interface, as shown in Example 2-4.

Example 2-4 *Nexus 2000 Port-Channel Verification*

```
NX5000# sho fex
  FEX            FEX             FEX                  FEX
Number       Description         State      Model            Serial
------------------------------------------------------------------------
  100            FEX100        Online      N2K-C2148T-1GE   JAF1318AALS
NX5000# sho fex 100 det
FEX: 100 Description: FEX100   state: Online
  FEX version: 4.0(1a)N2(1a) [Switch version: 4.0(1a)N2(1a)]
  FEX Interim version: 4.0(1a)N2(1a)
  Switch Interim version: 4.0(1a)N2(1a)
  Extender Model: N2K-C2148T-1GE, Extender Serial: JAF1318AALS
  Part No: 73-12009-05
  Card Id: 70, Mac Addr: 00:0d:ec:cd:26:c2, Num Macs: 64
  Module Sw Gen: 19 [Switch Sw Gen: 19]
 pinning-mode: static    Max-links: 1
  Fabric port for control traffic: Eth1/29
  Fabric interface state:
    Po100 - Interface Up. State: Active
    Eth1/17 - Interface Up. State: Active
    Eth1/29 - Interface Up. State: Active
  Fex Port        State  Fabric Port  Primary Fabric
       Eth100/1/1    Up       Po100       Po100
       Eth100/1/2    Up       Po100       Po100
       Eth100/1/3    Up       Po100       Po100
       Eth100/1/4  Down       Po100       Po100
       Eth100/1/5  Down       Po100       Po100
       Eth100/1/6  Down       Po100       Po100
       Eth100/1/7  Down       Po100       Po100
       Eth100/1/8  Down       Po100       Po100
       Eth100/1/9  Down       Po100       Po100
      Eth100/1/10    Up       Po100       Po100
```

```
      Eth100/1/11    Up       Po100       Po100
      Eth100/1/12    Up       Po100       Po100
      Eth100/1/13    Up       Po100       Po100
      Eth100/1/14  Down       Po100       Po100
      Eth100/1/15  Down       Po100       Po100
      Eth100/1/16  Down       Po100       Po100
      Eth100/1/17  Down       Po100       Po100
      Eth100/1/18  Down       Po100       Po100
      Eth100/1/19  Down       Po100       Po100
      Eth100/1/20  Down       Po100       Po100
      Eth100/1/21  Down       Po100       Po100
      Eth100/1/22  Down       Po100       Po100
      Eth100/1/23  Down       Po100       Po100
      Eth100/1/24  Down       Po100       Po100
      Eth100/1/25  Down       Po100       Po100
      Eth100/1/26  Down       Po100       Po100
      Eth100/1/27  Down       Po100       Po100
      Eth100/1/28  Down       Po100       Po100
      Eth100/1/29  Down       Po100       Po100
      Eth100/1/30  Down       Po100       Po100
      Eth100/1/31  Down       Po100       Po100
      Eth100/1/32  Down       Po100       Po100
      Eth100/1/33  Down       Po100       Po100
      Eth100/1/34  Down       Po100       Po100
      Eth100/1/35  Down       Po100       Po100
      Eth100/1/36  Down       Po100       Po100
      Eth100/1/37  Down       Po100       Po100
      Eth100/1/38  Down       Po100       Po100
      Eth100/1/39  Down       Po100       Po100
      Eth100/1/40  Down       Po100       Po100
      Eth100/1/41  Down       Po100       Po100
      Eth100/1/42  Down       Po100       Po100
      Eth100/1/43  Down       Po100       Po100
--------------------------------------------------------------------------------
      Eth100/1/44    Up       Po100       Po100
      Eth100/1/45    Up       Po100       Po100
      Eth100/1/46    Up       Po100       Po100
      Eth100/1/47    Up       Po100       Po100
      Eth100/1/48    Up       Po100       Po100
Logs:
[02/08/2010 18:26:44.953152] Module register received
[02/08/2010 18:26:44.954622] Registration response sent
[02/08/2010 18:26:44.989224] Module Online Sequence
[02/08/2010 18:26:46.868753] Module Online
[02/08/2010 19:15:20.492760] Module disconnected
```

```
[02/08/2010 19:15:20.493584] Offlining Module
[02/08/2010 19:15:20.494099] Module Offline Sequence
[02/08/2010 19:15:20.905145] Module Offline
[02/08/2010 19:15:57.354031] Module register received
[02/08/2010 19:15:57.355002] Registration response sent
[02/08/2010 19:15:57.383437] Module Online Sequence
[02/08/2010 19:15:59.212748] Module Online
NX5000#
```

Layer 2 Forwarding on a Nexus 7000

The Nexus 7000 is an entirely distributed Layer 2 forwarding platform. This means that every module in the system contains its own forwarding table. When a packet is received on a port, the ingress module performs both an ingress L2 lookup and an initial egress L2 lookup. When the packet arrives at the egress module, a second egress lookup is performed to ensure that the table has not changed. Each module is also responsible for learning MAC addresses in the local hardware. MAC addresses learned by an individual module are flooded across the fabric to all other modules in the system, and an additional software process ensures that MAC addresses are properly synchronized across the hardware modules. Aging of MAC addresses is also done locally by each line card but only for primary entries (entries learned locally). When a module ages a MAC address, it also notifies the supervisors so that the address can be removed from the other modules. MAC address aging is configurable on a per-VLAN basis, with a limit of 14 unique aging values per system.

To configure the MAC address aging timer, enter the following command:

```
switch(config)# mac address-table aging-time 600
```

To create a static MAC entry, enter the following command:

```
Congo(config)# mac address-table static 12ab.47dd.ff89 vlan 1 interface ethernet 2/1
```

L2 Forwarding Verification

During the normal operation of a switched infrastructure, certain tasks are required to validate the L2 forwarding process. These tasks include displaying the MAC address table to identify connected nodes or validate switching paths. In certain cases, it might also be necessary to clear the MAC address table, forcing the switch to repopulate with the latest information. The following examples clear the MAC table, create a static MAC entry, validate that the entry is inserted into the MAC table, and finally validate that it is synchronized across all modules within the system. Example 2-5 shows how to clear the MAC address table.

Example 2-5 *Clearing MAC Address Table*

```
Congo# clear mac address-table dynamic
Congo(config)# sho mac address-table aging-time
Vlan    Aging Time
----    ----------
1       600
```

Example 2-6 shows how to display the MAC address table.

Example 2-6 *Displaying the MAC Address Table*

```
Congo# show mac address-table static
Legend:
        * - primary entry, G - Gateway MAC, (R) - Routed MAC
        age - seconds since last seen,+ - primary entry using vPC Peer-Link
   VLAN     MAC Address      Type      age     Secure NTFY    Ports
---------+-----------------+--------+---------+------+----+------------------
G     -    001b.54c2.bbc1    static       -       F    F  sup-eth1(R)
* 1        12ab.47dd.ff89    static       -       F    F  Eth2/1
```

Due to the distributed forwarding nature of the Nexus 7000, each line card maintains its own forwarding table, which is synchronized across all cards. To verify the synchronization of tables, use the **show forwarding consistency l2 command**, as demonstrated in Example 2-7.

Example 2-7 *Checking Forwarding Table Consistency*

```
Congo# sho forwarding consistency l2 1
Legend:
        * - primary entry, G - Gateway MAC, (R) - Routed MAC
        age - seconds since last seen,+ - primary entry using vPC Peer-Link
Missing entries in the MAC Table
   VLAN     MAC Address      Type      age     Secure NTFY    Ports
---------+-----------------+--------+---------+------+----+------------------

Extra and Discrepant entries in the MAC Table
   VLAN     MAC Address      Type      age     Secure NTFY    Ports
---------+-----------------+--------+---------+------+----+------------------
Congo#
```

If there were any discrepancies between the two line cards, they would appear in the preceding output. Under normal circumstances, the two line cards should always be consistent and thus produce no output.

VLANs

VLANs provide a mechanism to segment traffic on a single switch into isolated networks. VLANs can be used to segment a switch for many reasons including security, business unit, or application/function. VLANs are configured on each switch in a given topology but can span multiple physical switches using 802.1Q trunks.

The Nexus 7000 switch supports 4096 VLANs per Virtual Device Context (VDC) for a system total of ~16k VLANs. Some of these VLANs are used by system-level functions and are not user-configurable. You can display the internal VLANs by using the **show vlan internal usage** command, as demonstrated in Example 2-8.

Example 2-8 *Internal VLAN Usage*

```
Congo# show vlan internal usage
VLAN        DESCRIPTION
---------   ----------------------------------------------------
3968-4031   Multicast
4032        Online diagnostics vlan1
4033        Online diagnostics vlan2
4034        Online diagnostics vlan3
4035        Online diagnostics vlan4
4036-4047   Reserved
4094        Reserved
```

Configuring VLANs

VLANs are configured in global configuration mode with the **vlan** *vlan-id configuration command*.

Example 2-9 shows how to add a VLAN to the local database.

Example 2-9 *Creating a New VLAN*

```
Congo# config t
Enter configuration commands, one per line. End with CNTL/Z.
Congo(config)# vlan 10
Congo(config-vlan)# name newvlan
```

Example 2-10 shows how you can create multiple VLANs by specifying a range using the **vlan** *vlan-range* command.

Example 2-10 *Creating Multiple VLANs*

```
Congo# config t
Enter configuration commands, one per line. End with CNTL/Z.
Congo(config)# vlan 10-15
```

```
Congo(config-vlan)# exit
```

VLAN Trunking Protocol

In large switched networks, VLAN Trunking Protocol (VTP) is sometimes used to allow the dissemination of VLAN definition across a large number of switches.

Note At press time, VTP is supported only on the Nexus 7000, and only transparent mode is supported.

With VTP, devices can operate in one of four distinct modes:

- **Off:** By default, NX-OS devices do not run VTP. Devices that are not running VTP will not send or receive VTP advertisements and will break the flow of VTP advertisements when inserted between two VTP devices.
- **VTP server mode:** In VTP server mode, VLANs can be created, modified, and deleted. VTP servers also define domainwide parameters such as a version and whether VTP pruning will be in effect. VTP servers send VTP advertisements to other devices within the VTP domain and update the VLAN database with advertisements received from other devices in the domain.
- **VTP Client mode:** VTP clients send and receive VTP advertisements and update their local VLAN database based on these advertisements; however, you cannot create, modify, or delete VLANs locally on the device.
- **VTP transparent mode:** Devices operating in VTP transparent mode relay messages received from other devices but do not advertise changes made to the devices' local database, nor will they modify the local database based on information received from other devices.

To configure VTP transparent mode, the code base must be loaded into memory by using the **feature** command, as demonstrated in Example 2-11.

Example 2-11 *Enabling the VTP Feature*

```
Congo# config t
Enter configuration commands, one per line. End with CNTL/Z.
Congo(config)# feature vtp
```

Example 2-12 shows how to specify VTP parameters in global configuration mode.

Example 2-12 *Specifying VTP Parameters*

```
Congo# config t
```

```
Enter configuration commands, one per line. End with CNTL/Z.
Congo(config)# vtp domain cisco
Congo(config)# vtp mode transparent
```

Assigning VLAN Membership

After the VLAN database has been created, ports can now be added to the VLAN based on the requirements of the devices connected to the switch. Additionally, links between switches might be required to carry multiple VLANs.

Example 2-13 shows how to add a port to a VLAN.

Example 2-13 *Adding a Port to a VLAN*

```
Kenya# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Kenya(config)# interface ethernet 2/20
Kenya(config-if)# switchport
Kenya(config-if)# switchport mode access
Kenya(config-if)# switchport access vlan 10
Kenya(config-if)#
```

Example 2-14 shows how to create a trunk interface.

Example 2-14 *Configuring a Trunk Interface*

```
Kenya# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Kenya(config)# interface ethernet 2/11
Kenya(config-if)# switchport
Kenya(config-if)# switchport mode trunk
Kenya(config-if)#
```

With this configuration, the trunk port carries all VLANs that are active in the local VLAN database. It is best practice to manually prune unnecessary trunk ports, limiting the VLANs carried to only those necessary using the following syntax:

```
Kenya\(config-if)# switchport trunk allowed vlan 10-20
```

As requirements change, it might be necessary to add or remove VLANs from a trunk port, using the **add** or **remove** keywords to the **switchport trunk allowed** command, as demonstrated in Example 2-15.

Example 2-15 *Adding and Removing VLANs from a Trunk*

```
Kenya(config-if)# switchport trunk allowed vlan add 5
Kenya(config-if)# switchport trunk allowed vlan remove 15
Kenya(config-if)# sho run interface ethernet 2/11
!Command: show running-config interface Ethernet2/11
!Time: Thu Oct 29 18:27:10 2009
version 4.2(2a)
interface Ethernet2/11
  switchport
  switchport mode trunk
  switchport trunk allowed vlan 5,10-14,16-20
  spanning-tree port type network
  no shutdown
```

Verifying VLAN Configuration

You can view the configured VLANs and the interfaces assigned to them with the **show vlan** command, as demonstrated in Example 2-16.

Example 2-16 *View Configured LANs and Interfaces*

```
Kenya# show vlan
VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
1    default                           active    Eth2/12, Eth2/40
2    VLAN0002                          active    Eth2/12
3    VLAN0003                          active    Eth2/12
4    VLAN0004                          active    Eth2/12
5    VLAN0005                          active    Eth2/11, Eth2/12
6    VLAN0006                          active    Eth2/12
7    VLAN0007                          active    Eth2/12
8    VLAN0008                          active    Eth2/12
9    VLAN0009                          active    Eth2/12
10   VLAN0010                          active    Eth2/11, Eth2/12, Eth2/20
11   VLAN0011                          active    Eth2/11, Eth2/12
12   VLAN0012                          active    Eth2/11, Eth2/12
13   VLAN0013                          active    Eth2/11, Eth2/12
14   VLAN0014                          active    Eth2/11, Eth2/12
15   VLAN0015                          active    Eth2/12
16   VLAN0016                          active    Eth2/11, Eth2/12
17   VLAN0017                          active    Eth2/11, Eth2/12
18   VLAN0018                          active    Eth2/11, Eth2/12
19   VLAN0019                          active    Eth2/11, Eth2/12
20   VLAN0020                          active    Eth2/11, Eth2/12
VLAN Type
```

```
---- -----
1    enet
2    enet
3    enet
4    enet
5    enet
6    enet
7    enet
8    enet
9    enet
10   enet
11   enet
12   enet
13   enet
14   enet
15   enet
16   enet
17   enet
18   enet
19   enet
20   enet
Remote SPAN VLANs
------------------------------------------------------------------------------
Primary  Secondary  Type             Ports
-------  ---------  ---------------  ---------------------------------------
```

Private VLANs

Private VLANs (PVLAN) offer a mechanism to divide a single VLAN into multiple isolated Layer 2 networks. PVLANs can be configured on a single Nexus switch or extended across multiple devices by trunking all the primary, isolated, and community VLANs to any other devices that need to participate in the PVLAN domain. Private VLANs are useful in several scenarios:

- **IP address management:** Typically speaking, a one-to-one relationship exists between a VLAN and an IP subnet. In situations in which many VLANs are required with a relatively small number of hosts per subnet, PVLANs can be used to configure aggregation layer with a larger subnet, and configure each isolated group of hosts into isolated VLANs, thus not requiring an IP address or subnet mask change if the host is moved from one isolated VLAN to another.
- **Security:** PVLANs offer an additional level of security at Layer 2. Isolated VLANs are allowed to communicate only at Layer 2 with other members of the same isolated VLAN. If communication between isolated VLANs is required, the communication

must flow through an upstream router or firewall, making it possible to apply security policy on hosts within the same broadcast domain.

- **Broadcast suppression:** PVLANs can also be used to control the propagation of broadcast traffic only to those devices that can benefit from receiving certain broadcasts.

Within a PVLAN domain, the two major types of VLANs follow:

- **Primary:** The primary VLAN is where the broadcast domain is defined. Promiscuous ports are part of the primary VLAN and can communicate with all other ports in the primary VLAN, and all isolated and community VLAN ports.
- **Secondary:** Subdomains that share IP address space with the primary VLAN but are isolated from each other in one of two ways:
  - **Isolated VLANs:** Each port within an isolated VLAN is restricted such that it can communicate only with promiscuous ports in the primary VLAN. Ports within the isolated VLANs cannot receive broadcasts from any other devices.
  - **Community VLANs:** Community VLAN ports are restricted from communicating with other community VLANs but might communicate with other ports in the same community VLAN and promiscuous ports belonging to the primary VLAN.

Multiple secondary VLANs can be associated with a single primary VLAN. These associations define a PVLAN domain.

Configuring PVLANs

In the following examples, you see the configuration of six hosts to meet the following requirements:

- **Host1(192.168.100.21):** Communicates only with Host2 and its default gateway
- **Host2(192.168.100.22):** Communicates only with Host1 and its default gateway
- **Host3(192.168.100.23):** Communicates only with Host4 and its default gateway
- **Host4(192.168.100.24):** Communicates only with Host3 and its default gateway
- **Host5(192.168.100.25):** Sends traffic only to its default gateway
- **Host6(192.168.100.26):** Sends traffic only to its default gateway

Figure 2-4 provides a visual representation of the configuration in the following examples.

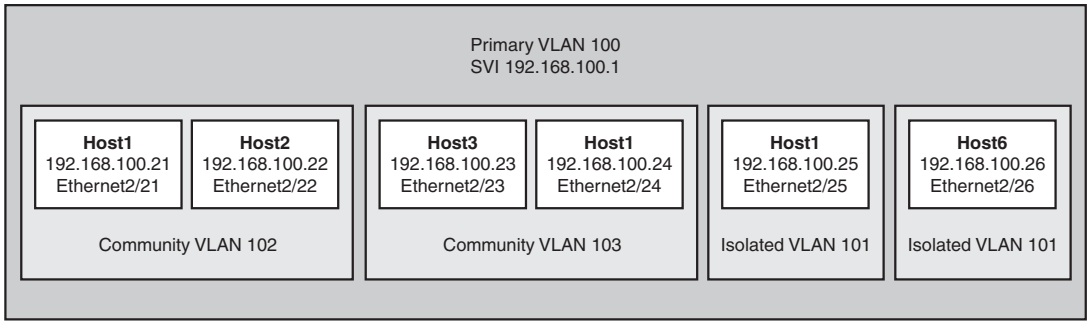

Figure 2-4 *Network Topology for PVLAN Configuration*

First, activate the code base for private VLANs by using the **feature** command as demonstrated in Example 2-17.

Example 2-17 *Enable Private VLANs*

```
Congo# config t
Enter configuration commands, one per line. End with CNTL/Z.
Congo(config)# feature private-vlan
```

Example 2-18 demonstrates how to configure the primary VLAN, isolated, and community VLANs and define their relationship.

Example 2-18 *Defining Private VLANs*

```
Congo(config)# vlan 101
Congo(config-vlan)# name VLAN100-ISOLATED
Congo(config-vlan)# private-vlan isolated
Congo(config-vlan)# vlan 102
Congo(config-vlan)# name VLAN100-COMMUNITY1
Congo(config-vlan)# private-vlan community
Congo(config-vlan)# vlan 103
Congo(config-vlan)# name VLAN100-COMMUNITY2
Congo(config-vlan)# private-vlan community
Congo(config-vlan)#
Congo(config)# vlan 100
Congo(config-vlan)# name VLAN100-PRIMARY
Congo(config-vlan)# private-vlan primary
Congo(config-vlan)# private-vlan association add 101-103
Congo(config-vlan)# exit
```

Example 2-19 shows how to define a Layer 3 switched virtual interface (SVI) and associate secondary Layer 2 VLANs.

Example 2-19 *Creating an SVI for the Primary VLAN*

```
Congo(config)# interface vlan 100
Congo(config-if)# ip address 192.168.100.1/24
Congo(config-if)# private-vlan mapping add 101-103
Congo(config-if)# exit
Congo(config)#
```

Example 2-20 shows how to define the PVLAN configuration on the access switch and assign the host ports into the appropriate secondary VLANs.

Example 2-20 *Private VLAN Access Switch Configuration*

```
Kenya(config)# feature private-vlan
Kenya(config)# vlan 101
Kenya(config-vlan)# name VLAN100-ISOLATED
Kenya(config-vlan)# private-vlan isolated
Kenya(config-vlan)# vlan 102
Kenya(config-vlan)# name VLAN100-COMMUNITY1
Kenya(config-vlan)# private-vlan community
Kenya(config-vlan)# vlan 103
Kenya(config-vlan)# name VLAN100-COMMUNITY2
Kenya(config-vlan)# private-vlan community
Kenya(config-vlan)# vlan 100
Kenya(config-vlan)# name VLAN100-PRIMARY
Kenya(config-vlan)# private-vlan primary
Kenya(config-vlan)# private-vlan association add 101-103
Kenya(config-vlan)#
Kenya(config)# interface ethernet2/21
Kenya(config-if)# description HOST1
Kenya(config-if)# switchport
Kenya(config-if)# switchport mode private-vlan host
Kenya(config-if)# switchport private-vlan host-association 100 102
Kenya(config-if)# exit
Kenya(config)# interface ethernet2/22
Kenya(config-if)# description HOST2
Kenya(config-if)# switchport
Kenya(config-if)# switchport mode private-vlan host
Kenya(config-if)# switchport private-vlan host-association 100 102
Kenya(config-if)# exit
Kenya(config)# interface ethernet2/23
Kenya(config-if)# description HOST3
```

```
Kenya(config-if)# switchport
Kenya(config-if)# switchport mode private-vlan host
Kenya(config-if)# switchport private-vlan host-association 100 103
Kenya(config-if)# exit
Kenya(config)# interface ethernet2/24
Kenya(config-if)# description HOST4
Kenya(config-if)# switchport
Kenya(config-if)# switchport private-vlan host-association 100 103
Kenya(config-if)# exit
Kenya(config)# interface ethernet2/25
Kenya(config-if)# description HOST5
Kenya(config-if)# switchport mode private-vlan host
Kenya(config-if)# switchport
Kenya(config-if)# switchport mode private-vlan host
Kenya(config-if)# switchport private-vlan host-association 100 101
Kenya(config-if)# exit
Kenya(config)# interface ethernet2/26
Kenya(config-if)# description HOST6
Kenya(config-if)# switchport
Kenya(config-if)# switchport mode private-vlan host
Kenya(config-if)# switchport private-vlan host-association 100 101
Kenya(config-if)# exit
```

Verifying PVLAN Configuration

Example 2-21 shows how to verify the mapping of the SVI for the primary VLAN and associated secondary VLANs.

Example 2-21 *Verifying Layer 3 SVI PVLAN Mapping*

```
Congo# show interface private-vlan mapping
Interface Secondary VLAN Type
--------- -------------- -----------------
vlan100   101            isolated
vlan100   102            community
vlan100   103            community
```

Example 2-22 shows how to verify the mapping of the primary VLANs, the associated secondary VLANs, and the host ports that belong to each on the access switch Kenya.

Example 2-22 *Verifying PVLAN Mapping*

```
Kenya# show vlan private-vlan
Primary  Secondary  Type            Ports
```

```
-------  ---------  --------------  ----------------------------------------
100      101        isolated        Eth2/25, Eth2/26
100      102        community       Eth2/21, Eth2/22
100      103        community       Eth2/23, Eth2/24
```

Spanning Tree Protocol

The Spanning Tree Protocol provides a mechanism for physically redundant network topologies to remain logically loop free. All devices in a bridging domain run spanning-tree calculations to discover the topology and calculate the best path to the root bridge. Through the spanning-tree process, redundant network links are placed into a blocking state preventing loops from occurring at Layer 2.

The Nexus series of switches implements two forms of standards-based Spanning Tree Protocols:

- **Rapid Per-VLAN Spanning Tree (Rapid-PVST/802.1w):** Rapid-PVST is the default spanning-tree mode on Nexus 7000 switches. As the name implies, in Rapid-PVST, each VLAN elects a single root bridge, and all other devices determine the lowest cost path to the root bridge. With Rapid-PVST topology, changes are isolated to that particular VLAN. One additional characteristic worth noting is that 802.1w is backward compatible with standard Per-VLAN Spanning Tree (PVST/802.1d) for migration or interoperability purposes.
- **Multiple Spanning Tree (MST/802.1s):** In large Layer 2 environments, MST can be used to provide a much simpler configuration with lower control plane overhead than Rapid-PVST. When MST is leveraged, VLANs with similar topologies share a single spanning-tree instance. MST instances with identical names, revision numbers, and VLAN mappings create a construct called an MST region. For further simplification of complex Layer 2 domains, each MST region is presented to the network as a single bridge. It is also worth noting that MST is backward compatible with Rapid-PVST.

For the following common data center configuration examples, refer to the topology illustrated in Figure 2-5.

In Figure 2-5, Congo and Egypt are redundant data center aggregation switches. First, the aggregation switches will be configured for Rapid-PVST+ with Congo configured as the root bridge for VLANs 1 to 10 (depicted in Figure 2-6) and Egypt as root for VLANs 11 to 20 (depicted in Figure 2-7) in the aggregation block. This type of *root staggering* is often desirable to maximize the amount of bandwidth that is available by reducing the number of blocking links within the spanning tree.

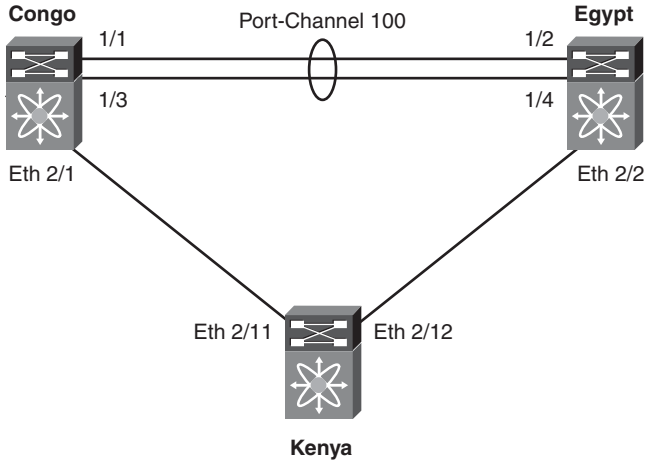

Figure 2-5 *Common Data Center Topology*

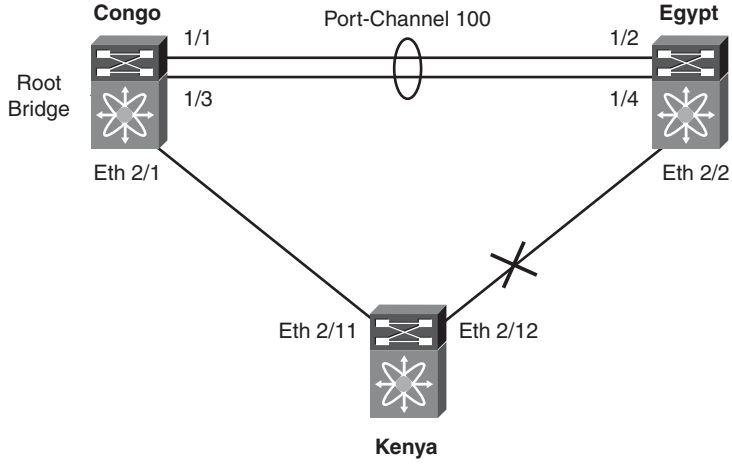

Figure 2-6 *STP Topology for VLANs 1 to 10*

Rapid-PVST+ Configuration

Typically, root bridge placement is influenced by modifying the priority. On NX-OS and most IOS devices, the default bridge priority is 32768, so you will be configuring considerably lower values on the aggregation switches.

Example 2-23 shows how to configure the spanning-tree priority on a range of VLANs.

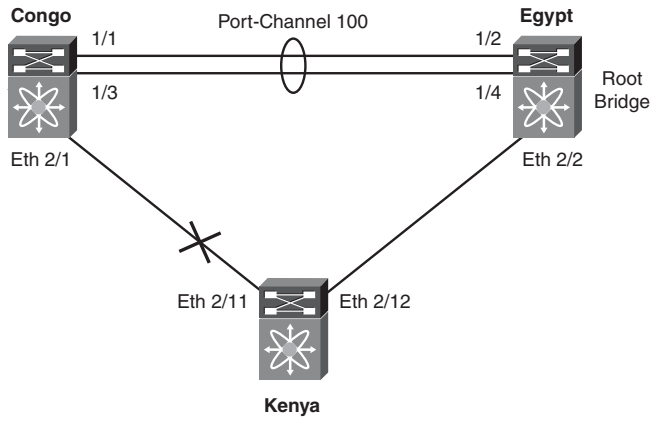

Figure 2-7 *STP Topology for VLANs 11 to 20*

Example 2-23 *Configuring Spanning Tree Bridge Priority*

```
Congo# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Congo(config)# spanning-tree mode rapid-pvst
Congo(config)# vlan 1-20
Congo(config-vlan)# exit
Congo(config)# spanning-tree vlan 1-10 priority 4096
Congo(config)# spanning-tree vlan 11-20 priority 8192
Congo(config)#
---------------------------------------------------------------------
Egypt# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Egypt(config)# spanning-tree mode rapid-pvst
Egypt(config)# vlan 1-20
Egypt(config-vlan)# exit
Egypt(config)# spanning-tree vlan 1-10 priority 8192
Egypt(config)# spanning-tree vlan 11-20 priority 4096
Egypt(config)#
```

Alternatively, you can manipulate the spanning-tree priority values using the **root** keyword, as demonstrated in Example 2-24.

Example 2-24 *Using the spanning-tree root Keyword*

```
Configuration on Congo
Congo(config)#spanning-tree vlan 1-10 root primary
Congo(config)#spanning-tree vlan 11-20 root secondary
---------------------------------------------------------------------
Egypt(config)#spanning-tree vlan 1-10 root secondary
```

```
Egypt(config)#spanning-tree vlan 11-20 root primary
```

Verifying Spanning-Tree State for a VLAN

Understanding the spanning-tree topology on a specific VLAN is important to ensure that the topology behaves as expected if a link or bridge failure occurs. Inconsistent spanning-tree configurations can lead to unexpected outages or slower reconvergence. Example 2-25 shows how to verify the spanning-tree state for a particular VLAN.

Example 2-25 *Displaying Spanning-Tree Information for a Specific VLAN*

```
Congo# show spanning-tree vlan 10
VLAN0010
  Spanning tree enabled protocol rstp
  Root ID    Priority    4106
             Address     001b.54c2.bbc1
             This bridge is the root
             Hello Time  2  sec  Max Age 12 sec  Forward Delay 9  sec
  Bridge ID  Priority    4106   (priority 4096 sys-id-ext 10)
             Address     001b.54c2.bbc1
             Hello Time  2  sec  Max Age 12 sec  Forward Delay 9  sec
Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Po100            Desg FWD 1         128.4195 Network P2p
Eth2/1           Desg FWD 4         128.257  Network P2p
```

Example 2-26 shows how to verify that Kenya has selected the best path to root; in this case, Ethernet 2/11 is blocking the redundant connection to Egypt.

Example 2-26 *Confirming Spanning Tree Bridge Priority*

```
Kenya# show spanning-tree vlan 10
VLAN0010
  Spanning tree enabled protocol rstp
  Root ID    Priority    4106
             Address     001b.54c2.bbc1
             Cost        4
             Port        267 (Ethernet2/11)
             Hello Time  2  sec  Max Age 12 sec  Forward Delay 9  sec
  Bridge ID  Priority    32778  (priority 32768 sys-id-ext 10)
             Address     001b.54c2.bbc3
             Hello Time  2  sec  Max Age 20 sec  Forward Delay 15 sec
Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Eth2/11          Root FWD 4         128.267  Network P2p
```

```
Eth2/12          Altn BLK 4         128.268  Network P2p
Kenya#
```

Spanning-Tree Timers

The hello, forward-delay, and max-age timers determine the operational characteristics of the spanning-tree bridge.

The hello timer defines how often the bridge sends Bridge Protocol Data Units (BPDU) to connected devices. On NX-OS, the default is 2 seconds but can be configured for 1 to 10 seconds.

The forward-delay timer specifies how long the bridge stays in the listening and learning states before transitioning into a forwarding state. By default, NX-OS waits 15 seconds before transitioning the port from listening to learning, and from learning to forwarding. The forward-delay timer is configurable from 15 to 30 seconds.

The max-age timer ensures backward compatibility with traditional 802.1D spanning-tree environments by specifying the length of time a BPDU received on a given port is stored. The default NX-OS max-age time is 20 seconds and can be configured from 6 to 40 seconds.

Example 2-27 shows how to verify the spanning-tree timers for a specific VLAN.

Example 2-27 *Default Spanning-Tree Timers*

```
Congo# show spanning-tree vlan 10
VLAN0010
  Spanning tree enabled protocol rstp
  Root ID    Priority    4106
             Address     001b.54c2.bbc1
             This bridge is the root
             Hello Time  2  sec  Max Age 20 sec  Forward Delay 15 sec
  Bridge ID  Priority    4106   (priority 4096 sys-id-ext 10)
             Address     001b.54c2.bbc1
             Hello Time  2  sec  Max Age 20 sec  Forward Delay 15 sec
Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Po100            Desg FWD 1         128.4195 (vPC peer-link) Network P2p
```

In smaller L2 domains, faster reconvergence can be achieved by manipulating these timers. Example 2-28 shows how to manually adjust the hello, forward-delay, and max-age timers.

Caution Although it might be desirable to manipulate spanning-tree timers for faster reconvergence, these timers and the Layer 2 topology should be well understood. Incorrect spanning-tree timers can produce undesirable results.

Example 2-28 *Modifying Spanning Tree Timers*

```
Congo(config)# spanning-tree mode rapid-pvst
Congo(config)# spanning-tree vlan 10 hello-time 1
Congo(config)# spanning-tree vlan 10 forward-time 10
Congo(config)# spanning-tree vlan 10 max-age 15
Congo(config)#
Congo(config)# sho spanning-tree vlan 10
VLAN0010
  Spanning tree enabled protocol rstp
  Root ID    Priority    4106
             Address     001b.54c2.bbc1
             This bridge is the root
             Hello Time  1  sec  Max Age 15 sec  Forward Delay 10 sec
  Bridge ID  Priority    4106   (priority 4096 sys-id-ext 10)
             Address     001b.54c2.bbc1
             Hello Time  1  sec  Max Age 15 sec  Forward Delay 10 sec
Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Po100            Desg FWD 1         128.4195 (vPC peer-link) Network P2p
```

To mitigate some of the risk associated with the manual manipulation of spanning-tree timers, NX-OS provides the **diameter** keyword, and if needed, adjusts these timers according to best practices. In this topology, a single-tier Layer 2 design is implemented with access switches connecting to both aggregation switches; therefore, the maximum number of bridges between any two stations (diameter) is 3. If no diameter is specified, the default of 7 applies. By specifying the diameter of the spanning-tree domain, hello, forward-delay, and max-age timers are adjusted for optimal reconvergence in the event that a spanning-tree recalculation occurs.

Example 2-29 demonstrates how the **diameter** keyword is used to manipulate spanning-tree timers.

Example 2-29 *Specifying the Spanning-Tree Diameter*

```
Congo(config)#spanning-tree mode rapid-pvst
Congo(config)#spanning-tree vlan 1-10 root primary diameter 3
Congo(config)#spanning-tree vlan 11-20 root secondary diameter 3
--------------------------------------------------------------------------------
Egypt(config)# spanning-tree vlan 1-10 root primary diameter 3
```

```
Egypt(config)# spanning-tree vlan 11-20 root secondary diameter 3
Egypt(config)# sho spanning-tree vlan 10
VLAN0010
  Spanning tree enabled protocol rstp
  Root ID    Priority    4106
             Address     001b.54c2.bbc1
             This bridge is the root
             Hello Time  2  sec  Max Age 12 sec  Forward Delay 9  sec
  Bridge ID  Priority    4106   (priority 4096 sys-id-ext 10)
             Address     001b.54c2.bbc1
             Hello Time  2  sec  Max Age 12 sec  Forward Delay 9  sec
Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Po100            Desg FWD 1         128.4195 (vPC peer-link) Network P2p
Egypt(config)# spanning-tree vlan 1-10 root primary diameter 3
Egypt(config)# spanning-tree vlan 11-20 root secondary diameter 3
Egypt(config)# sho spanning-tree vlan 10
VLAN0010
  Spanning tree enabled protocol rstp
  Root ID    Priority    4106
             Address     001b.54c2.bbc1
             This bridge is the root
             Hello Time  2  sec  Max Age 12 sec  Forward Delay 9  sec
  Bridge ID  Priority    4106   (priority 4096 sys-id-ext 10)
             Address     001b.54c2.bbc1
             Hello Time  2  sec  Max Age 12 sec  Forward Delay 9  sec
Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Po100            Desg FWD 1         128.4195 (vPC peer-link) Network P2p
```

As you can see in the previous example, the hello time, max age, and forward delay have been adjusted based on the STP diameter keyword.

MST Configuration

The examples in this section demonstrate the same configuration as the previous section using MST instead of Rapid-PVST. The configuration steps are similar; however, you see the additional steps of creating an instance, defining a revision number, and associating VLANs with an instance. These steps are required to define which VLANs share the same spanning-tree topology within MST.

Example 2-30 demonstrates basic MST configuration.

Example 2-30 *Basic MST Configuration*

```
Congo# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Congo(config)# spanning-tree mode mst
Congo(config)# spanning-tree mst configuration
Congo(config-mst)# name AGG1
Congo(config-mst)# revision 10
Congo(config-mst)# instance 1 vlan 1-10
Congo(config-mst)# instance 2 vlan 11-20
Congo(config-mst)#
```

Prior to exiting from MST configuration mode, it is recommended to review the changes being proposed. *Existing MST mode commits all changes prior to exiting.*

Example 2-31 shows MST verification.

Example 2-31 *MST Verification*

```
Congo(config-mst)# show pending
Pending MST Configuration
Name      [AGG1]
Revision  10    Instances configured 3
Instance  Vlans mapped
--------  ------------------------------------------------------------------
0         21-4094
1         1-10
2         11-20
-------------------------------------------------------------------------------
Congo(config-mst)# exit
Congo(config)#
```

Because MST changes are not committed until you exit MST configuration mode, the administrator has the ability to back out of the configuration without committing the changes. During the configuration of Egypt, we misconfigure the instance mapping, abort the changes, and reconfigure appropriately.

Example 2-32 shows how to abort pending MST changes.

Example 2-32 *MST Misconfiguration*

```
Egypt(config)# spanning-tree mode mst
Egypt(config)# spanning-tree mst configuration
Egypt(config)# name AGG1
Egypt(config-mst)# revision 10
Egypt(config-mst)# instance 1 vlan 1-9
Egypt(config-mst)# instance 2 vlan 11-20
```

```
Egypt(config-mst)# show pending
Pending MST Configuration
Name      [AGG1]
Revision  10    Instances configured 3
Instance  Vlans mapped
--------  ------------------------------------------------------------------
0         10,21-4094
1         1-9
2         11-20
-------------------------------------------------------------------------------
Egypt(config-mst)# abort
Aborting and exiting region configuration mode
Egypt(config)# spanning-tree mst configuration
Egypt(config-mst)# spanning-tree mst configuration
Egypt(config-mst)# name AGG1
Egypt(config-mst)# revision 10
Egypt(config-mst)# instance 1 vlan 1-10
Egypt(config-mst)# instance 2 vlan 11-20
Egypt(config-mst)# show pending
Pending MST Configuration
Name      [AGG1]
Revision  10    Instances configured 3
Instance  Vlans mapped
--------  ------------------------------------------------------------------
0         21-4094
1         1-10
2         11-20
-------------------------------------------------------------------------------
Egypt(config-mst)# exit
Egypt(config)#
```

Because the instance 1 VLAN mapping was input incorrectly, the pending changes were aborted, and reconfigured correctly before exiting/committing.

If PVLANs are used within the environment, it is required that all secondary VLANs share the same MST instance as their associated primary VLAN. MST provides a mechanism to automatically enforce this.

Example 2-33 shows VLAN synchronization for MST.

Example 2-33 *Private VLAN Synchronization*

```
Congo(config)# spanning-tree mst configuration
Congo(config-mst)#private-vlan synchronize
```

Example 2-34 shows how to verify the spanning-tree configuration with the **show running-config spanning-tree** command.

Example 2-34 *Verifying MST Configuration*

```
Egypt(config)# sho run spanning-tree
spanning-tree mode mst
spanning-tree port type edge bpduguard default
spanning-tree port type edge bpdufilter default
spanning-tree mst configuration
  name AGG1
  revision 10
  instance 1 vlan 1-10
  instance 2 vlan 11-20
interface port-channel1
  spanning-tree port type network
  spanning-tree guard root
interface port-channel100
  spanning-tree port type network
interface Ethernet2/2
  spanning-tree port type network
```

Like Rapid-PVST+, the spanning-tree root placement can be influenced by modifying the priority for each bridge; however, instead of configuring the priority on a per-VLAN basis, MST switches are configured with a priority per-instance.

Example 2-35 shows how to adjust the priority for an MST instance.

Example 2-35 *MST Priority Configuration*

```
Congo(config)# spanning-tree mst 1 priority 4096
Congo(config)# spanning-tree mst 2 priority 8192
--------------------------------------------------------------------------
Egypt(config)# spanning-tree mst 1 priority 8192
Egypt(config)# spanning-tree mst 2 priority 4096
```

Alternatively, Example 2-36 shows how to use the **root** keyword as done previously in the Rapid-PVST section.

Example 2-36 *MST Root Configuration*

```
Configuration on Congo
Congo(config)#spanning-tree mst 1 root primary
Congo(config)#spanning-tree mst 2 root secondary
```

Example 2-37 shows how to verify the configuration of an MST instance.

Example 2-37 *MST Verification*

```
Congo# show spanning-tree mst 1
##### MST1    vlans mapped:   1-10
Bridge        address 001b.54c2.bbc1  priority      4097  (4096 sysid 1)
Root          this switch for MST1
Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Po100            Desg FWD 1000      128.4195 Network P2p
Eth2/1           Desg FWD 20000     128.257  Network P2p Bound(PVST)
Congo#
```

The example shows that Congo is the root bridge for MST instance 1, which has VLANs 1 to 10 mapped to it.

Additional Spanning-Tree Configuration

The sections that follow cover the configuration required to manipulate some additional spanning-tree parameters.

Port Cost

Port cost is used to calculate the shortest path to the root bridge. By default, port costs are automatically calculated by the device based on the transmission speed of the physical link. Table 2-1 illustrates the default port costs.

From time to time, it might be necessary to statically define port costs; an example of this is with port-channels in which the cost might change depending on the number of links active within the bundle.

Example 2-38 shows the root ports on Kenya prior to change to port cost.

Table 2-1 *Default Spanning Tree Costs*

Link Speed	Default Spanning Tree Cost
10 Mbps	100
100 Mbps	19
1000 Mbps	4
10,000 Mbps	2

Example 2-38 *MST Verification*

```
Kenya# show spanning-tree root
                                       Root  Hello Max Fwd
```

```
Vlan                 Root ID          Cost  Time  Age Dly  Root Port
---------------- ------------------- ------- ----- --- ---  ---------------
VLAN0001         4097 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0002         4098 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0003         4099 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0004         4100 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0005         4101 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0006         4102 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0007         4103 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0008         4104 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0009         4105 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0010         4106 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0011         4107 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0012         4108 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0013         4109 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0014         4110 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0015         4111 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0016         4112 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0017         4113 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0018         4114 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0019         4115 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0020         4116 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
Kenya#
```

Example 2-39 shows the configuration of port cost on a link.

Example 2-39 *Configuring Port Cost*

```
Kenya# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Kenya(config)# interface ethernet 2/11
Kenya(config-if)# spanning-tree cost 128
Kenya(config-if)# exit
Kenya(config)# exit
```

Example 2-40 shows the output after adjusting the port cost. Now Ethernet 2/12 is the root port for all VLANs.

Example 2-40 *Spanning Tree Root Verification*

```
Kenya# show spanning-tree root
                                     Root  Hello Max Fwd
Vlan                 Root ID          Cost  Time  Age Dly  Root Port
---------------- ------------------- ------- ----- --- ---  ---------------
```

```
VLAN0001          4097 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0002          4098 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0003          4099 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0004          4100 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0005          4101 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0006          4102 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0007          4103 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0008          4104 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0009          4105 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0010          4106 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0011          4107 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0012          4108 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0013          4109 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0014          4110 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0015          4111 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0016          4112 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0017          4113 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0018          4114 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0019          4115 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0020          4116 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
Kenya#
```

Example 2-41 shows the configuration of port cost on a per VLAN basis.

Example 2-41 *Per VLAN Cost Configuration*

```
Kenya(config)# interface ethernet 2/11
Kenya(config-if)# spanning-tree vlan 1,3,5,7,9 cost 4
Kenya(config-if)# exit
```

Example 2-42 shows the result of the changes in the previous example. Ethernet2/11 is now root for VLANs 1, 3, 5, 6, and 9.

Example 2-42 *Spanning Tree Root Verification*

```
Kenya# show spanning-tree root
                                        Root  Hello Max Fwd
Vlan                 Root ID            Cost  Time  Age Dly  Root Port
---------------- ------------------- ------- ----- --- ---  ----------------
VLAN0001          4097 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0002          4098 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0003          4099 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0004          4100 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0005          4101 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
```

```
VLAN0006          4102 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0007          4103 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0008          4104 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0009          4105 001b.54c2.bbc1       4    2   12   9      Ethernet2/11
VLAN0010          4106 001b.54c2.bbc1       5    2   12   9      Ethernet2/12
VLAN0011          4107 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0012          4108 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0013          4109 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0014          4110 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0015          4111 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0016          4112 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0017          4113 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0018          4114 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0019          4115 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0020          4116 001b.54c2.bbc2       4    2   12   9      Ethernet2/12
VLAN0100         32868 001b.54c2.bbc1       5    2   20  15      Ethernet2/12
Kenya#
```

Port Priority

With a well-planned root placement in the aggregation switches, manipulation of other spanning-tree parameters is seldom needed; however, in certain cases it might be necessary to manipulate port-priority to influence the forwarding path. With VLAN access ports, the port-priority applies to the VLAN to which the port belongs. For interfaces that are carrying multiple VLANs using 802.1Q trunking, a port-priority can be specified on a per-VLAN basis. The default port priority is 128.

Example 2-43 shows the configuration of port-priority on an interface.

Example 2-43 *Spanning Tree Port Priority Configuration*

```
Kenya# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Kenya(config)# interface ethernet 2/12
Kenya(config-if)# spanning-tree port-priority 64
Kenya(config-if)# exit
Kenya(config)# exit
```

Spanning-Tree Toolkit

NX-OS provides many extensions to the operation of spanning tree. When used properly these extensions can improve the resiliency, performance, and security of spanning tree. The following sections take a look at some of the specific enhancements and then discuss some basic spanning-tree port types. We conclude by combining the techniques covered in this section with some sample port-profiles for various configurations.

BPDUFilter

BPDUFilter prevents the port from sending or receiving BPDUs. A BPDUFilter is usually used with BPDUGuard to prevent an inadvertent misconfiguration that can introduce loops into the environment. In the case where BPDUGuard is not enabled, BPDUFilter still provides some safeguard against accidental misconfiguration. A port configured with BPDUFilter initially sends a series of BPDUs, if the device receives BPDUs returns to the initial port state and transitions through the listening and learning phases.

To enable BPDUFilter on all edge ports, enter the following command:

```
Congo(config)# spanning-tree port type edge bpdufilter default
```

Example 2-44 shows how to enable BPDUFilter on a specific interface.

Example 2-44 *BPDU Interface Configuration*

```
Congo(config)# int ethernet 2/21
Congo(config-if)# spanning-tree bpdufilter enable
Congo(config-if)#
Congo(config-if)# exit
```

Example 2-45 shows how to disable BPDUFilter on a specific interface.

Example 2-45 *Disabling BPDUFilter*

```
Congo(config)# int ethernet 2/21
Congo(config-if)# spanning-tree bpdufilter disable
Congo(config-if)#
```

BPDUGuard

BPDUGuard shuts down an interface if a BPDU is received. This option protects the spanning tree from unauthorized switches being placed into the topology. BPDUGuard can also be useful in protecting against host misconfiguration that could introduce a loop into the environment.

To enable BPDUGuard on all edge ports, enter the following command:

```
Congo(config)# spanning-tree port type edge bpduguard default
```

Example 2-46 shows how to enable BPDUGuard on a specific interface.

Example 2-46 *Enabling BPDU Guard on a Specific Interface*

```
Congo(config)# int ethernet 2/1
Congo(config-if)# spanning-tree bpduguard enable
Congo(config-if)#
2009 Oct 28 14:45:20 Congo %STP-2-BLOCK_BPDUGUARD: Received BPDU on port
Ethernet2/1 with BPDU Guard enabled. Disabling port.
```

```
2009 Oct 28 14:45:21 Congo %ETHPORT-2-IF_DOWN_ERROR_DISABLED: Interface Ethernet
2/1 is down (Error disabled. Reason:BPDUGuard)
```

Example 2-47 shows how to disable BPDU guard on a specific interface.

Example 2-47 *Disabling BPDU Guard on a Specific Interface*

```
Congo(config)#
Congo(config)# interface ethernet 2/21
Congo(config-if)# spanning-tree bpduguard disable
Congo(config-if)# exit
```

To decrease administrative overhead, in a dynamic environment, it might be desirable to leverage the protection provided by BPDUGuard but undesirable to require manual intervention to enable ports that have been shut down. Ports that have been disabled due to BPDUGuard can be automatically enabled after a period of time by specifying an errdisable recovery time.

Example 2-48 shows how to configure errdisable recovery.

Example 2-48 *errdisable Recovery*

```
Congo(config)# errdisable recovery cause bpduguard
Congo(config)# errdisable recovery interval 60
```

RootGuard

RootGuard protects the root placement in the bridging domain. If a port configured with RootGuard receives a superior BPDU, the port is immediately placed into an inconsistent state. RootGuard is typically implemented in the data center aggregation layer to prevent misconfigured access switches from becoming the root bridge for the entire data center aggregation block. RootGuard can be implemented only on a port-by-port basis.

Example 2-49 shows how to enable RootGuard on a specific interface.

Example 2-49 *RootGuard Configuration*

```
Congo(config)# int ethernet 2/1
Congo(config-if)# spanning-tree guard root
Congo(config-if)#
```

Now, test RootGuard by changing the priority of a Kenya to a lower value.

Example 2-50 shows RootGuard in action.

Example 2-50 *RootGuard Verification*

```
Kenya(config)# spanning-tree vlan 1 priority 0
Output from Congo
Congo# 2009 Oct 28 14:50:24 Congo %STP-2-ROOTGUARD_BLOCK: Root guard blocking port
Ethernet2/1 on VLAN0001.
Congo#
! When we remove the priority command, port connectivity is restored.
Kenya(config)# no spanning-tree vlan 1 priority 0
Output from Congo
Congo# 2009 Oct 28 14:51:19 Congo %STP-2-ROOTGUARD_UNBLOCK: Root guard unblocking
port
Ethernet2/1 on VLAN0001.
Congo#
```

Example 2-51 shows how to remove RootGuard.

Example 2-51 *Disabling RootGuard*

```
Congo(config)# int ethernet 2/1
Congo(config-if)# no spanning-tree guard root
Congo(config-if)#
```

LoopGuard

LoopGuard prevents any alternative or root ports from becoming designated ports. This situation is typically caused by a unidirectional link.

To enable LoopGuard globally, enter the following command:

```
Egypt(config)# spanning-tree loopguard default
```

Example 2-52 shows how to enable LoopGuard on a specific interface.

Example 2-52 *Enabling LoopGuard on a Specific Interface*

```
Egypt# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Egypt(config)# int port-channel 100
Egypt(config-if)# spanning-tree guard loop
```

To disable LoopGuard on a specific interface, the **no** form of this command should be used, as shown in Example 2-53.

Example 2-53 *Disabling LoopGuard on a Specific Interface*

```
Egypt# conf t
```

```
Enter configuration commands, one per line. End with CNTL/Z.
Egypt(config)# int port-channel 100
Egypt(config-if)# no spanning-tree guard loop
```

Dispute Mechanism

The 802.1D-2004 standard specifies a dispute mechanism that can prevent loops created for a variety of reasons. Two common cases in which the dispute mechanism helps are unidirectional links or port-channel misconfiguration. Dispute mechanism is enabled by default and cannot be disabled.

Bridge Assurance

Bridge Assurance is a new feature that can eliminate issues caused by a malfunctioning bridge. With Bridge Assurance, all ports send and receive BPDUs on all VLANs regardless of their state. This creates a bidirectional keepalive using BPDUs, and if a bridge stops receiving BPDUs, these ports are placed into an inconsistent state. This functionality can prevent loops that can be introduced as a result of a malfunctioning bridge. Bridge Assurance is enabled by default on any port that is configured with a spanning-tree port type network but can be disabled globally with the following command:

```
Congo(config)# no spanning-tree bridge assurance
```

To enable Bridge Assurance by setting the spanning-tree port type, enter the following commands:

```
Congo(config)# int port-channel 1
Congo(config-if)# spanning-tree port type network
```

An interesting side effect of Bridge Assurance is an automatic *pruning* function. In the topology from Figure 2-5, if a VLAN is defined on Congo but not on Egypt, Bridge Assurance puts that VLAN into a blocking state because it is not receiving BPDUs for that VLAN from Egypt. Example 2-54 demonstrates this functionality.

Example 2-54 *Bridge Assurance as a Pruning Mechanism*

```
Congo# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Congo(config)# vlan 500
Congo(config-vlan)# exit
Congo(config)# 2009 Oct 28 14:06:53 Congo %STP-2-BRIDGE_ASSURANCE_BLOCK: Bridge
Assurance blocking port Ethernet2/1 VLAN0500.
2009 Oct 28 14:06:53 Congo %STP-2-BRIDGE_ASSURANCE_BLOCK: Bridge Assurance
  blocking port port-channel100 VLAN0500.
```

After the VLAN is defined on Egypt, Bridge Assurance can detect the presence of BPDUs for that VLAN and allow it to move into a forwarding state, as demonstrated in Example 2-55.

Example 2-55 *Detecting VLAN BPDUs and Advancing in State*

```
Egypt(config)# vlan 500
Egypt(config-vlan)# exit
Egypt(config)#
Congo#
Congo# 2009 Oct 28 14:10:42 Congo %STP-2-BRIDGE_ASSURANCE_UNBLOCK: Bridge
  Assurance unblocking port port-channel100 VLAN0500.
Congo#
```

Spanning-Tree Port Types

NX-OS provides three basic switch port types that ease the administrative burden of configuring STP extensions:

- **Normal ports:** By default, a switchport is a normal port for the purpose of spanning tree. Normal ports remain unmodified and operate as standard bridge ports.
- **Network ports:** Network ports define connections between two bridges. By default, Bridge Assurance is enabled on these ports.
- **Edge ports:** Previously known as PortFast, a port configured as a spanning-tree edge denotes that the port should transition immediately into a forwarding state, bypassing the listening and learning states. Only nonbridging Layer 2 devices should be configured as edge ports. This port type should be reserved for data center hosts that cannot create a Layer 2 loop; this includes single attached hosts, Layer 3 routers and firewalls, or multihomed devices that leverage some form of NIC teaming.

Example 2-56 shows how to specify the default spanning-tree port type.

Example 2-56 *Defining Default Spanning-Tree Port Type*

```
Congo(config)#
Congo(config)# spanning-tree port type edge default
Warning: this command enables edge port type (portfast) by default on all
interfaces.
You should now disable edge port type (portfast) explicitly on switched ports
leading to hubs, switches and bridges as they may create temporary bridging loops.
! -OR-
Congo(config)#
Congo(config)# spanning-tree port type network default
Congo(config)#
```

To define the spanning-tree port type on a specific interface, enter the following commands:

```
Kenya(config)# interface ethernet 2/11
Kenya(config-if)# spanning-tree port type network
```

Virtualization Hosts

Due to the recent trend of virtualization in the data center, a hybrid of the two interface types exists as well. Although historically, 802.1Q trunks were reserved for interconnecting network devices only, virtualization hosts often require 802.1Q trunk connectivity directly to hosts. Even though these hosts tag traffic with 802.1Q headers, they are typically not true bridges and therefore can be treated as hosts and bypass the listening and learning stages of spanning-tree initialization. This configuration is sometimes referred to as *TrunkFast*.

Example 2-57 shows how to enable TrunkFast on a specific interface.

Example 2-57 *EnablingTrunkFast*

```
Kenya(config)# interface ethernet 2/40
Kenya(config-if)# switchport
Kenya(config-if)# spanning-tree port type edge trunk
Warning: Edge port type (portfast) should only be enabled on ports connected to
a single host. Connecting hubs, concentrators, switches, bridges, etc... to this
 interface when edge port type (portfast) is enabled, can cause temporary bridging
loops.
 Use with CAUTION
Kenya(config-if)#
```

Caution Virtualization techniques vary greatly; consult your vendor's documentation to determine whether this feature should be implemented.

Configuring Layer 2 Interfaces

Now that the initial spanning-tree configurations are complete, you can begin adding additional interfaces into the switching environment. The following examples discuss three different types of switchports; edge, trunk, and an edge trunk port.

Trunk Ports

Example 2-58 shows a sample configuration that would be used for access to aggregation links where multiple VLANs exist.

Example 2-58 *Standard Trunk Port Configuration*

```
interface Ethernet 2/9
switchport
switchport mode trunk
switchport trunk allowed vlan 100-103
spanning-tree port type network
```

Standard Host

Example 2-59 shows a sample configuration that would be used for standard Linux/Windows hosts that belong to a single VLAN.

Example 2-59 *Sample Access Port Configuration*

```
interface Ethernet1/7
  no shutdown
  switchport
  switchport mode access
  switchport access vlan 10
  spanning-tree port type edge
  spanning-tree bpduguard enable
  spanning-tree bpdufilter enable
```

Link to Virtualization Host

Virtualization has changed the way that network administrators must think about edge and trunk ports. In the past, physical hosts typically hosted a single MAC/IP pair, which mapped to a single VLAN. With virtualization, internal software provides some level of switching function making it possible for a virtualized host to contain many different MAC/IP pairs that might map to more than one VLAN. To perform optimally, special consideration should be made for these hosts at the physical network edge. Example 2-60 shows a sample configuration used for a virtualization host that uses an internal softswitch and guests that reside on multiple VLANs.

Example 2-60 *Sample Virtualization Host Port Configuration*

```
Kenya(config-if)# interface Ethernet1/7
Kenya(config-if)# no shutdown
Kenya(config-if)# switchport mode trunk
Kenya(config-if)# switchport trunk allowed vlan 101-103
Kenya(config-if)# spanning-tree port type edge trunk
Kenya(config-if)# spanning-tree bpduguard enable
```

Port-Profiles

Beginning in NX-OS 4.2(2), port-profiles can simplify the configuration of multiple ports that share common configuration components.

Example 2-61 shows a sample configuration of a port profile and how it is applied and verified.

Example 2-61 *Port Profiles*

```
Kenya(config)# port-profile COMMUNITY1
Kenya(config-ppm)# switchport
Kenya(config-ppm)# switchport mode access
Kenya(config-ppm)# switchport private-vlan host-association 100 102
Kenya(config-ppm)# spanning-tree port type edge
Kenya(config-ppm)# spanning-tree bpdufilter enable
Kenya(config-ppm)# spanning-tree bpduguard enable
Kenya(config-ppm)# no shutdown
Kenya(config-ppm)# state enabled

Kenya# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Kenya(config)# interface ethernet 2/28
Kenya(config-if)# inherit port-profile COMMUNITY1
Kenya(config-if)# exit
Kenya(config)# exit
Kenya# sho run int ethernet 2/28
!Command: show running-config interface Ethernet2/28
!Time: Fri Oct 30 08:52:29 2009
version 4.2(2a)
interface Ethernet2/28
  inherit port-profile COMMUNITY1
Kenya#
Kenya# sho port-profile
port-profile COMMUNITY1
 type: Ethernet
 description:
 status: enabled
 max-ports: 512
 inherit:
 config attributes:
  switchport
  switchport mode access
```

```
  switchport private-vlan host-association 100 102
  spanning-tree port type edge
  spanning-tree bpdufilter enable
  spanning-tree bpduguard enable
  no shutdown
evaluated config attributes:
  switchport
  switchport mode access
  switchport private-vlan host-association 100 102
  spanning-tree port type edge
  spanning-tree bpdufilter enable
  spanning-tree bpduguard enable
  no shutdown
 assigned interfaces:
  Ethernet2/21
  Ethernet2/28
Kenya#
```

Port-Channels

Where multiple links exist between two switches, it is often desirable to treat them as a single link from a spanning-tree perspective. The benefit of this logical bundling is that redundant physical connectivity is not blocked by spanning tree, making more bandwidth available for data traffic. Port-channels also create a level of redundancy as the failure of a physical link can no longer cause spanning tree to reconverge. The Nexus 7000 enables up to eight links to be aggregated in a port-channel. For optimal performance, it is recommended that the number of links be a power of 2 (for example, 2, 4, or 8 links). Members of a port-channel can be on the same line card, or to protect against line card failure, can be distributed across multiple modules in the system. Port-channels use various algorithms to hash frames as they arrive and load balance traffic across the physical interfaces, where any given flow always hashes to the same physical interface. A common misconception regarding port-channels is that the logical interface is a 20/40/80-Gbps link. For 10-Gbps member links, however, no single flow would exceed the transmission speed of the physical links which are members. An analogy here would be that port-channels add new lanes to the highway but do not increase the speed limit. A port-channel can be configured as either a Layer 2 or Layer 3 link depending on the requirements.

The hashing used to load balance traffic across the links is user-configurable, and the following options are available on the Nexus 7000:

- Source IP
- Destination IP
- Source MAC
- Destination MAC

- Source port
- Destination port
- Source and destination IP
- Source and destination MAC
- Source and destination port

You can configure these options globally, or because of the distributed nature of the Nexus 7000, on a line card-by-line card basis. If Virtual Device Contexts are used, these parameters are defined in the default VDC.

Example 2-62 shows how to configure and verify the load balancing algorithm.

Example 2-62 *Port-Channel Load Balancing Algorithm*

```
Congo(config)# port-channel load-balance ethernet source-dest-ip-vlan
Congo(config)# show port-channel load-balance
Port Channel Load-Balancing Configuration:
System: source-dest-ip-vlan
Port Channel Load-Balancing Addresses Used Per-Protocol:
Non-IP: source-dest-mac
IP: source-dest-ip-vlan
```

Assigning Physical Ports to a Port-Channel

Two options exist for assigning members to the logical interface:

- Configure member links to run the industry standard 802.3ad Link Aggregation Control Protocol (LACP).
- Statically configure the port as a member of the port channel. This mode is on, and no aggregation protocol information is exchanged between the devices.

There is no right or wrong method to use, and implementations vary based on personal preference. Some administrators like the environment to be deterministic and opt for the static configuration, whereas others might want to take advantage of some of the enhanced features that LACP offers. One of the benefits offered by LACP is a level of protection against misconfigured ports inadvertently becoming a member of the channel that could lead to Layer 2 loops, or black-holing data traffic. This level of protection is especially desirable in Virtual Port Channel configurations, which are discussed later in this chapter.

The **channel-group** command associates member interfaces with the port-channel. If an existing configuration is applied to the interface that makes it incompatible with the port-channel, the channel-group might be rejected from time to time. In these instances, channel compatibility can be ensured by adding the **force** command.

To assign a physical port to a port-channel without LACP, enter the following commands:

```
Egypt(config)# interface ethernet 1/4
Egypt(config-if)# channel-group 100 mode on
```

LACP is a modular process within NX-OS and must be explicitly enabled before configuration can begin. This is accomplished with the **feature** command.

When enabled, ports can be negotiated into a channel by specifying one of two modes:

- **Active:** This mode actively tries to negotiate channel membership by sending LACP packets.
- **Passive:** This mode listens for LACP packets and responds to them but does not send LACP negotiation packets.

For a link to bundle between two devices, the ports must be configured in either an active/active or active/passive fashion. If both sides of the link are configured for passive, they will not be bundled.

Example 2-63 shows how to assign a physical port to a port-channel with LACP.

Example 2-63 *LACP Configuration*

```
Congo(config)# feature lacp
Congo(config)# sho feature | inc lacp
lacp                  1         enabled

Congo(config)# interface ethernet 1/1,ethernet1/3
Congo(config-if-range)# channel-group 100 mode active
Congo(config-if-range)#
------------------------------------------------------------------------
Egypt(config)# interface ethernet 1/2, ethernet 1/4
Egypt(config-if-range)# channel-group 100 mode active
Egypt(config-if-range)#
```

During the negotiation phase, many parameters are verified to ensure that the port is compatible with the port-channel.

Example 2-64 shows the compatibility parameters that must match for a port to bundle.

Example 2-64 *Channel Compatibility*

```
Egypt# show port-channel compatibility-parameters
* port mode
Members must have the same port mode configured, either E,F or AUTO. If
they are configured in AUTO port mode, they have to negotiate E or F mode
when they come up. If a member negotiates a different mode, it will be
```

```
suspended.
* speed
Members must have the same speed configured. If they are configured in AUTO
speed, they have to negotiate the same speed when they come up. If a member
negotiates a different speed, it will be suspended.
* MTU
Members have to have the same MTU configured. This only applies to ethernet
port-channel.
* MEDIUM
Members have to have the same medium type configured. This only applies to
ethernet port-channel.
* Span mode
Members must have the same span mode.
* load interval
Member must have same load interval configured.
* sub interfaces
Members must not have sub-interfaces.
* Duplex Mode
Members must have same Duplex Mode configured.
* Ethernet Layer
Members must have same Ethernet Layer (switchport/no-switchport) configured.
* Span Port
Members cannot be SPAN ports.
* Storm Control
Members must have same storm-control configured.
* Flow Control
Members must have same flowctrl configured.
* Capabilities
Members must have common capabilities.
* Capabilities speed
Members must have common speed capabilities.
* Capabilities duplex
Members must have common speed duplex capabilities.
* rate mode
Members must have the same rate mode configured.
* 1G port is not capable of acting as peer-link
Members must be 10G to become part of a vPC peer-link.
* port
Members port VLAN info.
* port
Members port does not exist.
* switching port
Members must be switching port, Layer 2.
* port access VLAN
```

```
Members must have the same port access VLAN.
* port native VLAN
Members must have the same port native VLAN.
* port allowed VLAN list
Members must have the same port allowed VLAN list.
* port egress queuing policy
10G port-channel members must have the same egress queuing policy as the
port-channel.
* Port Security policy
Members must have the same port-security enable status as port-channel
Egypt#
```

Example 2-65 shows how to quickly verify the channel configuration with the **show port-channel summary** command.

Example 2-65 *LACP Configuration*

```
Egypt# show port-channel summary
Flags:  D - Down        P - Up in port-channel (members)
        I - Individual  H - Hot-standby (LACP only)
        s - Suspended   r - Module-removed
        S - Switched    R - Routed
        U - Up (port-channel)
--------------------------------------------------------------------------------
Group Port-       Type     Protocol  Member Ports
      Channel
--------------------------------------------------------------------------------
100   Po100(SU)   Eth      LACP      Eth1/2(P)    Eth1/4(P)
```

Logical interfaces parameters apply to all the member links, giving the administrator an easy way to manipulate multiple ports.

Note In IOS devices, port-channel interfaces are initially put into an administratively down state, whereas, in NX-OS newly created port-channels are active as soon as they are created.

Port Channel Flow Control

Flow control is supported on port-channel interfaces as well. For flow control to work properly, both sides of the port-channel must be configured. By default, the port-channel flow control is desired. Flow control can be statically configured for on or off and for each direction.

Example 2-66 shows how to configure and verify port-channel flow control.

Example 2-66 *Port Channel Flow Control*

```
Congo(config)# interface port-channel 100
Congo(config-if)# flowcontrol send on
Congo(config-if)# flowcontrol receive on
Congo# show interface port-channel 100 flowcontrol
--------------------------------------------------------------------------------
Port       Send FlowControl  Receive FlowControl  RxPause TxPause
           admin    oper     admin    oper
--------------------------------------------------------------------------------
Po100      on       on       on       on          0       0
Congo#
```

Verifying Load Distribution Across a Port-Channel

Unequal traffic distribution across physical ports can be caused for a variety of reasons, including configuration of a suboptimal load balancing algorithm, or a nonpower of 2 number of links, (for example, 3). From time to time, it is good to verify that traffic is load-balanced across all of the available members. A useful command to get a quick snapshot of these statistics is **show port-channel traffic.** Example 2-67 shows example output from this command.

Example 2-67 *Verifying Load Distribution*

```
Congo# show port-channel rbh-distribution
 ChanId    Member port    RBH values         Num of buckets
--------  -------------  -----------------  ----------------
 11        Eth2/17        4,5,6,7            4
 11        Eth1/17        0,1,2,3            4
--------  -------------  -----------------  ----------------
 13        Eth2/18        4,5,6,7            4
 13        Eth1/18        0,1,2,3            4
--------  -------------  -----------------  ----------------
 15        Eth2/25        4,5,6,7            4
 15        Eth1/25        0,1,2,3            4
francevdc1#
Congo(config)# show port-channel traffic
ChanId      Port Rx-Ucst Tx-Ucst Rx-Mcst Tx-Mcst Rx-Bcst Tx-Bcst
------ --------- ------- ------- ------- ------- ------- -------
   100    Eth1/1    53.67%   46.32%  97.39%  97.27%    0.0%    0.0%
   100    Eth1/3    46.32%    53.67%   2.60%   2.72%    0.0%    0.0%
```

In the previous output, the first command shows the number of hash buckets that each member link is assigned to, and the second shows the amount of traffic each link has forwarded. When troubleshooting port-channels, one additional task is to determine which

link a particular flow will be hashed to. As shown in Example 2-68, the **show port-channel loadbalance forwarding-path** command can be used to gather this information.

Example 2-68 *Determining Which Link a Particular Flow Will Use*

```
francevdc1# show port-channel load-balance forwarding-path interface port-channel 11
src-ip 172.16.30.25 dst-ip 192.168.10.236 module 2
Missing params will be substituted by 0's.
Module 2: Load-balance Algorithm: source-dest-ip-vlan
RBH: 0x2        Outgoing port id: Ethernet1/17
francevdc1# show port-channel load-balance forwarding-path interface port-channel 11
src-ip 172.16.30.25 dst-ip 192.168.10.235 module 2
Missing params will be substituted by 0's.
Module 2: Load-balance Algorithm: source-dest-ip-vlan
RBH: 0x5        Outgoing port id: Ethernet2/17
```

By specifying inputting in the required values based on the hash algorithm in use, you've identified that for traffic from 172.16.30.25 destined for 192.168.10.236 interface, Ethernet1/17 forwards traffic, and traffic destined for 172.16.30.25 selects Ethernet2/17 to forward traffic.

Virtual Port Channels

The Nexus 7000 and 5000 series switches take port-channel functionality to the next level by enabling links that are connected to different devices to be aggregated into a single, logical link. This technology was introduced in NX-OS version 4.1(4) and is called Virtual Port Channel (vPC). In addition to link redundancy provided by port-channels, vPC's offer some additional benefits:

- Device level redundancy with faster convergence than multiple port-channels using traditional Spanning Tree
- Further elimination of spanning-tree blocked ports by providing a loop-free topology
- Better bandwidth utilization

Caution Port-channels configured as vPCs can be used only as Layer 2 links, and no dynamic routing protocol should be used across the link.

vPCs are configured by associating two Nexus devices into a vPC domain. Within the vPC domain, information is exchanged between vPC peers across two special links:

- **vPC peer-keepalive link:** Provides heartbeating between vPC peers to ensure that both devices are online, and also to avoid active/active or split-brain scenarios that

could introduce loops into the vPC topology. The vPC peer-keepalive link can be either 1 Gbps or 10 Gbps.

- **vPC peer link:** Used to exchange state information between the vPC peers and also provides additional mechanisms that can detect and prevent split-brain scenarios.

Note The mgmt0 interface can be used as the vPC peer-keepalive link but should be avoided if at all possible. On the Nexus 7000, the mgmt0 is actually a logical interface representing the physical management port of the active supervisor. During processes such as supervisor switchover during hardware failure or In-Service Software Upgrades (ISSU), the physical link supporting the mgmt0 interface might change, causing a disruption of the keepalive messages. By using normal switch interfaces, additional levels of redundnancy in the port-channels can be used. If the mgmt0 interface is used as the peer-keepalive link, it is critical to ensure that all physical management ports are connected to an external device, such as a management switch.

The remainder of this section demonstrates configuration based on the topology illustrated in Figure 2-8.

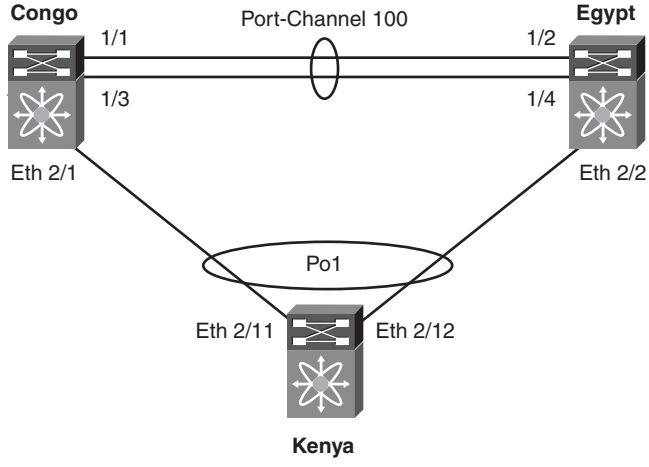

Figure 2-8 *VPC Topology*

To configure vPC, perform the following steps.

Step 1. Enable the vPC feature on each vPC peer:

```
! Congo
Congo# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Congo
Congo(config)# feature vpc
```

```
Congo(config)# exit
! Egypt
Egypt(config)# feature vpc
Egypt(config)# exit
```

Step 2. Create VRF for the VPC keepalive link:

```
! Congo
Congo(config-if)# vrf context vpc-keepalive
Congo(config-vrf)# exit
! Egypt
Egypt(config)# vrf context vpc-keepalive
Egypt(config-vrf)# exit
! Congo
Congo(config)# int ethernet 2/47
Congo(config-if)# vrf member vpc-keepalive
Congo(config-if)# ip address 1.1.1.1 255.255.255.252
Congo(config-if)# no shutdown
Congo(config-if)# exit
Congo(config)# exit
! Egypt
Egypt(config)# interface ethernet 2/48
Egypt(config-if)# no switchport
Egypt(config-if)# vrf member vpc-keepalive
Egypt(config-if)# ip address 1.1.1.2 255.255.255.252
Egypt(config-if)# no shutdown
Egypt(config-if)# exit
Egypt(config)# exit
! Congo
Congo(config-if)# vrf context vpc-keepalive
Congo(config-vrf)# exit
! Egypt
Egypt(config)# vrf context vpc-keepalive
Egypt(config-vrf)# exit
! Congo
Congo(config)# int ethernet 2/47
Congo(config-if)# vrf member vpc-keepalive
Congo(config-if)# ip address 1.1.1.1 255.255.255.252
Congo(config-if)# no shutdown
Congo(config-if)# exit
Congo(config)# exit
! Egypt
Egypt(config)# interface ethernet 2/48
Egypt(config-if)# no switchport
Egypt(config-if)# vrf member vpc-keepalive
Egypt(config-if)# ip address 1.1.1.2 255.255.255.252
```

```
Egypt(config-if)# no shutdown
Egypt(config-if)# exit
```

Step 3. Verify connectivity of the VPC peer keepalive link:

```
Congo# ping 1.1.1.2 vrf vpc-keepalive
PING 1.1.1.2 (1.1.1.2): 56 data bytes
64 bytes from 1.1.1.2: icmp_seq=0 ttl=254 time=0.958 ms
64 bytes from 1.1.1.2: icmp_seq=1 ttl=254 time=0.617 ms
64 bytes from 1.1.1.2: icmp_seq=2 ttl=254 time=0.595 ms
64 bytes from 1.1.1.2: icmp_seq=3 ttl=254 time=0.603 ms
64 bytes from 1.1.1.2: icmp_seq=4 ttl=254 time=0.645 ms
--- 1.1.1.2 ping statistics ---
5 packets transmitted, 5 packets received, 0.00% packet loss
round-trip min/avg/max = 0.595/0.683/0.958 ms

Congo(config)# vpc domain 1
Congo(config-vpc-domain)# peer-keepalive destination 1.1.1.2 source
1.1.1.1 vrf vpc-
keepalive
Congo(config-vpc-domain)# exit
Egypt(config)# vpc domain 1
Egypt(config-vpc-domain)# peer-keepalive destination 1.1.1.1 source
1.1.1.
  2 vrf vpc-keepalive
Egypt(config-vpc-domain)# exit
```

Step 4. Verify that VPC peer keepalive link is working:

```
Congo# sho vpc
Legend:
                (*) - local vPC is down, forwarding via vPC peer-link
vPC domain id                     : 1
Peer status                       : peer link not configured
vPC keep-alive status             : peer is alive
Configuration consistency status: failed
Configuration consistency reason: vPC peer-link does not exists
vPC role                          : none established
Number of vPCs configured         : 0
Peer Gateway                      : Disabled
Dual-active excluded VLANs        : -
Congo#
```

Step 5. Configure the vPC peer-link:

```
! Congo
Congo# conf t
```

```
Enter configuration commands, one per line. End with CNTL/Z.
Congo(config)# interface port-channel 100
Congo(config-if)# vpc peer-link
Please note that spanning tree port type is changed to "network" port
type on vPC peer-
link.
This will enable spanning tree Bridge Assurance on vPC peer-link
provided the STP
Bridge Assurance (which is enabled by default) is not disabled.
Congo(config-if)#
! Egypt
Egypt# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Egypt(config-if)# switchport mode trunk
Egypt(config)# interface port-channel 100
Egypt(config-if)# vpc peer-link
Please note that spanning tree port type is changed to "network" port
type on vPC peer-link.
This will enable spanning tree Bridge Assurance on vPC peer-link
provided the STP Bridge Assurance.(which is enabled by default) is not
disabled.
Egypt(config-if)#
```

Note Interfaces that are members of the VPC peer-link must be 10 GbE ports, and it is recommended that they are in dedicated rate-mde.

Step 6. Verify configuration consistency checks pass, and you have an active VPC configuration:

```
Egypt# show vpc
Legend:
                (*) - local vPC is down, forwarding via vPC peer-link
vPC domain id                     : 1
Peer status                       : peer adjacency formed ok
vPC keep-alive status             : peer is alive
Configuration consistency status: success
vPC role                          : secondary
Number of vPCs configured         : 0
Peer Gateway                      : Disabled
Dual-active excluded VLANs        : -
vPC Peer-link status
---------------------------------------------------------------------
id   Port   Status Active vlans
--   ----   ------ --------------------------------------------------
1    Po100  up     1-20,100
```

If the configuration consistency status returns anything other than success, additional information about the inconsistencies can be derived with the following command:

```
Congo# show vpc consistency-parameters global


    Legend:
        Type 1 : vPC will be suspended in case of mismatch

Name                        Type  Local Value            Peer Value
-------------               ----  ---------------------- -------------
---------
STP Mode                    1     Rapid-PVST             Rapid-PVST
STP Disabled                1     VLANs 91               VLANs 91
STP MST Region Name         1     customer               customer
STP MST Region Revision     1     1                      1
STP MST Region Instance to  1
 VLAN Mapping
STP Loopguard               1     Disabled               Disabled
STP Bridge Assurance        1     Enabled                Enabled
STP Port Type, Edge         1     Normal, Disabled,      Normal,
Disabled,
BPDUFilter, Edge BPDUGuard        Disabled               Disabled
STP MST Simulate PVST       1     Enabled                Enabled
Interface-vlan admin up     2     40-43,50,60,70-71,91,1 40-
43,50,60,70-71,91,1
                                  00-103                 00-103
Allowed VLANs               -     40-43,50,60,91,100-103 9,40-
43,50,60,91,100-1
                                  ,1000                  03,1000
Local suspended VLANs       -     -                      -
```

Step 7. Add port-channels to a VPC:

```
! Congo
Congo(config-if)# exit
Congo(config)# interface ethernet 2/1
Congo(config-if)# channel-group 1 mode active
Congo(config-if)# no shutdown
Congo(config-if)# exit
Congo(config)# interface port-channel 1
Congo(config-if)# switchport
Congo(config-if)# switchport mode trunk
Congo(config-if)# vpc 1
! Egypt
Egypt(config)# int ethernet 2/2
Egypt(config-if)# channel-group 1 mode active
```

```
Egypt(config-if)# no shutdown
Egypt(config-if)# exit
Egypt(config)# int port-channel 1
Egypt(config-if)# switchport
Egypt(config-if)# switchport mode trunk
Egypt(config-if)# vpc 1
! Kenya
Kenya(config)# interface ethernet 2/11
Kenya(config-if)# exit
Kenya(config)# interface ethernet 2/11-12
Kenya(config-if-range)# switchport
Kenya(config-if-range)# channel-group 1 mode active
Kenya(config-if-range)# no shutdown
Kenya(config-if-range)# exit
Kenya(config)# exit
Kenya(config)# int port-channel 1
Kenya(config-if)# switchport
Kenya(config-if)# switchport mode trunk
Kenya(config-if)#
```

Step 8. Verify that the vPC is operational:

```
Congo# show vpc
Legend:
                (*) - local vPC is down, forwarding via vPC peer-link
vPC domain id                     : 1
Peer status                       : peer adjacency formed ok
vPC keep-alive status             : peer is alive
Configuration consistency status: success
vPC role                          : primary
Number of vPCs configured         : 1
Peer Gateway                      : Disabled
Dual-active excluded VLANs        : -
vPC Peer-link status
---------------------------------------------------------------------
id   Port   Status Active vlans
--   ----   ------ --------------------------------------------------
1    Po100  up     1-20,100
vPC status
----------------------------------------------------------------------
id   Port   Status Consistency Reason                     Active vlans
--   ----   ------ ----------- -------------------------- ------------
1    Po1    up     success     success                    1-20,100
```

VPC Peer-Gateway

The VPC Peer-Gateway feature was introduced in NX-OS 4.2(1). This feature is designed to enable certain storage, application servers or load balancers to implement *fast-path functionality*. This causes nodes to send return traffic to a specific MAC address of the sender rather than HSRP address. By default, this traffic might be dropped as VPC loop avoidance does not allow traffic received on a VPC peer-link to be forwarded out a VPC interface (loop avoidance). A VPC Peer-Gateway enables the VPC peer device to forward packets destined for its peer router MAC locally.

To enable the peer-gateway, enter the following command:

```
Congo(config-vpc-domain)# peer-gateway
```

Unidirectional Link Detection

Full-duplex communication creates a situation in which a node can receive traffic but cannot transmit or vice versa; this condition is known as a *unidirectional link* and can be problematic for protocols that require a bidirectional exchange of information. This condition is most often attributed to fiber optic cabling as the physical medium. It is also possible for unidirectional link conditions to exist in other mediums such as twisted-pair copper cabling. Although upper layer protocols can overcome this scenario easily, this scenario can produce unexpected results at Layer 2. For these Layer 2 protocols, Cisco provides a mechanism for detecting and disabling links where bidirectional traffic flow is not possible. Cisco developed Unidirectional Link Detection (UDLD) to eliminate any negative behavior associated with the failure of bidirectional communication. UDLD enables each device to send packets to the directly connected device at periodic intervals (15 seconds by default); the sending and receiving of these packets ensures that traffic flow is bi-directional. The Cisco UDLD implementation defines two modes that devices can operate:

- **Aggressive mode:** Typically used for copper-based mediums such as twisted pair copper
- **Nonaggressive mode:** Typically used for fiber-based networks

Example 2-69 shows the configuration and verification of UDLD.

Example 2-69 *UDLD Configuration*

```
! Enabling UDLD
Congo(config)# feature udld
Congo(config)#
! Verifying UDLD global status
Congo# show udld global
UDLD global configuration mode: enabled
UDLD global message interval: 15
```

```
Congo# show udld
Interface Ethernet1/1
------------------------------
Port enable administrative configuration setting: device-default
Port enable operational state: enabled
Current bidirectional state: bidirectional
Current operational state:  advertisement - Single neighbor detected
Message interval: 7
Timeout interval: 5
        Entry 1
        ---------------
        Expiration time: 20
        Cache Device index: 1
        Current neighbor state: unknown
        Device ID: TBM12224047
        Port ID: Ethernet1/2
        Neighbor echo 1 devices: TBM12224047
        Neighbor echo 1 port: Ethernet1/1
        Message interval: 7
        Timeout interval: 5
        CDP Device name: Egypt(TBM12224047)
! Enabling UDLD aggressive mode
Congo(config-if)# udld aggressive
Congo(config-if)# exit
Congo(config)# sho udld ethernet 1/1
Interface Ethernet1/1
------------------------------
Port enable administrative configuration setting: enabled-aggressive
Port enable operational state: enabled-aggressive
Current bidirectional state: bidirectional
Current operational state:  advertisement - Single neighbor detected
Message interval: 15
Timeout interval: 5
        Entry 1
        ---------------
        Expiration time: 44
        Cache Device index: 1
        Current neighbor state: bidirectional
        Device ID: TBM12224047
        Port ID: Ethernet1/2
        Neighbor echo 1 devices: TBM12224047
        Neighbor echo 1 port: Ethernet1/1
        Message interval: 15
        Timeout interval: 5
```

```
        CDP Device name: Egypt(TBM12224047)

Congo# show udld neighbors
Port             Device Name    Device ID   Port ID         Neighbor State
--------------------------------------------------------------------------
Ethernet1/1      TBM12224047    1           Ethernet1/2     bidirectional
Ethernet1/3      TBM12224047    1           Ethernet1/4     bidirectional
```

Summary

The Layer 2 switching capabilities of NX-OS provide a scalable, resilient foundation for your data center. This chapter covered the following topics:

- VLANs segment traffic at Layer 2 to address security concerns or define failure domains.
- PVLANs provide an additional level of security by enabling administrators to subdivide VLANs.
- Spanning Tree provides a mechanism to ensure that physically redundant networks are logically loop-free. You learned how to configure and tune Rapid-PVST+ and MST.
- Enhancements to Spanning Tree such as BPDUGuard, BPDUFilter, RootGuard, and LoopGuard.
- Links can be aggregated through the use of port-channels and vPCs to simplify Spanning Tree topologies while making additional bandwidth available for data traffic.

With NX-OS on the Nexus 7000 and 5000, Cisco provides a highly available implementation of standards-based Layer 2 technology and builds upon it with new innovation required to meet the specific demands within today's data centers.